Approximating inverse submatrices for parallel finite element preconditioning

M.C. Dracopoulos

Department of Numerical Mathematics

Note NM-N9402 September 1994

Approximating Inverse Submatrices for Parallel Finite Element Preconditioning

Michael C. Dracopoulos

CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Abstract

A preliminary investigation on building preconditioners based on approximate inverses of certain sub-blocks of the coefficient matrix is given. The derived preconditioners are suitable for the iterative solution of finite element problems and they are closely related to the finite element discretization. Some initial results are also given.

AMS Subject Classification (1991): 65F10, 65F50, 65N30, 73Kxx
CR Subject Classification (1991): G.1.3, G.1.8, J.2
Keywords & Phrases: Parallel conjugate gradient algorithms, finite elements, hierarchical preconditioning, coarse/fine mesh preconditioning, Quasi-Newton methods

1. Introduction

Iterative methods have long been established as alternatives to direct techniques for the solution of finite element problems. Apart from their desirable properties for conventional computer architectures [2, 4] (related mainly to the exploitation of the matrix sparsity), recent interest in iterative solvers has been related to two additional issues: the inherently parallel nature of the iterative solution (as opposed to elimination methods) which makes them ideal candidates for parallel processing [4, 12, 25, 33], and the on-going research on multigrid ideas [29].

However, the successful application of iterative methods is by no means guaranteed since their convergence, which depends on the eigenvalue spectrum of the problem, is notoriously unreliable. Several *preconditioning* techniques have been proposed, that attempt to overcome these difficulties by transforming the original problem

$$Kx = f \tag{1.1}$$

into

$$M^{-1}Kx = M^{-1}f$$

by choosing M to be an approximation to K.

In structural analysis, K is the structure *stiffness* matrix, x are the discretized *displacements* and f is the *external force* vector. Furthermore, in finite element analysis, full assembly of K is not required for the iterative solution of (1.1). Instead, only element information can be used in the implementation of an iterative solver on (1.1). This is particularly advantageous in parallel computations and dictates a certain *element-level* approach to finite element parallelization. Unfortunately, element information is not enough to define "traditional" *global*

Report NM-N9402
ISSN 0169-0388
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

preconditioners M which, therefore, become difficult to parallelize. Element level preconditioners are an alternative, but so far little work has been done towards this direction.

The present work focuses on another option for finite element preconditioning, namely, the derivation of *global* preconditioners from entirely *element-level* information. Preconditioners which are closely linked to the finite element idealization, fall easily within this framework. They are based either on a hierarchical [3, 37], or a coarse/fine mesh formulation [14, 15]. However, although these preconditioners can be *derived in parallel*, their *implementation* within the iterative solution is *not entirely* parallel, since it requires some form of direct solution. This report investigates the possibility of avoiding this "bottleneck" by working with an approximate inverse of the preconditioning matrix instead. In particular, only certain sub-blocks of M need to be approximately inverted. Two techniques for approximate inversion are examined here, that are highly parallelizable using only element information.

The organization of this report is as follows. Section 2 describes an element-by-element approach to parallelism with emphasis on their implementation on message passing architectures. The hierarchical and the coarse/fine preconditioner are described in Section 3. Section 4 describes a technique for constructing approximate inverses based on norm minimization, while Section 5 presents some optimization techniques that are also suitable for such a task. Finally, some initial results and concluding remarks are given in Sections 6 and 7 respectively.

2. Finite element parallelism in conjugate gradients

A natural approach to parallelizing a finite element iterative solution scheme stems from the concept of the "element" as a building block in the finite element methodology. Elements are treated as dejoint and decoupled in many stages of finite element computations (namely the formation of element level characteristics, such as element stiffness, strains and stresses) which are therefore inherently parallel in nature. In addition, when an iterative solver is employed, the explicit formation of the global stiffness matrix can be avoided by inducing the element connectivity information to the out-of-balance force (residual) vector. In other words, matrix-vector multiplication with the global stiffness matrix can alternatively be replaced by element level matrix vector multiplications as long as the out-of-balance force vector is properly updated. The above reasons suggest that an element oriented problem partitioning and processor mapping can be particularly efficient in parallel finite element computations. It should be emphasized that in this report, the term "element" is used in a broad sense and can also signify substructures (subdomains) or "superelements" (subdomain contributions to some reduced interface problem) [13].

2.1 Data parallel systems

Little need to be said about the iterative solution of finite element problems on data parallel systems. The main idea is to work with *global* (structure level) vectors and *element* matrices. This guarantees that the global vectors are properly updated and that the global matrix-vector multiplication involved can be performed on an element-by-element basis.

2.2 Message passing systems

The "locality" underlying a message passing system suggests a different processor mapping whereby *both* element matrices *and* element vectors are kept and take part in the solution procedure. Each processor is associated with all calculations for a particular "element" and the structured connectivity information is induced through the network topology by means of communicating information across (in order to keep the element vectors properly updated).

For the solution of the physical problem described by (1.1), each processor is assigned an element and all the related element vectors. For the sake of formality, it is assumed here that local (substructure) vectors and matrices are expanded to the global (structure) size, by padding the redundant positions with zeroes. The global displacement vector x in (1.1) is distributed in a natural sense among the N_p (number of elemens) processors, i.e., foreach degree-of-freedom i present in element e:

$$x_e(i) = x(i), \qquad e = 1, \dots, N_p. \tag{2.2}$$

This can be viewed as satisfying the compatibility requirements for each interface node, in the sense that the interface displacements are common to any two adjacent elements, and it is exactly those interface displacements that are stored in more than one processors. On the other hand, the entire force vector f exists (theoretically) distributed among the subdomains

$$f = \sum_{e=1}^{N_p} f_e. \tag{2.3}$$

This follows from the equilibrium relation for each interface node, where it is natural to expect that the interface related entries in f can be distributed in any manner among the substructures as long as they sum up to the external force for each node under consideration. The various vectors arising in a parallel implementation of iterative solvers are derived either from x or from f and they, therefore, fall under two categories: *displacement*-type vectors that are partitioned according to (2.2) and *force*-type vectors that satisfy (2.3).

