Efficient parallel solvers for the biharmonic equation

Milan D. Mihajlović[1] and David J. Silvester[2]

Abstract. We examine the convergence characteristics and performance of parallelised Krylov subspace solvers applied to the linear algebraic systems that arise from low-order mixed finite element approximation of the biharmonic problem. Our strategy results in preconditioned systems that have nearly optimal eigenvalue distribution, which consists of a tightly clustered set together with a small number of outliers. We implement the preconditioner operator in a "black-box" fashion using publicly available parallelised sparse direct solvers and multigrid solvers for the discrete Dirichlet Laplacian. We present convergence and timing results that demonstrate efficiency and scalability of our strategy when implemented on contemporary computer architectures.

Key words. biharmonic equation, finite elements, preconditioning, Krylov solvers, sparse direct linear solvers, multigrid.

1 Introduction

In this paper we examine convergence and performance characteristics of a new iterative algorithm designed for the solution of linear algebraic systems that arise from the mixed finite element approximation of the classical biharmonic problem

$$\begin{cases} \Delta^2 u = f & \text{on } \Omega, \\ u = \frac{\partial u}{\partial \hat{n}} = 0 & \text{on } \partial\Omega, \end{cases} \tag{1}$$

where $\Omega \subset \mathcal{R}^2$ is a convex polygonal domain and $\partial\Omega$ is the Lipschitz continuous boundary with the outward unit normal vector $\hat{n}$. We also suppose

[1]Department of Computer Science, The University of Manchester, Oxford Road, Manchester M13 9PL, UK. e-mail: `milan@cs.man.ac.uk`

[2]Department of Mathematics, University of Manchester Institute of Science and Technology, Manchester M60 1QD, UK. e-mail: `na.silvester@na-net.ornl.gov`

that the load function f is sufficiently smooth (i.e. $f \in H^{-1}(\Omega)$). The mixed approximation of the problem (1) reads:

Find a pair (v, u) such that

$$\begin{cases} -\Delta u = v \\ -\Delta v = f \end{cases} \quad \text{in } \Omega \qquad u = \frac{\partial u}{\partial \hat{n}} = 0 \quad \text{on } \partial\Omega. \tag{2}$$

This problem arises in fluid mechanics, where u and v represents streamline function and vorticity, respectively. It is also a model for plate bending, where u is the deflection and v is the bending moment.

Let us mention one additional property of the biharmonic operator that is not characteristic for the Laplacian (but exists for all elliptic higher-order operators, of which the biharmonic is the simplest example), namely, the existence of the infinite number of sign variations in the principal eigenmode in the vicinity of acute corners of the domain Ω (see [13],[15]). A typical manifestation of this property in fluid mechanics is the existence of an infinite sequence of counter-rotating eddies of exponentially decreasing magnitude and diameter, or an infinite number of bending lines of a clamped plate near the corner. This phenomenon was explained theoretically in [13]. However, due to the extreme sensitivity and exceptionally small magnitudes of the eigenfunction where the phenomenon occurs, various numerical computations were unable to reproduce this effect accurately until recently [4],[7]. In either case special direct methods were developed (see [3],[6]) in order to achieve the desired accuracy.

Unlike this previous work, where a very accurate approximation to the solution of the biharmonic problem is sought in a small proportion of the domain, we address here the problem of finding an approximation of the global solution of the problem (1) when only moderate accuracy is required. We consider finite element discretisation of the problem (2) and develop a new preconditioning technique that enables a rapid convergence of the Krylov subspace solvers when employed in this context.

The paper is organised as follows. In Section 2 we outline the finite element discretisation of the problem (2). In Section 3 we present the new indefinite constraint preconditioner. More details on theoretical aspects related with the preconditioning methodology and its relation with similar approaches that may be used in this context can be found in [21]. In Section 4 we present numerical experiments that demonstrate rapid convergence of both exact and inexact indefinite constraint preconditioners that we propose in this context. We demonstrate the computational efficiency and simplicity of our inexact approach by solving mixed approximations of an arbitrary degree $1 \leq p \leq 3$ using a simple multigrid approximation of the discrete Dirichlet Laplacian, with only minor deterioration in the efficiency of the overall solver. In Section 5 we examine the performance and scalability of two particular "black-box" implementations of the Krylov solver with the exact and inexact constraint preconditioner using the efficient publicly available codes from the NAG [17] and HSL [11] libraries, as well as the public codes from [17] and [14] for multigrid. For completeness, we compare our performance results with those of a general sparse direct solver from the HSL library [11]. Finally, Section 6 contains a discussion of the obtained results and gives conclusions.

2 Problem formulation

In this section we outline the essential features of the theoretical model, (for further details, see [9] and [21]). The weak formulation of the problem (2) reads as follows:

Find $(v, u) \in H^1(\Omega) \times H_0^1(\Omega)$ such that

$$\begin{array}{rcll} (v, w_1) - (\nabla u, \nabla w_1) & = & 0 & \forall w_1 \in H^1(\Omega) \\ -(\nabla v, \nabla w_2) & = & -(f, w_2) & \forall w_2 \in H_0^1(\Omega). \end{array} \tag{3}$$

Here $(\cdot, \cdot)$ denotes the vector or scalar inner product in $L^2(\Omega)$, and the space $H^1(\Omega)$ is the usual Sobolev space of functions that have square-integrable first derivatives. $H_0^1(\Omega)$ is a restriction of $H^1(\Omega)$ to functions that satisfy the homogeneous Dirichlet boundary condition on $\partial\Omega$. In order to get an approximate solution of the problem (3), we take the finite dimensional subspaces $S^h \subset H^1(\Omega)$ and $S_0^h = S^h \cap H_0^1(\Omega)$, where h is a mesh parameter associated with a subdivision $\mathcal{T}_h$ of the domain Ω into non-overlapping triangles. This leads to a discrete weak formulation of the problem (3):

Find $(v_h, u_h) \in S^h \times S_0^h$ satisfying

$$\begin{array}{rcll} (v_h, w_{1,h}) - (\nabla u_h, \nabla w_{1,h}) & = & 0 & \forall w_{1,h} \in S^h \\ -(\nabla v_h, \nabla w_{2,h}) & = & -(f, w_{2,h}) & \forall w_{2,h} \in S_0^h. \end{array} \tag{4}$$

Since the discrete constraint space

$$Z_h = \left\{ v_h \in S^h \,|\, (\nabla v_h, \nabla w_{2,h}) = 0 \quad \forall w_{2,h} \in S_0^h \right\} \tag{5}$$

is not a subspace of its continuous counterpart, the usual argument showing that well-posedness of (4) follows from that of (3) does not apply. This means that appropriate mesh dependent norms on $S^h \times S^h_0$ (that are introduced and analysed in [1]) are needed in order to show that the natural approximation spaces S^h and S^h_0 (consisting of piecewise polynomials of degree $p \geq 1$) satisfy Brezzi's stability conditions [5]. This inherent lack of stability in the mixed formulation is what makes it so difficult to construct spectrally equivalent preconditioners and robust multigrid methods (especially of the "black-box" type) for the fast solution of the discrete problem (4).