In their standard form, non-stationary iterative methods [19], such as the conjugate gradient method, are built around three basic operations:

1. *Matrix-vector multiplication.* This operation is normally required as an intermediate step for computing the residual vector and involves the coefficient matrix K and a displacement-type vector and results to a force-type vector. The global matrix K is the result of a finite element assembly operation

   $$K = \sum_{e=1}^{N_p} K_e.$$

   Bearing in mind that the effect of multiplying a local displacement-type vector y_e (expanded to the full problem size) by an element stiffness matrix K_e is as if the global vector y is involved instead, i.e.

   $$K_e y_e = K_e y,$$

   a global matrix vector multiplication can be replaced by *entirely* local operations:

   $$z = Ky = \sum_{e=1}^{N_p} K_e y = \sum_{e=1}^{N_p} K_e y_e = \sum_{e=1}^{N_p} z_e.$$

2. *Linked triads:*

$$z := z + \alpha y$$

where y, z are vectors and α a scalar. Global linked triad operations can be replaced by local ones

$$z_e := z_e + \alpha y_e \tag{2.4}$$

provided that the scalar quantity α is known to all processors. If both y and z are displacement or force type vectors, then (2.4) is an entirely parallel operation. On the other hand, if *both* y and z are *not* partitioned in a similar manner then it is necessary to coerce the non-conforming vector to the type of the left-hand-side vector before computing (2.4). Such a coercion involves a local interprocessor communication, i.e. communication among processors assigned to adjacent subdomains. In a parallel implementation of the conjugate gradient method, for example, only one local communication per iteration is required, with all other linked triads involved being completely parallel (Figure 0.1).

3. *Inner products.* Each iteration of an iterative solver requires the evaluation of a number of dot products. The trick here is to always enforce this kind of operations between a displacement-type (y) and a force-type vector (z), and using (2.2) and (2.3) to get

$$\tau = z^T y = \sum_{e=1}^{N_p} z_e^T y = \sum_{e=1}^{N_p} z_e^T y_e = \sum_{e=1}^{N_p} \tau_e.$$

The computation of inner products very much depends on the specific processor interconnection scheme and requires interprocessor communication. For some topologies, such as the hypercube it is possible to implement an inner product operation with a number of local communications [20, 32] while in other configurations, the inner product computations require a global type communication whereby the local contributions τ_e are first *gathered* and added together on a single processor and then the result is *broadcasted* back to each individual processor.

A typical conjugate gradient algorithm, running on each processor is shown in the flowchart in Figure 0.1, where the local and global communications are indicated by thick bordered boxes.

A more detailed discussion on the parallel implementation of the conjugate gradient method based on a substructuring approach is given in [5, 30].

3. Finite element preconditioning

As it was shown in the preceding section the key aspect in parallelizing the finite element method is the element level (in a broad sense) orientation of the solution procedure. Sadly, the most successful preconditioners for conventional computers, such as incomplete factorization [1, 19, 35] and SSOR techniques [21, 34] can only be constructed from *global matrix* information. In addition, their underlying structure is inherently *sequential* [4, 33].

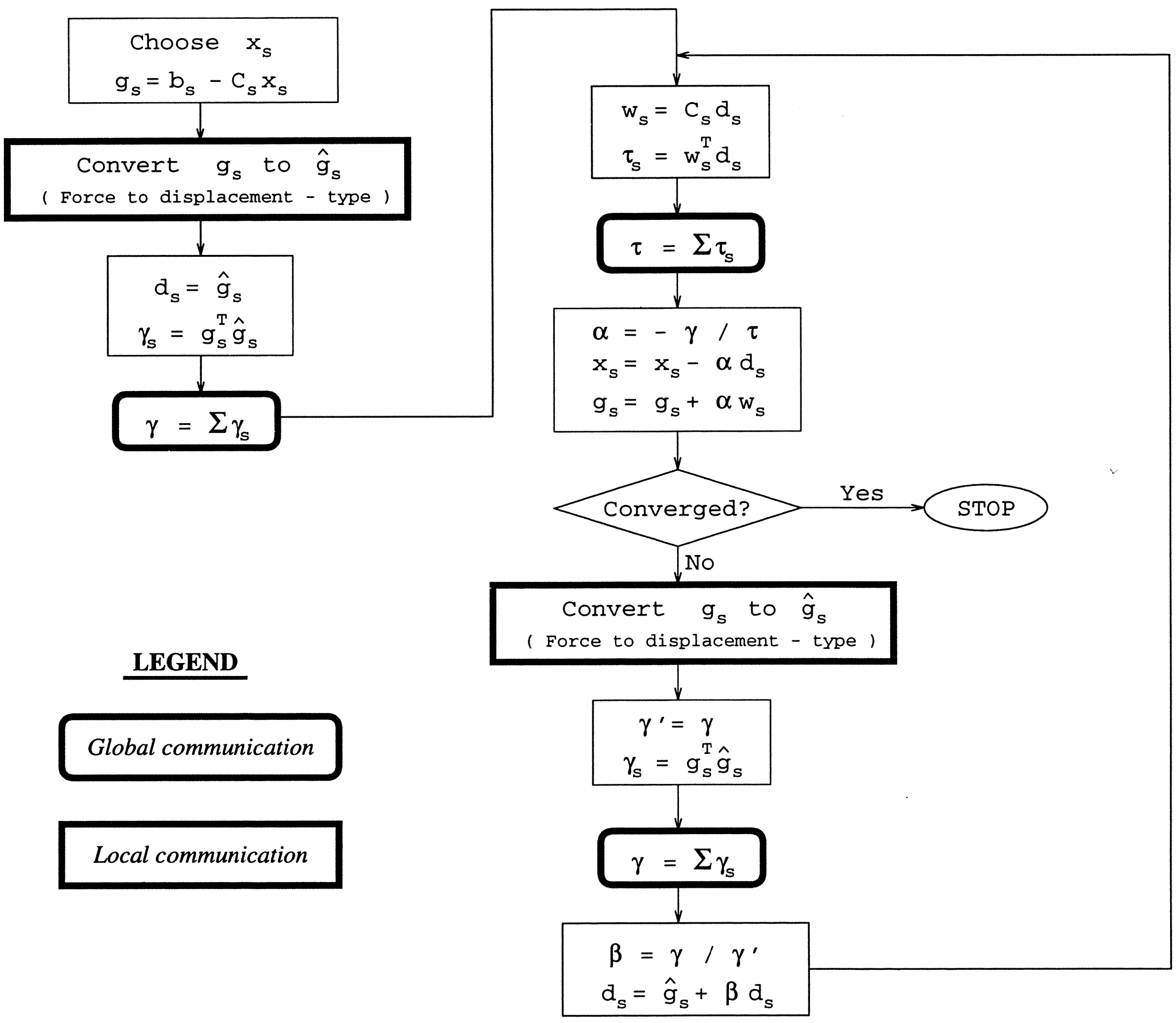

Choose x_s
$g_s = b_s - C_s x_s$
Convert g_s to $\hat{g}_s$
(Force to displacement - type)
$d_s = \hat{g}_s$
$\gamma_s = g_s^T \hat{g}_s$
$\gamma = \Sigma \gamma_s$
$w_s = C_s d_s$
$\tau_s = w_s^T d_s$
$\tau = \Sigma \tau_s$
$\alpha = - \gamma / \tau$
$x_s = x_s - \alpha d_s$
$g_s = g_s + \alpha w_s$
Converged?
Yes
STOP
No
Convert g_s to $\hat{g}_s$
(Force to displacement - type)
$\gamma' = \gamma$
$\gamma_s = g_s^T \hat{g}_s$
$\gamma = \Sigma \gamma_s$
$\beta = \gamma / \gamma'$
$d_s = \hat{g}_s + \beta d_s$
LEGEND
Global communication
Local communication

Figure 0.1: Parallel conjugate gradient algorithm for processor s

Ongoing research on element-by-element preconditioners [26, 23, 18] has shown some promising results for certain classes of problems. On the other hand, since element preconditioners span a fairly recent and limited "research lifetime", they lack in robustness and general applicability when compared with global matrix techniques.

A different approach to finite element preconditioning has been adopted in the present work. The focal point is on *global preconditioners* implemented with element level information. This differs from "pure" element preconditioners in the sense that instead of element level preconditioning matrices, approximate inverses of *global preconditioners* are constructed from element information. The preconditioning operation now involves a highly parallel matrix-vector multiplication rather than the conventional solution of the preconditioning system. It should be noted, that in certain cases, the full preconditioner does not necessarily have to be explicitly assembled. Another particular of the current work is the strong link of the adopted global preconditioners with the finite element discretization procedure, namely hierarchical and coarse/fine mesh formulations.

3.1 Hierarchical preconditioning

If hierarchical shape functions [8] are used to interpolate the displacements within a finite element domain, the displacement function can be expresses in terms of the discretized displacements as

$$u = N_l^T x_l + N_h^T x_h$$

where N_l and N_h are the lower and higher order shape functions respectively, and x_h involves only *relative* nodal displacements.

Under the hierarchical formulation the governing stiffness equations (on the element or the structure level) take the form:

$$\begin{pmatrix} K_{ll} & K_{lh} \\ K_{hl} & K_{hh} \end{pmatrix} \begin{pmatrix} x_l \\ x_h \end{pmatrix} = \begin{pmatrix} f_l \\ f_h \end{pmatrix}$$

where K_{ll} and K_{hh} represent the lower and the higher order stiffness elements respectively, while K_{lh} and K_{hl} represent the coupling between higher and lower order variables.

Since the lower order variables are clearly more significant, one could use as a preconditioning matrix

$$M = \begin{pmatrix} K_{ll} & 0 \\ 0 & D_{hh} \end{pmatrix} \tag{3.5}$$

with D_{hh} being the diagonal part of K_{hh}. It is clear that M is positive definite since K_{ll}, being a stiffness matrix in its own right, is positive definite.

This preconditioning technique was applied by Crisfield [7] in conjunction with a 2-parameter conjugate gradient-like iteration [6]. Various implementations of this method can also be found in [36, 37] as well as in [3] where some of the detailed mathematical aspects are considered.

Hierarchical preconditioning is both successful and popular in finite element analyses but it can only be implemented where hierarchical bases have been introduced.

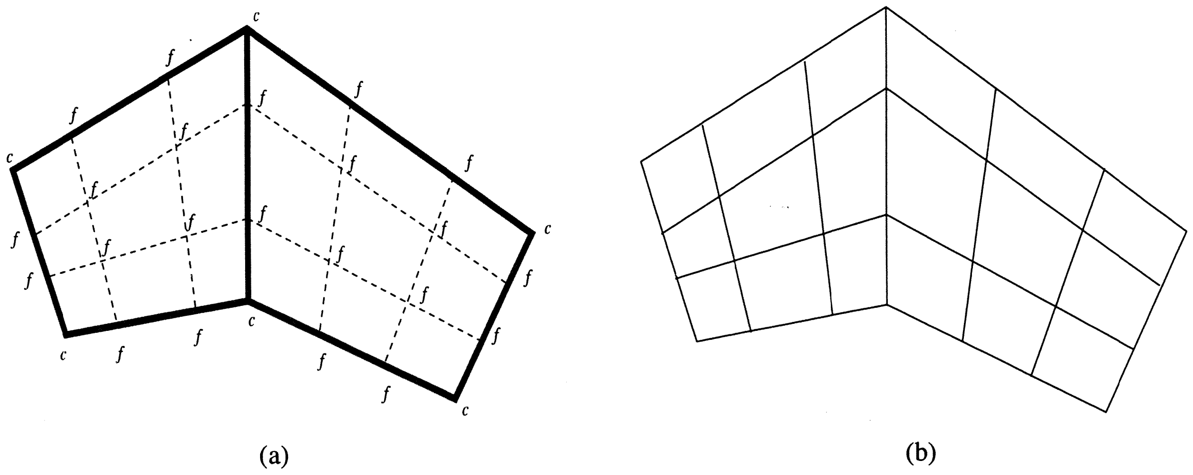


Figure 0.2: Initial coarse mesh and derived fine mesh

3.2 Coarse/fine mesh preconditioning
The hierarchical method can be viewed as a special coarse/fine mesh generalization [8]. The former procedure is the easier to implement while the latter is more general and can be applied without directly implementing hierarchically based displacement functions. In addition it is more flexible since it can be implemented with various levels of coarseness.

A typical finite element idealization over a domain (Figure 0.2(b)) can be viewed as a refinement of an initial coarse mesh discretization (Figure 0.2(a)). Alternatively, one can always specify an auxiliary coarse mesh on top of a fine mesh discretization. If a distinction is made between nodal quantities common to both meshes (referred to as "coarse" variables herein) and variables that are present only in the fine mesh ("fine" mesh variables in this context) and the "fine" mesh displacements $\bar{p}_f$ are expressed in terms *relative* to the coarse mesh variables p_c, the finite element discretization procedure yields

$$u = N_C^T x_c + N_F^T \bar{x}_f \tag{3.6}$$

where u are the displacements and N_C and N_F are the standard shape functions for the coarse and the fine mesh respectively. Separating "coarse" from "fine" mesh variables, the global stiffness equations (1.1) can be written as

$$\bar{f} = \begin{pmatrix} \bar{f}_c \\ f_f \end{pmatrix} = \begin{pmatrix} K_{cc} & \bar{K}_{cf} \\ \bar{K}_{fc} & K_{ff} \end{pmatrix} \begin{pmatrix} x_c \\ \bar{x}_f \end{pmatrix} = \bar{K}\bar{x}. \tag{3.7}$$

As shown in [14], the coarse stiffness block K_{cc} is exactly the full stiffness matrix of the coarse mesh (Figure 0.2(a)), while the "fine" mesh stiffness block K_{ff} is unaffected by the introduction of relative degrees-of-freedom and is identical in both K and $\bar{K}$ (if the former is written in a partitioned form).