In order to rewrite the discrete weak form (4) as a linear algebraic problem, we introduce the finite element basis sets

$$S^h = \text{span}\{\phi_i\}_{i=1}^{n_I+n_B}, \qquad S^h_0 = \text{span}\{\phi_j\}_{j=1}^{n_I} \tag{6}$$

where n_I and n_B stand for the number of degrees of freedom in the interior of Ω, and on the boundary $\partial\Omega$, respectively. Then, the functions v_h, u_h and f_h can be uniquely represented by

$$v_h = \sum_{i=1}^{n_I+n_B} v_i\phi_i, \qquad u_h = \sum_{i=1}^{n_I} u_i\phi_i, \qquad f_h = \sum_{i=1}^{n_I} f_i\phi_i. \tag{7}$$

with $\bar{v} = [v_1 \cdots v_{n_I+n_B}]^T$, $\bar{u} = [u_1 \cdots u_{n_I}]^T$, and $\bar{f} = [f_1 \cdots f_{n_I}]^T$. By defining the mass matrix $M \in \mathcal{R}^{(n_I+n_B)\times(n_I+n_B)}$ with the elements $M_{ij} = (\phi_i, \phi_j)$, the constraint matrix $K \in \mathcal{R}^{n_I\times(n_I+n_B)}$ with the elements $K_{ji} = -(\nabla\phi_j, \nabla\phi_i)$, and the right hand side vector $\bar{f} \in \mathcal{R}^{n_I}$ with the elements $f_i = -(f, \phi_i)$, we obtain the following linear system

$$\begin{bmatrix} M & K^T \\ K & 0 \end{bmatrix} \begin{bmatrix} \bar{v} \\ \bar{u} \end{bmatrix} = \begin{bmatrix} \bar{0} \\ \bar{f} \end{bmatrix}. \tag{8}$$

Our main goal is to explore the use of efficient "black-box" sparse linear solvers and multigrid approximations designed for the Dirichlet Laplace problem as a preconditioner for (8). In order to design such a preconditioner, we decompose the function v_h into the sum of interior and boundary contributions

$$v_h = \sum_{i=1}^{n_I+n_B} v_i\phi_i = \sum_{j=1}^{n_I} v_j\phi_j + \sum_{k=1}^{n_B} v_{n_I+k}\phi_{n_I+k}. \tag{9}$$

Such a decomposition induces an additional partitioning of the constraint matrix in (8) as $K = (K_I, K_B)$, where K_I is a discrete Dirichlet Laplacian matrix, and the mass matrix M is partitioned in 2×2 block form corresponding to the contributions from the interior and the boundary nodes. The right-hand side vector is partitioned accordingly. In this way we obtain a 3×3 block system

$$\begin{bmatrix} M_I & M_C^T & K_I \\ M_C & M_B & K_B^T \\ K_I & K_B & 0 \end{bmatrix} \begin{bmatrix} \bar{v}_I \\ \bar{v}_B \\ \bar{u} \end{bmatrix} = \begin{bmatrix} \bar{0} \\ \bar{0} \\ \bar{f} \end{bmatrix}. \tag{10}$$

In (10) we assume $K_I = K_I^T$, as the discrete Dirichlet Laplacian is a symmetric positive definite matrix. In the next section we present a new methodology for preconditioning the system (10) based on the constraint preconditioning approach that is developed in [12]. Inexact versions of this methodology have been suggested in the context of mixed finite element formulations of magnetostatics problems [19] and some optimal control problems [2], but have not been exploited in the context of the biharmonic problem until now.

3 The indefinite constraint preconditioner

As mentioned earlier, constraint preconditioning has been used in various contexts [2], [10], [12], [19]. In the case of the biharmonic problem (10) a very efficient implementation can be obtained by exploiting the special block structure of the coefficient matrix. Specifically, we consider the following constraint preconditioner

$$P = \begin{bmatrix} 0 & 0 & K_I \\ 0 & M_B & K_B^T \\ K_I & K_B & 0 \end{bmatrix}. \tag{11}$$

Note that the matrix P in (11) is symmetric, but indefinite. As a consequence, we only consider non-symmetric Krylov subspace methods here. Motivated by the issues of robustness and wide applicability, we thus turn to the NAG library implementation of BiCGSTAB(2) [17]. Note also that the action of P^{-1} in (11) can be achieved by a simple backsubstitution at a block level. However, in the case of an "exact" version of the preconditioner P, the block K_I needs to be factorised before calling the Krylov solver (note that the block M_B, as the mass matrix block, can be readily approximated by its diagonal, see [24]).

We can further simplify the action of P^{-1} by approximating the inverses of K_I in (11). Namely, the block K_I represents a discrete Dirichlet Laplacian and can readily be approximated by a small fixed number of multigrid V cycles with a simple smoother (such as damped Jacobi). If we denote by K_* the action of MG on K_I, we obtain the following inexact version of the indefinite constraint preconditioner

$$P_* = \begin{bmatrix} 0 & 0 & K_* \\ 0 & M_B & K_B^T \\ K_* & K_B & 0 \end{bmatrix}. \tag{12}$$

Bearing in mind that multigrid is a spectrally equivalent preconditioner for the discrete Dirichlet Laplacian problem, we expect only minor deterioration in convergence characteristics of the algorithm in comparison with the exact version. Thus a Krylov solver with an inexact indefinite constraint preconditioner should have potentially better asymptotic behaviour (both in terms of the execution time and storage requirements) than the exact implementation. Another feature in the inexact case is that there is no initial cost associated with assembling the preconditioner (a sparse Cholesky factorisation is typically computed in the exact case). Examples in Section 5 clearly demonstrate this point. In the case of complex domains discretised by non-structured grids, multigrid can be replaced by algebraic multigrid (AMG) [23] applied in a "black-box" fashion as an approximation to the blocks K_I.

Detailed analysis of the preconditioned biharmonic operator

$$\begin{bmatrix} M_I & M_C^T & K_I \\ M_C & M_B & K_B^T \\ K_I & K_B & 0 \end{bmatrix} \begin{bmatrix} \bar{v}_1 \\ \bar{v}_2 \\ \bar{v}_3 \end{bmatrix} = \lambda \begin{bmatrix} 0 & 0 & K_I \\ 0 & M_B & K_B^T \\ K_I & K_B & 0 \end{bmatrix} \begin{bmatrix} \bar{v}_1 \\ \bar{v}_2 \\ \bar{v}_3 \end{bmatrix} \tag{13}$$

is given in [21] and we outline only the major points here. The matrix P in (11) is non-singular and the preconditioned system (13) has an eigenvalue $\lambda = 1$ with multiplicity $\geq 2n_I$. The preconditioned biharmonic operator (13) satisfies the conditions of Theorem 2.3 from [12]. Therefore, we can conclude that for the multiple eigenvalue $\lambda = 1$ we have n_I eigenvectors of the form $[\bar{0}\ \bar{0}\ \bar{v}_3]^T$, and the remaining eigenvectors satisfy the condition 2) of Theorem 2.3 from [12]. The remainder of the eigenvalues ($\leq n_B$) are contained in an interval $[c, Ch^{-1}]$, where c and C are positive constants as described in [18]. Although the upper bound on the spectrum is inversely proportional to h, only a small number of the eigenvalues differ from unity, and the proportion of those eigenvalues in total number decreases as the dimension of the linear system (10) increases (bearing in mind that $n_B = O(h^{-1})$ and $n_I = O(h^{-2})$). In this situation we can anticipate rapid convergence of a Krylov subspace solver when applied to the preconditioned system. In fact, Theorem 3.5 from [12] asserts that the dimension of the Krylov space in our case is at most $n_B + 2$. This means that the Krylov solver should converge in at most $n_B + 2$ iterations, providing exact arithmetic is used. In [10] the authors proposed a CG algorithm for the reduced problem (which is equivalent to the original indefinite problem). In this case we need to work in the null space of the constraint operator K from (8). This approach is possible to apply in our case as well. However, our idea was to construct a "black-box" preconditioner that works efficiently for the original formulation of the problem (8). In this case both the coefficient matrix and the preconditioner are indefinite and we need to employ a Krylov solver suitable for non-symmetric systems.

4 Convergence results

In this section we examine the convergence characteristics of the Krylov subspace solver with exact (11) and inexact (12) versions of the indefinite constraint preconditioner. We report our results in terms of the iteration counts of the preconditioned Krylov subspace solver for the linear systems obtained from various degrees of approximation $1 \leq p \leq 3$. Implementation details and performance results are deferred until the next section.

We adopt BiCGSTAB(2) (see [22]) as a representative Krylov subspace method. BiCGSTAB(2) is a suitable choice when both the coefficient matrix and the preconditioner are indefinite. The method exhibits a robust and reliable performance in our case: it has never failed to converge. In our implementation, complying with the aim of developing a fully "black-box" solver, we take the BiCGSTAB(ℓ) algorithm that is implemented as a part of the NAG Mark 20 library suite F11BDF, F11BEF, and F11BFF (see [17]). An exact version of the indefinite constraint preconditioner is implemented using the code HSL_MP62 from the HSL library [11]. This code is a sparse direct linear solver based on a multiple front method [20]. The code exploits a domain decomposition strategy and is targeted towards solving symmetric positive definite systems that arise from finite element discretisations. The code employs data input at element matrix level and MPI [16] for message passing. Parallel aspects of this package will be discussed in more detail in Section 5. An inexact version of the indefinite constraint preconditioner is implemented using the multigrid solver provided within the NAG library Mark 20 [17] as subroutine D03EDF. This routine essentially implements Wesseling's `mgd1` algorithm [25] that solves a general elliptic PDE on a rectangular domain that can be discretized as a seven point difference operator. The smoother is based on an incomplete Crout decomposition and linear interpolation and analogous restriction are used. We also experimented with a parallel multigrid solver `mgd2` originally written by Bunner [14]. This uses a domain decomposition technique and employs staggered grids with cell-centered operations. It uses full weighting for the restriction and bilinear interpolation for the correction. The code is designed for fluid mechanics problems, and some adjustments are needed in our case (namely, transformation from vertex-centered to cell-centered case). Notice that in the context of using multigrid as a preconditioner, higher-order discretization (quadratic and cubic) operators can be approximated using the linear finite difference multigrid operator and appropriately adjusting the grid dimensions. Indeed, restrictions in the library multigrid code that we used (D03EDF) are the reason why we implemented the inexact version of the preconditioner in this fashion. This will, however, have an impact to the convergence characteristics (see Table 2, parts b) and c) and the related discusion).

Example 1. We test our preconditioning strategy on the linear systems obtained by the discretisation of (2) on the unit square domain $\Omega = [0, 1]^2$. The discretisation is performed on uniform meshes (congruent right-angled triangles) using standard (linear, quadratic or cubic) Lagrangian elements. The right-hand side vector is chosen randomly (although the elements of $\bar{f}$ in (8) are scaled like h^2 thus simulating a randomly distributed load). We performed the experiments for several different right-hand sides and we report the average iteration counts (only minor variations in the iteration counts were observed as the right-hand side vector was changed).