The above approach to the coarse/fine mesh procedure is closely linked to the hierarchical formulation. As shown in [14], however, the stiffness equations (3.7) can alternatively (and more conveniently) be derived from the standard formulation, without considering (3.6), by introducing appropriate *transformation* matrices that connect the displacements p and $\bar{p}$ in both bases:

$$x = \begin{pmatrix} x_c \\ x_f \end{pmatrix} = \begin{pmatrix} I & 0 \\ T_{fc} & I \end{pmatrix} \begin{pmatrix} x_c \\ \bar{x}_f \end{pmatrix} = T\bar{x}. \tag{3.8}$$

The submatrix T_{fc} is obtained by evaluating the coarse mesh shape functions N_C at the coordinates of the "fine" mesh points

$$T_{fc}(f,c) = N_C(c) \quad \text{computed at position } f. \tag{3.9}$$

This procedure is only applied once at the beginning of the analysis. The "relative" stiffness matrix $\bar{K}$ in (3.7) and the equivalent force vector $\bar{q}$ can be obtained from the standard fine mesh stiffness matrix K and force vector q (provided that the "coarse" and "fine" mesh variables have been reordered) as transformation operations

$$\bar{K} = T^T K T, \qquad \bar{f} = T^T f$$

which can be applied on either the structure or the element level. Once the stiffness equations (3.7) are solved, the "relative" displacements $\bar{x}$ can be transformed to x by (3.8).

The derivation of the transformation matrices T_{fc} and the transformed system (3.7) are highly parallel operations if a substructure based problem partitioning is adopted. In a parallel computer environment, each processor is assigned a substructure defined by one or more coarse elements. Then the operations for defining the transformation matrices for the fine mesh elements (or nodes) within each substructure can proceed independently and in parallel.

If an iterative solver is applied to (3.7) then starting from the above coarse/fine mesh formulation various preconditioners can be constructed as discussed in [14]. A particular case which favours parallel processing is to define an approximation to the "relative" stiffness matrix $\bar{K}$ by

$$M = \begin{pmatrix} K_{cc} & 0 \\ 0 & D_{ff} \end{pmatrix} \tag{3.10}$$

where K_{cc} is the stiffness matrix of the coarse mesh discretization and D_{ff} the diagonal of K_{ff} in (3.7). It is important that the preconditioner (3.10) does not involve an approximation on K_{cc} because this matrix provides a valid coarse mesh solution and the induced strain energy $1/2 x_c^T K_{cc} x_c$ will dominate the fine mesh solution.

3.3 Remarks on implementation

In both the the coarse/fine mesh (3.10) and hierarchical preconditioner (3.5) , the coarse mesh (low order) variables are completely decoupled from the fine mesh (high order) variables and the preconditioning operation of an iterative solver on the residual g

$$z = M^{-1} g$$

can be carried out in two independent steps:

$$z_f = D_{ff}^{-1} g_f \tag{3.11}$$

for the fine mesh (or the high order) variables, and

$$z_c = K_{cc}^{-1} \bar{g}_c \tag{3.12}$$

for the coarse mesh (or the low order) variables.

Clearly, in a multiprocessor environment, the "fine" mesh preconditioning (3.11) that is just diagonal scaling is also a "perfectly" parallel operation. The construction of the *coarse* part of the preconditioner, on the other hand, requires a direct factorization of K_{cc} and furthermore the distributed implementation of (3.12) involves a forward and backward substitution. These operations are inherently sequential, since they are governed by the same problems as the parallelization of a direct solver: namely, high interprocessor communication and load imbalancing.

A way out of these problems was investigated in [15], whereby the factorization of the coarse stiffness block and the forward/back substitution in the preconditioning step of each iteration could be performed *sequentially* on a single dedicated processor. This serial implementation of the factorization and the preconditioning step in a parallel environment, also requires global communication operations between the central processor and the parallel nodes. For the above preconditioners these operations involve only a relative small part of the global stiffness matrix (especially in the case of coarse/fine mesh preconditioning where the size of the coarse mesh can be easily adjusted to the particular needs for an efficient implementation of the global communication operation, by choosing a suitable level of discretization for the selected coarse mesh).

On the contrary, this switch to sequential processing is not possible when implementing other types of preconditioners, since they usually involve operations on a full problem scale. This is true even for incomplete Choleski preconditioners where the rejection of the fill-in terms depends on their magnitude [35]: although they can end up with a preconditioner of any size between diagonal and the full factors, they still require the factorization procedure to be applied on the full stiffness matrix.

Another option that will be further investigated in the present work is to approximate the *inverse* of the coarse (low order) stiffness block and apply this directly to (3.12). Clearly, a successful application of this family of finite element preconditioners heavily relies on a precise representation of the coarse stiffness block K_{cc} (or K_{ll} respectively) in the preconditioning matrix M. Therefore it is crucial that the method generating the approximate inverse of K_{cc} to be as accurate and reliable as possible. This rules out the case of polynomial type approximations to the inverse of a matrix [19] based on the expansion of the Neumann series. Although such an approximation can be generated by element level matrix-vector multiplications, it presents an extremely high operation count and the commonly used truncation to about the fourth term in the series, yields an approximation to the inverse which is usually not better than the diagonal approximation itself [4, 13]. Some more promising alternatives will be subsequently presented.

A final issue in this approach is related to the fact that inverses of even very sparse matrices are usually dense. However, bearing in mind that the size of the coarse mesh is usually only a small fraction of the full scale problem, a fact that in many cases does not seriously affect the efficiency of the preconditioner. If τ is the density factor of the coefficient matrix (τN^2 number of nonzeros in a N-by-N matrix) and n is the order of the coarse mesh problem, then for

$$n/N \approx \sqrt{\tau}$$

a fully dense approximate inverse of the coarse submatrix yields a preconditioner with the same number of nonzeros as the coefficient matrix. This is normally an acceptable size as far as storage demands and operation count are concerned. Typical values of τ from problems encountered in finite element analysis on regular grids range from $15/N$ (for 2-dimensional triangular element problems) to $243/N$ (for 3-dimensional 20-node brick problems).

4. Approximate inverses based on some norm minimization

Recently various techniques have been presented in the literature, that attempt to construct approximations to the inverse of a matrix to be used for preconditioners. Their common point is that the approximate inverse is obtained as the solution to some norm minimization problem

$$\min \|KM - I\| \tag{4.13}$$

most often the Frobenius norm. The choice of the latter is justified by the fact that the l_1 or l_2 norms are considerably more expensive to be minimized. Moreover, it is well known that, the convergence behaviour of the conjugate gradient method, for instance, is particularly sensitive to the eigenvalue distribution of the coefficient matrix [24], and minimizing the Frobenius norm should cluster most eigenvalues in the vicinity of 1, as it is observed in [22]. A final reason for choosing this particular norm is that it enables the solution of (4.13) to be obtained in a highly parallel fashion. More specifically, since

$$\|KM - I\|_F^2 = \sum_{i=1}^{N} \|(KM - I)e_i\|_2^2$$

the solution of (4.13) in the Frobenius norm degenerates into N independent least squares problems

$$\min_{m_i} \|Km_i - e_i\|, \quad i = 1, \ldots, N \tag{4.14}$$

with m_i, e_i being the ith column of the preconditioning and the identity matrix respectively.