The iteration counts for the exact version of the indefinite constraint preconditioner are summarised in Table 1. In the table, n denotes the total dimension of the discrete biharmonic system (10) (that is $n = 2n_I + n_B$), and N_{it} denotes the number of BiCGSTAB(2) iterations needed to achieve the prescribed tolerance. The stopping criterion used by the BiCGSTAB(2) routine is

$$\|\bar{r}_i\|_\infty \leq \varepsilon \left(\|\bar{f}\|_\infty + \|A\|_\infty \|\bar{x}_i\|_\infty \right), \tag{14}$$

where A is the matrix from (10) in our case, and $\bar{x}_i$ is the current iterate (see

[17]). We consider two values of the tolerance ε, namely 10^{-7} and 10^{-9}. The first choice gives accuracy $\|\bar{r}_i\|_\infty \sim 10^{-4}$, while for the tighter tolerance we have $\|\bar{r}_i\|_\infty \sim 10^{-6}$. It can be seen in Table 1 that for the weaker tolerance, the number of iteration steps is independent of the mesh parameter h and the order of approximation p. For the tighter tolerance the number of iteration steps is loosely dependent of h (as might be anticipated from the eigenvalue distribution established in [21]), but are independent of the order of the approximation p.

Table 1: Convergence of the BiCGSTAB(2) algorithm preconditioned by the exact indefinite constraint preconditioner (sparse Cholesky factorisation of the discrete Dirichlet Laplacian by HSL_MP62) when applied to the discrete biharmonic system (10).

a) piecewise linear elements

mesh	96×96	144×144	192×192	288×288	384×384	576×576	768×768
n	18434	41474	73730	165890	294914	663554	1179650
N_{it} (10^{-7})	7	7	7	7	7	7	7
N_{it} (10^{-9})	19	21	23	23	27	27	27

b) piecewise quadratic elements

mesh	48×48	72×72	96×96	144×144	192×192	288×288	384×384
n	18434	41474	73730	165890	294914	663554	1179650
N_{it} (10^{-7})	9	9	9	9	9	9	9
N_{it} (10^{-9})	17	17	21	23	25	25	25

c) piecewise cubic elements

mesh	32×32	48×48	64×64	96×96	128×128	192×192	256×256
n	18434	41474	73730	165890	294914	663554	1179650
N_{it} (10^{-7})	9	9	9	9	9	9	9
N_{it} (10^{-9})	19	21	21	23	23	25	25

Next we turn our attention to the convergence of the BiCGSTAB(2) algorithm preconditioned by the inexact version of the indefinite constraint preconditioner. The experimental settings is otherwise the same as in the case of an exact version. In the case of higher-order approximation we apply the same multigrid solver as in the case of piecewise linear approximation: specifically we perform a small number of V(1,1) multigrid cycles (one pre-smoothing step before residual restriction and one post-smoothing step after coarse grid correction). Convergence characteristics are examined as a function of the number of V cycles. The convergence results are summarised in Table 2. We test three different choices of the number of V cycles: 1, 3 and 5. In the case of piecewise linear approximation this choice seems to have a crucial effect on the convergence characteristics, both in terms of the iteration counts and the total execution time (presented in Table 11). Taking $V_n = 5$ gives essentially the same results as the exact version.

It is obvious that as the dimension of the linear system (10) increases, the convergence rate of the solver using an inexact preconditioner with a small number of V cycles starts to deteriorate. It is sensible in this situation (in the case of linear approximation) to increase gradually the number of V cycles. In the case of higher-order approximation the number of V cycles does not have an impact on the convergence characteristics. Therefore $V_n = 1$ is a suitable choice for these cases, regardless of the system dimension (as it has the shortest execution time, see Table 11). Note, however, that this is the consequence of the particular implementation that we used (linear multigrid operator is used in all cases). If the appropriate higher-order multigrid operator was employed (or the matrix K_I was submitted to AMG [23]), the advantage of applying larger number of V cycles would become obvious.

Example 2. In the introduction we highlighted the fact that a specific feature of the biharmonic operator gives rise to rapidly oscillating eigenfunctions. In order to compute accurately this part of the solution, one needs to employ highly non-uniform and non-structured grids, with the ratio between the largest and the smallest triangle in the mesh being several orders of magnitude. This will make the linear algebraic system (8) much more ill-conditioned than in the case of uniform grid discretisation studied in Example 1. Besides, the asymptotic ratios for the number of interior and boundary nodes (that are $n_I = O(h^{-2})$, $n_B = O(h^{-1})$ in the case of uniform grid discretisation) do not apply here. This is important, as the theoretical upper bound for the number of iterations in the proposed method is $n_B + 2$ (cf. [12, Theorem 3.5]). Moreover, in order to compute the solution accurately, a much tighter tolerance than adopted in Example 1 is needed here (see [6]). All previously described features makes the problem described here a great challenge for the iterative solver. Again, we adopt the unit square domain, but now we discretise it by the non-uniform grid with smooth refinements in geometric fashion towards the corners. The grid is generated using Delaunay-Voronoi mesh generator from NAG library [17] (routines D06BAF, D06ABF, D06CAF). We study only the case of discretisation using linear el-

Table 2: Convergence of the BiCGSTAB(2) algorithm preconditioned by the inexact indefinite constraint preconditioner (routine D03EDF is used as a linear multigrid solver) when applied to the solution of the discrete biharmonic system (10). The results are given as a function of the system dimension n and the number of the multigrid cycles V_n.

a) piecewise linear elements

mesh	114×114	258×258	450×450	642×642	1026×1026	1794×1794
n	25994	133130	405002	824330	2105354	6436874
$V_n = 1\,(10^{-7}/10^{-9})$	16/34	28/44	36/60	42/78	56/85	80/138
$V_n = 3\,(10^{-7}/10^{-9})$	10/18	10/20	10/26	14/32	20/56	20/ 66
$V_n = 5\,(10^{-7}/10^{-9})$	10/18	12/20	12/24	10/27	10/29	10/ 33

b) piecewise quadratic elements

mesh	57×57	129×129	225×225	321×321	513×513	897×897
n	25994	133130	405002	824330	2105354	6436874
$V_n = 1\,(10^{-7}/10^{-9})$	28/44	36/52	54/68	62/84	73/99	112/154
$V_n = 3\,(10^{-7}/10^{-9})$	28/38	36/50	50/68	57/83	71/95	108/148
$V_n = 5\,(10^{-7}/10^{-9})$	28/38	34/50	50/68	55/81	72/91	108/148

c) piecewise cubic elements

mesh	38×38	86×86	150×150	214×214	342×342	598×598
n	25994	133130	405002	824330	2105354	6436874
$V_n = 1\,(10^{-7}/10^{-9})$	40/46	48/62	64/77	79/98	88/112	123/171
$V_n = 3\,(10^{-7}/10^{-9})$	34/42	48/62	64/75	77/97	89/109	123/165
$V_n = 5\,(10^{-7}/10^{-9})$	34/42	47/62	63/75	77/97	87/109	123/165

ements, and solve the linear system (8) by the preconditioned BiCGSTAB(2) algorithm with both exact and inexact version of the preconditioner. The only difference from Example 1 is that in the case of the inexact preconditioner, instead of geometric multigrid, we employ AMG [23]. The results are summarised in Table 3 and Table 4.

In Table 3 and Table 4 the total dimension of the linear system (10) is denoted by n ($n = 2n_I + n_B$), n_I and n_B being the numbers of interior and boundary nodes, respectively. $h_{\min}$ denotes the smallest triangle size within the mesh. We report the iteration counts for 3 different levels of tolerance: besides 2 standard levels introduced in Example 1, we introduce tighter tolerance $\varepsilon = 10^{-12}$, which gives approximately 9 correct digits in the solution. In the case of inexact preconditioner we employ the AMG code AMG1R5 [14] and perform a fixed number of V(1,1) AMG cycles with

Table 3: Convergence of the BiCGSTAB(2) algorithm preconditioned by the exact indefinite constraint preconditioner (sparse Cholesky factorisation of the discrete Dirichlet Laplacian by HSL_MP62) when applied to the discrete biharmonic system (10). Non-uniform grid refined near the corners, piecewise linear elements.

n	2878	9254	18248	28720
n_I	1343	4431	8728	13764
n_B	192	392	792	1192
$h_{\min}$	0.011235	0.004545	0.000378	0.000028
N_{it} (10^{-7})	9	11	13	13
N_{it} (10^{-9})	19	23	25	31
N_{it} (10^{-12})	27	35	43	47

Table 4: Convergence of the BiCGSTAB(2) algorithm preconditioned by the inexact indefinite constraint preconditioner (routine AMG1R5 [14] is used as a multigrid solver) when applied to the solution of the discrete biharmonic system (10). Non-uniform grid refined near the corners, piecewise linear elements. The results are given as a function of the system dimension n and the number of the multigrid cycles V_n.

n	2878	9254	18248	28720
n_I	1343	4431	8728	13764
n_B	192	392	792	1192
$h_{\min}$	0.011235	0.004545	0.000378	0.000028
$V_n = 1\,(10^{-7}/10^{-9}/10^{-12})$	20/26/36	24/34/50	36/52/66	40/55/75
$V_n = 3\,(10^{-7}/10^{-9}/10^{-12})$	10/18/30	12/22/36	14/28/48	18/32/56
$V_n = 5\,(10^{-7}/10^{-9}/10^{-12})$	10/18/28	12/22/34	14/28/44	18/32/51

Gauss-Seidel smoother.

From Table 3 and Table 4 we can conclude that the behaviour of the preconditioned iterative solver is within theoretical predictions and is only slightly worse than in the case of the uniform mesh. The iteration counts in both exact and inexact case grow moderately with the mesh refinement, even for the case of the tightest tolerance that we studied. The iteration counts growth is in line with theoretical predictions. In the inexact case, an increase in the number of AMG V cycles has the same beneficiary effect as in Example 1, with 3-5 cycles typically being as good as the direct sparse factorisation of K_I.

5 Performance results

In this section we consider the computational efficiency of our "black-box" iterative solvers for the discrete biharmonic problem. For comparison we also report the performance results for a general sparse solver HSL_MP42 from the HSL library [11] when applied in a "black-box" fashion to the same problem. We restrict the presentation in this section to the problem presented in Example 1 of Section 4.

First we describe our experimental environment. All our tests are performed on Origin 2000 architecture with 16 R10000 processors (195MHz – corresponding to a theoretical peak performance of 390MFlops per processor), 6GB of RAM and OS 6.5.12f. We used a CPUset facility for a dedicated processor and memory access with 8 processors and 3GB of RAM. This facility provides more consistent execution times of a code, as the Origin is an architecture with non-uniform memory access (so that execution times of a given code may vary significantly). All our codes are written in Fortran 90 (the same applies to the HSL library codes, while NAG routines and the code `mgd2` are written in Fortran 77, and compiled by the f90 compiler version 7.3.1.2m using the maximal optimisation (`-O3 -OPT:Olimit=0`)). The BLAS routines needed in the HSL software are taken from the SCSL 1.3 library, and MPI is taken from the MPT 1.5 library.

As stated previously, our aim is to develop an efficient iterative biharmonic solver based on the preconditioning methodology in Section 3. In making a decision about which part of the code to parallelise, we considered the simple execution time model for an iterative solver that is introduced in [8]. In this model, the execution time of a single loop within a Krylov solver is divided into three parts (for simplicity, consider a Krylov solver with constant arithmetic cost per iteration); time needed to compute the action of a preconditioner T_p, time for performing a matrix-vector multiplication T_m, and time to perform vector updates T_o. In addition to these, there is a one-off cost T_c of assembling the preconditioner matrix (and/or to factorise it) before entering the Krylov solver loop. In the case of an exact indefinite constraint preconditioner the HSL_MP62 subroutine is used to solve systems $K_I\bar{s} = \bar{r}$. When executed on a single processor of the Origin 2000 architecture, the relative contributions (in %) of T_o, T_m and T_p to the total loop time of a single BiCGSTAB(2) iteration are given in Table 5. Here we present only the case of piecewise linear approximation, but the relative contributions are similar when higher-order elements are used.