Naturally, the solution to (4.14) yields a *dense* approximate inverse (since it simulates, in a sense, the exact inverse which is usually dense even for very sparse cases) and can only be treated effectively if a sparsity pattern is enforced upon it. Not strangely enough, these methods share some common approaches with incomplete factorization techniques, in the way they treat fill-in elements: the generated matrix elements are included in the final preconditioner either if they fall within a prescribed sparsity pattern [27], or if they are "large" in magnitude [22]. A justification for the latter, is given in [9] where it is observed that very often most of the entries in the exact inverse K^{-1} are indeed very small.

In the present work, such a technique was tested at some stage for preconditioning the global stiffness equations. More specifically, a preconditioner was constructed based on the solution of a 2 dimensional structural analysis problem discretized with 8-node plane stress elements (similar to the ones described in Section 6 that follows). The preconditioner was derived as the solution to (4.14) for the fully assembled stiffness matrix, with the sparsity pattern for M being the same as that of K. This was required for a possible future application within an element-by-element framework: where such an approximate inverse could be stored implicitly as a number of element level matrices M_e and be used for element-by-element matrix vector multiplications. Symmetry was also imposed on the derived matrix (in order to be used with the conjugate gradient method) by setting it equal to $(M+M^T)/2$. Unfortunately, for this type of test problem (which is representative of a commonly encountered class of engineering problems involving 2-dimensional structured meshes) the resulting matrix was singular and further investigation towards this direction was abandoned. As is proved in [22], if the predefined sparsity pattern of M generates residuals to the solution of (4.14) that satisfy

$$r_i = \|Km_i - e_i\| < \varepsilon, \quad 1,\ldots,N \tag{4.15}$$

and $\sqrt{N}\varepsilon < 1$ or $\sqrt{p}\varepsilon < 1$ (with p being the maximum number of nonzeros over all r_i) it can be guaranteed that the resulting matrix will be nonsingular. This demonstrates that the success of approximate inverse preconditioning with fixed sparsity patterns is extremely problem dependent.

A recently developed adaptive procedure for controlling the magnitude of the "fill-in" elements is described in [22], and makes use of a drop-off parameter ε to check if the current sparsity pattern enforced to the solution of (4.14) is acceptable or not: If (4.15) is not satisfied and the maximum level of allowed fill-in has not been reached, then extra positions are allowed to be filled in M and the whole procedure described by (4.14) is updated to accommodate the new sparsity pattern.

Adaptive schemes like the one above, that allow the sparsity pattern of M to be progressively augmented, are expected to be robust and generally applicable provided that there is a certain amount of flexibility in the available storage. A natural extension to these methods would involve their application for approximating the inverse of the coarse (or low order) stiffness block in the coarse/fine mesh (or hierarchical) preconditioner previously described. As was explained in the preceding paragraph, in many cases, only a very small fraction of the full scale problem is required to define the coarse mesh preconditioner without loss in the overall efficiency. The approximating inverse of K_{cc}, therefore, can either be a fully dense matrix (where possible) or one derived from an adaptive technique such as the ones described above. However, an investigation towards this direction has not been undertaken at present.

5. Approximate inverses based on optimization techniques

The solution of the stiffness equations can be viewed as a minimization procedure, satisfying a condition of minimum potential energy at equilibrium [8]. The potential energy is expressed in terms of a quadratic function

$$\Phi = \frac{1}{2}x^T K x - f^T x + c. \tag{5.16}$$

This is a special case of unconstrained optimization which can be treated with a *Quasi-Newton* type procedure [10, 17, 28]. The underlying idea behind Quasi-Newton (also known as *variable* *metric methods*) is to generate successive approximations to the inverse of the Hessian matrix K based on information obtained in each iteration. The basic structure of a Quasi-Newton iteration applied to the quadratic function can be described as:

Step 1 Calculate direction of search:

$$d_k = -H_k g_k$$

with H_k being the current approximation to the inverse Hessian, and g_k the gradient vector of (5.16).

Step 2 Set $q_k = K d_k$ and calculate step length:

$$\alpha_k = -\frac{g^T d}{d^T q}$$

Step 3 Update minimum and residual vector by:

$$x_{k+1} = x_k + \alpha_k d_k \tag{5.17}$$

$$g_{k+1} = g_k + \alpha_k q_k \tag{5.18}$$

Step 4 Update inverse Hessian approximation H_k to get H_{k+1}.

Usually the update of the inverse Hessian in Step 4 above, involves either *rank-1* or *rank-2* updates. It is generally accepted that the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update defined for quadratics as

$$H_{k+1}^{BFGS} = H_k + (1 + \frac{q_k^T H_k q_k}{\gamma_k})\frac{d_k d_k^T}{\gamma_k} - \frac{d_k q_k^T H_k + H_k q_k d_k^T}{\gamma_k} \tag{5.19}$$

with

$$\gamma_k = q_k^T d_k$$

exhibits performance superior to that of other update formulas and forms the basis for the so-called Broyden family of updates. In exact arithmetic, N steps of the BFGS update will generate the exact inverse.

A consequence of the *Dixon's theorem*[11] on the quadratic case states that the *preconditioned* conjugate gradient (with preconditioner M) and the BFGS Quasi-Newton algorithm are identical for the same starting point x_0 and symmetric positive definite $H_0 = M$, in the sense that they generate the same direction of search vectors d_k [31]. Therefore the linked triad updates (5.17) and (5.18) are exactly those found in the conjugate gradient algorithm. Although not explicitly needed, the information required to build approximations to the inverse of the coefficient matrix K is, therefore, present in the conjugate gradient procedure and can be defined by the vectors d_k and q_k generated in each iteration.

Aspect ratio	$10\ (\kappa = 1.3 \times 10^{12})$		$20\ (\kappa = 2.7 \times 10^{12})$	
n/N	0.1	0.2	0.1	0.2
$\kappa(K_{cc}^{-1}\bar{K})$	1.8×10^5	4×10^4	1.8×10^6	4.3×10^5
$\kappa(H_{cc}\bar{K})$	1.8×10^5	4×10^4	1.3×10^8	2.8×10^6

Table 0.1: Condition numbers

The above approach can be used to approximate the inverse of the coarse stiffness block K_{cc} for the coarse/fine mesh preconditioner (3.10) (or the hierarchical preconditioner). An initial application of a diagonally preconditioned conjugate gradient solver can be applied to the coarse stiffness equations in order to construct an approximate inverse of K_{cc}, to be subsequently used to precondition the subsequent fine mesh solution. Usually, the approximate inverse could be explicitly formed, (since, as was shown in Section 3.3, the storage and operation count considerations can be minimal when applied to the coarse mesh problem). Alternatively, the matrix-vector multiplication involved in the coarse mesh preconditioning (3.12) can be formed explicitly, using the diagonal of K_{cc} (here equal to the initial inverse Hessian in Quasi-Newton terminology) and the vectors d, q and scalars γ necessary to define the rank-2 updates in (5.19). The latter approach is particularly desirable when storage considerations are at premium. More specifically, if only $2mn$ positions are available (instead of n^2), it is possible either *(i)* to stop updating H after the mth iteration and set $M = H_{m+1}$, or *(ii)* to reset $H_{m+1} = \text{diag}(K_{cc})$ and proceed collecting the $m+1, \ldots, 2m$ subsequent vectors for the definition of H.

It is clear, that the condition number of the coarse mesh stiffness K_{cc} affects how close the BFGS approximation will be to the exact inverse. On the other hand, numerical evidence shows that the convergence behaviour of the coarse/fine mesh preconditioner is affected by the conditioning of the coarse mesh. In other words, for two different coarse meshes with the same number of elements, it is the well-conditioned case that should be used for this type of preconditioning (mesh conditioning is mainly affected by the aspect ratio of the individual elements [16]). The coarse/fine mesh preconditioner with approximate inverses from (5.19), therefore, is likely to be more effective in cases where an initial "well-conditioned" mesh is further refined to a "badly-conditioned" final mesh.

6. Some preliminary results

The coarse/fine mesh preconditioner with BFGS approximation on the coarse mesh has initially been tested on two cantilever beam examples with aspect ratios equal to 10 and 20. The test examples and the various discretizations used for the auxiliary coarse meshes are given in Figure 0.3 for the example with aspect ratio 10. Both test cases were discretized using 8-node plane stress elements for both the fine and the coarse mesh. This rather simple structure was chosen because it can yield increasingly ill-conditioned problems as a result of an increase in its aspect ratio. The condition number for the test cases as well as the condition numbers of the scaled coefficient matrix are given in Table 0.1. The stopping tolerance for the conjugate gradient solver was set to $\|g\|/\|f\| < 10^{-6}$, where g and f are the residual and right-hand-side vector respectively. The convergence history for the test case with aspect ratio 10 is given in Figure 0.4 for the diagonal and the coarse/fine mesh preconditioners (the