Table 5: Relative contributions (in %) to the total BiCGSTAB(2) loop time from the preconditioning T_p, matrix-vector-multiplication T_m and the vector updates T_o as a function of the problem size n. Piecewise linear approximation, the unit square domain is divided into 4×4 subdomains, code runs on a single processor.

mesh	96×96	144×144	192×192	288×288	384×384	576×576	768×768
n	18434	41474	73730	165890	294914	663554	1179650
T_o	5.57	4.20	4.06	3.44	3.09	2.85	2.63
T_m	6.45	4.92	5.32	4.34	3.80	3.68	3.49
T_p	87.98	90.88	90.62	92.22	93.11	93.47	93.88

It is clear from Table 5 that the preconditioning time T_p contributes by far the largest proportion of the BiCGSTAB(2) loop time, and that this proportion increases as the dimension n grows (the vector updates and matrix-vector multiplication have an optimal arithmetic cost $O(n)$, whilst the backsubstitution phase of solving $K_I\bar{s} = \bar{r}$ requires $O(n^\alpha)$ flops with $\alpha > 1$). We also assess the relative contribution (in %) of the total execution time of time T_c associated with sparse factorisation of the block K_I. Table 6 presents these results when the code is executed on a single processor.

Table 6: Relative contributions (in %) of the factorisation time of the block K_I to the overall execution time as a function of the problem size and the domain partition ($m \times m$). The results are given for piecewise linear elements for both tolerances $10^{-7}/10^{-9}$.

mesh	96×96	144×144	192×192	288×288	384×384	576×576	768×768
n	18434	41474	73730	165890	294914	663554	1179650
4×4	30.0/19.1	39.6/18.9	44.3/20.6	53.1/27.0	59.5/28.9	65.7/35.6	66.4/36.7
8×8	17.4/ 9.5	22.9/ 9.9	24.7/10.2	29.8/13.1	34.5/13.8	36.8/16.2	40.3/18.1

Having established that $T_p \gg T_o + T_m$ in both exact and inexact cases, we adopt the strategy of parallelising only the preconditioner step. This is accomplished by calling the publicly available parallel solvers for discrete Dirichlet Laplacian problems; HSL_MP62 in the exact version, and `mgd2` in the inexact case. Matrix-vector products are performed sequentially using the NAG routine F11XEF. Timing results (measured in wall clock seconds) for the BiCGSTAB(2) algorithm with an exact indefinite constraint preconditioner are reported in Table 7 for $\varepsilon = 10^{-7}$ and Table 8 for $\varepsilon = 10^{-9}$. The results are given for various degrees of approximation $1 \leq p \leq 3$, as a function of the problem size n and the domain partition $m \times m$. N_p denotes the number of processors.

Table 7: Timing results for the BiCGSTAB(2) algorithm preconditioned by the exact indefinite constraint preconditioner (implemented using HSL_MP62 code) as a function of the problem size and domain decomposition. $\varepsilon = 10^{-7}$.

a) piecewise linear elements

mesh		96×96	144×144	192×192	288×288	384×384	576×576	768×768
part	N_p	18434	41474	73730	165890	294914	663554	1179650
4×4	1	4.40	11.78	27.47	96.76	248.08	1059.08	3181.00
	2	2.82	7.90	17.95	59.51	147.40	598.41	1753.71
	4	2.15	5.40	12.11	37.95	89.65	327.49	960.78
	8	1.91	4.34	9.04	27.04	61.11	206.60	561.01
8×8	1	5.58	12.01	24.90	72.19	161.45	575.09	1550.93
	2	3.70	8.14	16.38	44.67	96.22	327.88	856.09
	4	2.88	5.86	11.83	31.58	65.46	203.93	514.22
	8	2.52	4.79	9.51	24.97	50.45	147.61	350.41

b) piecewise quadratic elements

mesh		48×48	72×72	96×96	144×144	192×192	288×288	384×384
part	N_p	18434	41474	73730	165890	294914	663554	1179650
4×4	1	3.82	11.20	24.38	75.64	177.48	663.81	1926.20
	2	2.38	7.66	16.69	49.57	112.92	405.98	1131.25
	4	1.82	5.18	11.37	32.40	71.16	232.52	644.76
	8	1.63	4.04	8.58	23.98	51.80	153.26	375.13
8×8	1	5.40	13.19	25.78	69.09	146.93	524.47	1247.44
	2	3.42	9.07	17.68	46.31	95.41	328.20	747.84
	4	2.58	6.34	12.65	32.38	64.85	190.17	440.48
	8	2.22	5.12	10.00	25.59	50.76	137.46	290.78

c) piecewise cubic elements

mesh		32×32	48×48	64×64	96×96	128×128	192×192	256×256
part	N_p	18434	41474	73730	165890	294914	663554	1179650
4×4	1	3.66	10.38	21.63	62.86	144.56	511.11	1474.25
	2	2.45	7.55	15.43	43.06	95.44	330.28	890.84
	4	1.95	5.29	11.17	29.96	63.87	193.17	523.47
	8	1.79	4.31	8.71	23.11	47.97	137.05	341.67
8×8	1	5.06	13.00	24.43	62.51	128.03	382.31	987.52
	2	3.52	9.57	17.89	44.40	89.03	261.50	633.84
	4	2.88	6.81	13.22	31.88	63.14	167.59	425.02
	8	2.54	5.46	10.79	25.95	50.82	131.04	287.41

Table 8: Timing results for the BiCGSTAB(2) algorithm preconditioned by the exact indefinite constraint preconditioner (implemented using HSL_MP62 code) as a function of the problem size and domain decomposition. $\varepsilon = 10^{-9}$.

a) piecewise linear elements

mesh		48×48	72×72	96×96	144×144	192×192	288×288	384×384
part	N_p	18434	41474	73730	165890	294914	663554	1179650
4 × 4	1	6.92	24.70	59.09	190.30	510.01	1852.23	5745.58
	2	4.30	16.66	39.77	122.12	320.69	1118.50	3289.65
	4	3.27	11.09	26.70	78.93	198.79	645.04	1822.92
	8	2.92	8.83	19.77	57.27	139.80	423.94	1042.36
8 × 8	1	10.21	27.79	60.42	163.57	402.33	1301.75	3461.52
	2	6.65	18.93	40.94	107.33	254.32	808.53	2000.88
	4	5.06	13.15	28.76	74.00	170.27	506.89	1214.39
	8	4.33	10.50	22.47	57.76	132.18	371.50	833.93

b) piecewise quadratic elements

mesh		48×48	72×72	96×96	144×144	192×192	288×288	384×384
part	N_p	18434	41474	73730	165890	294914	663554	1179650
4 × 4	1	6.43	18.76	48.62	158.92	388.86	1375.98	3916.85
	2	3.94	12.85	33.48	105.61	252.70	862.49	2330.77
	4	3.01	8.61	22.85	68.79	159.02	500.88	1325.74
	8	2.70	6.65	16.95	50.93	116.01	330.66	757.37
8 × 8	1	9.37	22.21	51.79	152.31	343.44	1069.85	2746.44
	2	5.88	15.45	36.37	101.95	223.59	675.41	1672.62
	4	4.32	10.69	25.65	70.20	151.42	400.66	993.64
	8	3.71	8.49	19.97	54.86	116.28	299.37	613.13

c) piecewise cubic elements

mesh		32×32	48×48	64×64	96×96	128×128	192×192	256×256
part	N_p	18434	41474	73730	165890	294914	663554	1179650
4 × 4	1	6.92	20.96	43.57	134.38	302.04	1132.82	3095.53
	2	4.63	14.67	31.22	93.13	202.29	733.73	1900.37
	4	3.63	10.15	22.47	64.75	135.05	425.55	1126.15
	8	3.35	8.43	17.37	49.81	101.33	302.33	721.11
8 × 8	1	9.84	26.31	49.59	138.16	279.80	893.95	2186.75
	2	6.89	19.09	36.90	98.83	195.67	619.08	1423.67
	4	5.26	13.55	26.84	69.53	136.72	408.11	982.86
	8	4.59	11.22	21.58	56.42	108.83	304.19	725.99

We observe from Table 7 and Table 8 that our strategy enables us to solve relatively large systems (up to 1.2 million equations fits into 3Gb of RAM) in a feasible amount of time (especially in parallel mode). We also observe that the total execution time for the systems of a particular size decreases as the order of approximation increases. This can be explained by the better performance of the HSL_MP62 when applied to the finite element matrices obtained using higher degree of approximation (see Table 10). It is also clear from Table 7 that the optimal execution time for a particular problem depends on the way that the domain is partitioned and on the number of processors that are used. In general, it is beneficial to increase the number of subdomains and the number of processors as the problem size is increased. In our experiments, we obtained the best timings in the case of 4×4 splitting for the systems of moderate size and 8×8 for large systems.

Table 9 gives parallel efficiency results for the timings reported in Table 7 (only the case of the lower tolerance for piecewise linear elements is presented as the results are similar using the tighter tolerance). The parallel efficiency grows as the problem size is increased, but drops as the number of subdomains in the partition is increased. The latter finding can be explained by the fact that when the number of subdomains is increased, the granularity of tasks running in parallel decreases. Also, the dimension of the interface problem (involving the unknowns that correspond to the finite element nodes at the boundaries between the subdomains) grows as the number of subdomains is increased. Bearing in mind that this interface problem is solved sequentially, it is clear that this leads to the lower efficiency (see [11],[20]). Consequently, the best values for parallel efficiency (for a fixed system size n) are observed for a partitioning into a smaller number of coarse partitions, although the usual trade-off between speed and parallel efficiency is self-evident.

Table 10 shows the performance (in MFlops) of the numerical factorisation phase of the HSL_MP62 code, when applied to discrete Dirichlet Laplacian matrices K_I in the context of the preconditioner (10). Note that the numerical factorisation phase dominates the overall process (see [20] for further numerical experiments), and we conclude that the HSL code performs well in our context, especially when using higher-order approximations.

Next we present timing results for the BiCGSTAB(2) algorithm with the inexact indefinite constraint preconditioner (12). Sample results are presented in Table 11. In all cases the multigrid method designed for linear elements is used to define K_* by taking an appropriate grid size.

Notice that the particular choice of meshes in these experiments is mo-

Table 9: Parallel efficiency for the BiCGSTAB(2) algorithm preconditioned by the exact indefinite constraint preconditioner (implemented using HSL_MP62 code) as a function of the problem size and domain decomposition. $\varepsilon = 10^{-7}$.

mesh		96×96	144×144	192×192	288×288	384×384	576×576	768×768
part	N_p	18434	41474	73730	165890	294914	663554	1179650
4×4	1	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	2	0.780	0.746	0.765	0.813	0.842	0.885	0.907
	4	0.512	0.545	0.567	0.637	0.692	0.808	0.828
	8	0.288	0.339	0.380	0.447	0.507	0.641	0.709
8×8	1	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	2	0.754	0.738	0.760	0.808	0.839	0.877	0.905
	4	0.484	0.512	0.526	0.571	0.617	0.705	0.754
	8	0.277	0.313	0.327	0.361	0.400	0.487	0.553

Table 10: Speed in MFlops obtained in the numerical factorisation phase of the matrix K_I by the code HSL_MP62 on a single processor. The results are given as a function of the problem size and the domain partition. Note that the theoretical peak performance of a single R10000 processor is 390MFlops.

a) piecewise linear elements

n	18434	41474	73730	165890	294914	663554	1179650
4×4	50.5	53.7	54.5	55.3	56.2	55.6	54.6
8×8	59.5	66.8	71.9	74.6	74.4	73.1	72.4

b) piecewise quadratic elements

n	18434	41474	73730	165890	294914	663554	1179650
4×4	101.9	124.2	141.5	150.8	170.6	176.1	177.6
8×8	84.7	106.1	123.0	126.4	161.7	169.8	177.7

c) piecewise cubic elements

n	18434	41474	73730	165890	294914	663554	1179650
4×4	97.3	125.5	145.7	173.3	188.2	199.4	197.5
8×8	80.6	104.3	122.1	149.3	168.4	185.6	192.8

tivated by the efficiency of the multigrid code D03EDF (for the maximal efficiency the number of grid lines in each coordinate direction is recommended to be $n_x = n_y = m \cdot 2^{\ell-1}+1$, see [17]), and the intention to have linear systems (10) of the same dimension for the differing degrees of ap-

Table 11: Timing results for the BiCGSTAB(2) algorithm preconditioned by the inexact indefinite constraint preconditioner (implemented using the sequential multigrid code D03EDF code from the NAG library) as a function of the problem size and the number of V(1,1) multigrid cycles.