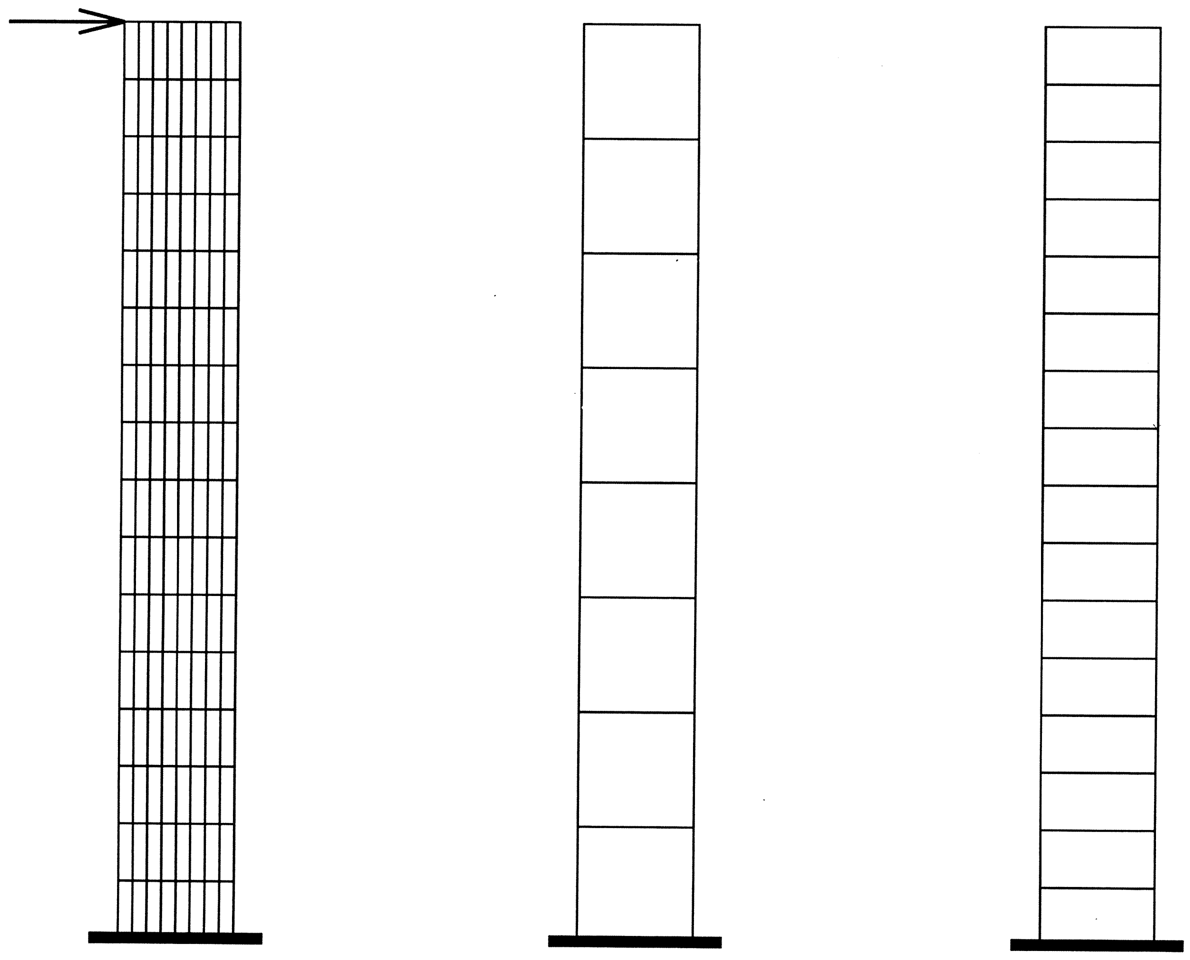


Figure 0.3: Fine and coarse mesh discretizations for the cantilever examples

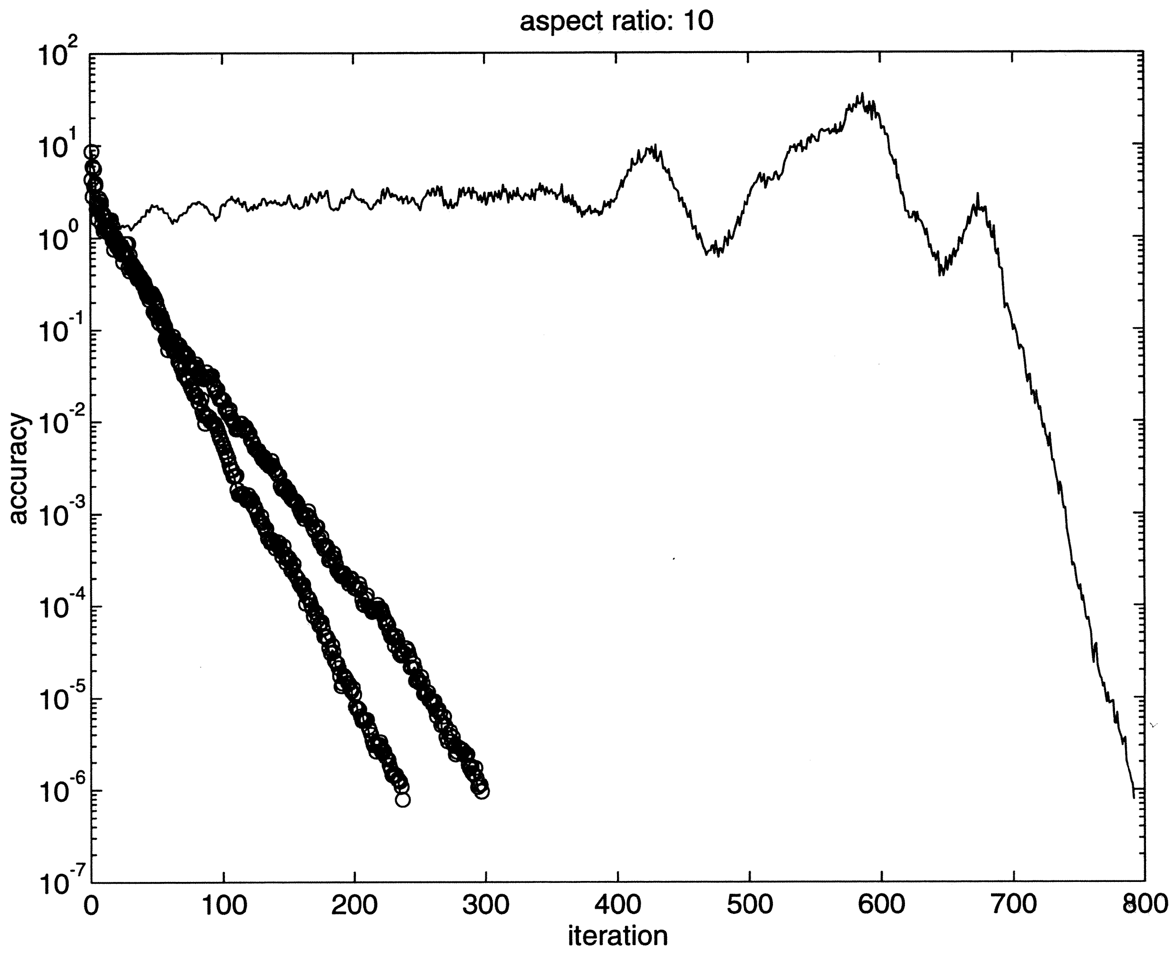
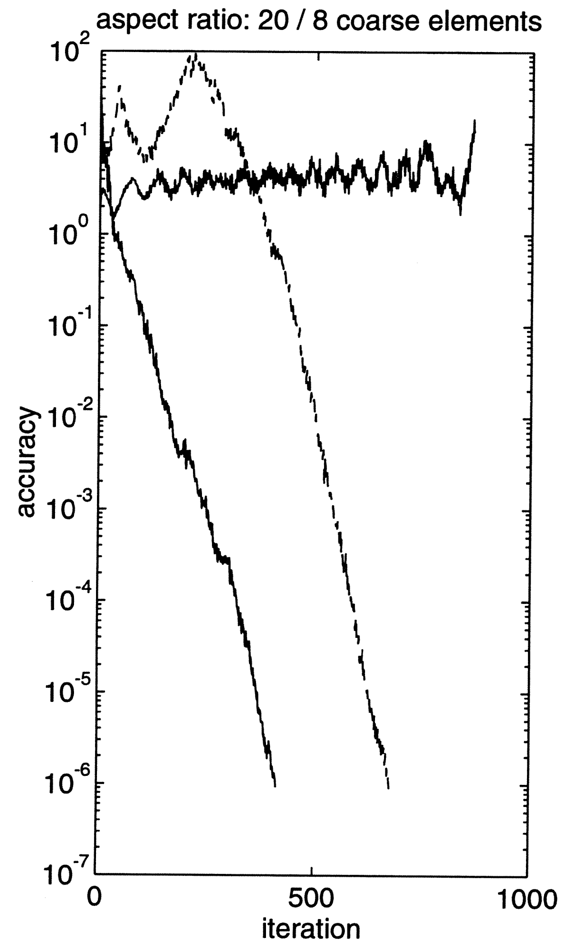


Figure 0.4: Convergence history. From top to bottom: diagonal, 8-coarse elements and 16-coarse elements

latter with both exact (solid lines) and approximate (circle marks) inverses of K_{cc}. Diagonal conjugate gradients converge marginally in this case at about N iterations. On the other hand, the coarse/fine mesh preconditioners converge rapidly for both exact and approximate inverses of the coarse stiffness. In fact, as expected from the condition numbers in Table 0.1, the approximate inverse preconditioners are identical to those with the exact inverses. It should be mentioned that the conjugate gradient on the coarse stiffness block that generated the approximate inverses required n iterations (the maximum theoretical limit) and the gave a final tolerance of slightly below 0.1. In the second test case (with aspect ratio equal to 20) diagonal preconditioning does not converge as can be seen in Figure 0.5. It can also be seen, that both approximate inverse preconditioners converge. The convergence of the one derived from the 16 element, is almost identical to that with the exact inverse of K_{cc}. On the other hand, with 8 element modelling the coarse mesh, the approximate inverse preconditioner converges slower compared with the case of the exact inverse. However, the approximation merely introduces a shift in the convergence, and after some point both history lines remain parallel. It should be mentioned that in this example both approximate inverses were generated from



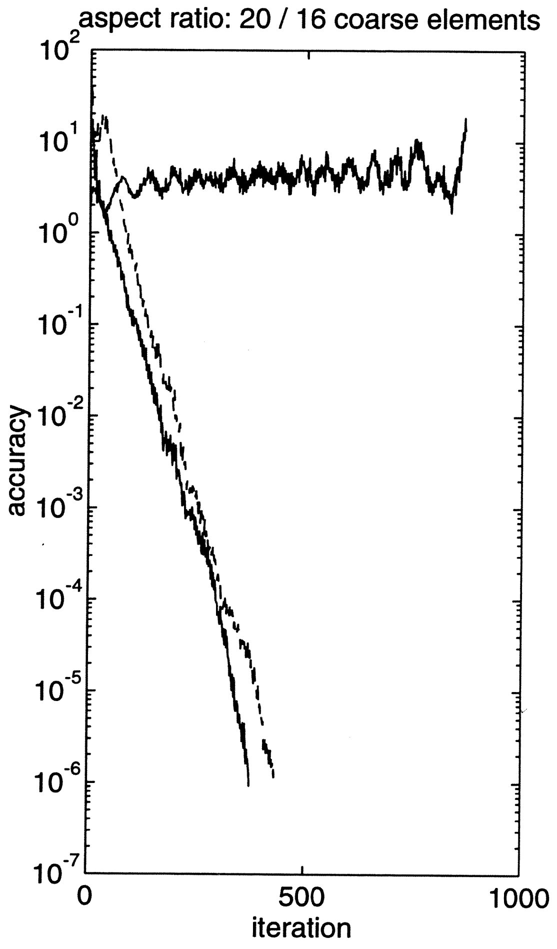


Figure 0.5: Convergence history

a non-converging diagonal conjugate gradient procedure on the coarse mesh.

7. Conclusions

The few initial results presented in the previous section, although related to limited size problems, are particularly encouraging. It seems to be feasible to obtain reasonable approximations to the inverse coarse stiffness that do not seriously affect the performance of the coarse/fine mesh preconditioner (compared with the case where the exact inverses are employed).

However, there is still a lot of work to be done in order to establish the properties and the conditions under which these methods can be of practical use.

Acknowledgements

The work on Quasi-Newton updates for the approximate inverses was motivated by a discussion with Mike Crisfield. Reference [22] was provided by Henk van der Vorst. Finally, the financial support of the ERCIM Fellowship Programme is gratefully acknowledged.

References

1. M. A. Ajiz and A. Jennings. A robust incomplete Choleski conjugate gradient algorithm. *IntNumerEng*, 20:949–966, 1984.

2. O. Axelsson and V. A. Barker. *Finite Element Solution of Boundary Value Problems: Theory and Computation.* Academic Press, 1984.

3. I. Babuška, A. Craig, J. Mandel, and J. Pitkäranta. Efficient preconditioning for the p-version finite element method in two dimensions. *SIAM J. Numer. Anal.*, 28:624–661, 1991.

4. R. Barret, M. Berry, T. F. Chan, J.Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM Publications, 1993.

5. W. T. Carter Jr., T. L. Sham, and K. H. Law. A parallel finite element method and its prototype implementation on a hypercube. *Comput. & Structures*, 31(6):921–934, 1989.

6. M. A. Crisfield. A faster modified Newton-Raphson iteration. *Comp. Meth. Appl. Mech. Eng.*, 20:267–278, 1979.

7. M. A. Crisfield. Iterative solution procedures for linear and nonlinear structural analysis. Technical Report LR900, Transport and Road Research Laboratory, 1979.

8. M. A. Crisfield. *Finite Elements and Solution Procedures for Structural Analysis.* Pineridge Press, 1986.

9. S. Demko, W. F. Moss, and P. W. Smith. Decay rates for inverses of band matrices. *Math. Comp.*, 43(168):491–499, 1984.

10. J. E. Dennis and J. J. Moré. Quasi-Newton methods, motivation and theory. *SIAM Rev.*, 19(1):46–89, 1977.

11. L. C. W. Dixon. Quasi-Newton algorithms generate identical points. *Math. Programming*, 2:383–387, 1972.

12. J. J. Dongarra, I. S. Duff, D. Sorensen, and H. A. van der Vorst. *Solving Linear Systems on Vector and Shared Memory Computers.* SIAM Publications, 1991.

13. M. C. Dracopoulos. *Finite Element Solution Procedures for Multi-processor Computer Architectures.* PhD thesis, University of London, 1993.

14. M. C. Dracopoulos and M. A. Crisfield. Coarse/fine mesh preconditioners for the iterative solution of finite element problems. Submitted to *Int. J. Num. Meth. Eng.*

15. M. C. Dracopoulos and M. A. Crisfield. A partially sequential preconditioner for a parallel and efficient finite element solution. Submitted to *Computing Systems in Engineering.*

16. C. Farhat and M. Lesoinne. Automatic partitioning of unstructured meshes for the parallel solution of problems in computational mechanics. *Internat. J. Numer. Methods Engrg.*, 36:745–764, 1993.

17. R. Fletcher. *Practical Methods of Optimization.* J. Wiley, 2nd edition, 1987.

18. M. B. Gijzen. *Iterative Solution Methods for Linear Equations in Finite Element Computations.* PhD thesis, Technische Universiteit Delft, 1994.

19. G.H. Golub and C.F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, MD, second edition, 1989.

20. J. L. Gustafson, G. R. Montry, and R. E. Benner. Development of parallel methods for

a 1024-processor hypercube. *SIAM J. Sci. Statist. Comput.*, 9(4):609–638, 1988.

21. L.A. Hageman and D.M. Young. *Applied Iterative Methods.* Academic Press, New York, 1981.
22. T. Huckle and M. Grote. A new approach to parallel preconditioning with sparse approximate inverses. Technical Report SCCM-94-03, Stanford University, 1994.
23. T. J. R. Hughes, I. Levit, and J. Winget. An element-by-element solution algorithm for problems of structural and solid mechanics. *Comp. Meth. Appl. Mech. Eng.*, 36:241–254, 1983.
24. A. Jennings. Influence of the eigenvalue spectrum on the convergence rate of the conjugate gradient method. *J. Inst. Math. Appl.*, 20:61–72, 1977.
25. A. Jennings and J. J. McKeown. *Matrix Computation.* John Wiley and Sons, 2nd edition, 1992.
26. O.G. Johnson, C.A. Micchelli, and G. Paul. Polynomial preconditioners for conjugate gradient calculations. *SIAM J. Numer. Anal.*, 20:362–376, 1983.
27. L. Yu Kolotilina and A. Yu Yeremin. Factorized sparse approximate inverse preconditioning. *SIAM J. Matrix Anal. Appl.*, 14(1):45–58, 1993.
28. D. G. Luenberger. *Introduction to Linear and Nonlinear Programming.* Addison-Wesley, 2nd edition, 1989.
29. S. F. McCormick, editor. *Multigrid Methods.* SIAM, 1987.
30. A. Meyer. A parallel preconditioned conjugate gradient method using domain decomposition and inexact solvers on each subdomain. *Computing*, 45:217–234, 1990.
31. L. Nazareth. A relationship between the BFGS and conjugate gradient algorithm and its implications for new algorithms. *SIAM J. Numer. Anal.*, 16(5):794–800, 1979.
32. J. M. Ortega. *Introduction to Parallel and Vector Solution of Linear Systems.* Plenum Press, New York, 1988.
33. J.M. Ortega. *Matrix Theory: A Second Course.* Plenum Press, New York, 1988.
34. M. Papadrakakis and M. C. Dracopoulos. A global preconditioner for the element-by-element solution methods. *Comp. Meth. Appl. Mech. Eng.*, 88:275–286, 1991.
35. M. Papadrakakis and M. C. Dracopoulos. Improving the efficiency of incomplete Choleski preconditionings. *Comm. Appl. Num. Methods*, 7:603–612, 1991.
36. A. Samuelsson, N.-E. Wiberg, and L. Bernspång. A study of the efficiency of iterative methods for linear problems in structural mechanics. *Internat. J. Numer. Methods Engrg.*, 22:209–218, 1986.
37. N.-E. Wiberg and P. Möller. Formulation and solution of hierarchical finite element equations. *Internat. J. Numer. Methods Engrg.*, 26:1213–1233, 1988.