a) piecewise linear elements

mesh	114×114	258×258	450×450	642×642	1026×1026	1794×1794
V_n	25994	133130	405002	824330	2105354	6436874
1 (1E-7)	4.59	57.86	273.73	687.71	2537.51	15174.78
1 (1E-9)	9.64	90.65	455.36	1273.98	3845.98	26011.29
3 (1E-7)	3.82	27.49	100.59	304.34	1212.45	4908.98
3 (1E-9)	6.83	54.28	258.94	693.55	3408.34	16205.72
5 (1E-7)	4.14	40.76	148.18	271.50	771.94	3007.11
5 (1E-9)	8.47	68.29	295.37	733.08	2235.98	9932.82

b) piecewise quadratic elements

mesh	57×57	129×129	225×225	321×321	513×513	897×897
V_n	25994	133130	405002	824330	2105354	6436874
1 (1E-7)	8.15	75.21	416.71	1026.43	3357.57	21269.96
1 (1E-9)	12.76	108.44	524.40	1390.27	4555.73	29200.19
3 (1E-7)	10.80	98.80	505.93	1259.39	4321.14	26705.45
3 (1E-9)	14.63	137.00	688.10	1831.39	5779.67	35638.64
5 (1E-7)	13.41	115.36	634.16	1512.04	5443.77	32179.16
5 (1E-9)	18.20	168.79	864.41	2222.49	6878.42	44398.16

c) piecewise cubic elements

mesh	38×38	86×86	150×150	214×214	342×342	598×598
V_n	25994	133130	405002	824330	2105354	6436874
1 (1E-7)	12.50	105.59	521.84	1355.90	4611.11	24189.16
1 (1E-9)	14.34	136.36	627.38	1681.89	5885.10	33703.14
3 (1E-7)	13.82	136.79	673.17	1725.11	6087.32	31098.43
3 (1E-9)	17.03	176.59	787.82	2173.89	7450.37	41712.59
5 (1E-7)	16.98	165.31	816.24	2114.38	7025.83	37583.36
5 (1E-9)	20.87	218.00	971.03	2662.55	8801.23	50543.17

proximation $1 \leq p \leq 3$. The latter enables easier comparison of the timing and convergence results. Timing results from Table 11 should be analysed in conjuction with the appropriate convergence results from Table 2. It is obvious that much larger systems can be solved in this case due to the lower memory requirements compared to the exact case. Whilst the largest dimension n that is tested is approximately $6.5 \cdot 10^6$, the memory associated with the CPUset facility that we used (3Gb) can accommodate even larger problems. We noted earlier that in the case of piecewise linear approximation it is beneficiary to increase gradually the number of V(1,1) multigrid cycles as the dimension n of the linear system (10) increases, whereas this approach did not show any benefits in the case of higher-order approximation ($p \geq 2$), due to the particular implementation that we employed. Consequently, the best execution times observed in the case of piecewise linear approximation are dependent upon n and V_n, while for the case of piecewise quadratic and cubic approximation the best choice is always $V_n = 1$.

Table 12 contains the timing results of the BiCGSTAB(2) algorithm using the inexact version of the preconditioner that is implemented using the parallel multigrid solver `mgd2`. We report our results only for the case of piecewise linear approximation (although the code can be applied in the same fashion as D03EDF to higher-order approximations). A particular choice of linear system dimensions is, as in the case of sequential multigrid solver, a consequence of the constraints related to the efficiency of the multigrid package. In the case of a single processor `mgd2` was essentially employed as a global multigrid solver (and can be directly compared with D03EDF). When executed on two processors, the unit square domain is subdivided into 2×1 partitions, and in the case of four processors, the domain is partitioned into 2×2 subdomains. Note that when using `mgd2` the number of processors and the number of subdomains must be the same.

Comparing the timing results in Table 12 (obtained using the `mgd2` code) with those in Table 11 (obtained using the NAG library code), we observe that the parallel `mgd2` code is 2-3 times slower than the NAG code. A possible explanation may be in the optimisation of the package (NAG library codes are tuned to ensure fast performance, whereas `mgd2` comes with no guarantees in this regard). One overhead when using the `mgd2` code is related to the format adjustments (our grid-centered data needed to be interpolated to a cell-centered format every time the multigrid solver was invoked, and the results returned again to grid-centered format). These facts notwithstanding, the `mgd2`-based code outperforms the NAG version if run on a sufficient number of processors.

In Table 13 we report on the parallel efficiency of the BiCGSTAB(2) algorithm with the `mgd2`-based inexact version of the preconditioner. The results are given only for the case $\varepsilon = 10^{-7}$. From the table it is obvious that the algorithm achieves a good scalability (despite the fact that only the preconditioning phase is parallelised). Parallel efficiency is virtually independent of the problem size and it grows as the number of V cycles is increased.

Table 12: Timing results for the BiCGSTAB(2) algorithm preconditioned by the inexact indefinite constraint preconditioner (implemented using the parallel multigrid code `mgd2`) as a function of the problem size and the number of V(1,1) multigrid cycles. The results are given for both tolerances $10^{-7}/10^{-9}$.

mesh		128×128	256×256	512×512	1024×1024	2048×2048
V_n	N_p	32770	131074	524290	2097154	8388610
1	1	17.51/25.46	111.14/182.04	664.11/1279.59	4338.85/6991.08	34648.81/60560.62
	2	13.12/19.04	83.46/137.09	499.35/ 965.06	3199.79/5193.92	25439.60/44595.56
	4	10.90/15.89	69.25/114.38	406.90/ 787.94	2632.83/4262.80	20772.57/36482.38
3	1	13.26/26.74	69.05/137.14	486.11/ 912.80	2727.08/6581.28	13684.98/47780.09
	2	9.23/18.68	45.03/ 89.66	314.89/ 592.73	1852.59/4477.16	8991.35/31434.17
	4	6.73/13.69	33.42/ 66.59	231.02/ 434.61	1408.68/3399.35	6579.22/22971.29
5	1	21.91/39.30	112.85/202.95	496.50/1065.07	1756.17/5089.85	8499.86/27178.64
	2	14.03/25.21	69.54/125.31	306.11/ 656.68	1077.31/3126.48	5266.23/16860.15
	4	9.84/17.69	47.15/ 84.97	207.93/ 446.08	725.59/2106.83	3529.89/11305.69

Table 13: Parallel efficiency results of the BiCGSTAB(2) algorithm preconditioned by the inexact indefinite constraint preconditioner (with the parallel multigrid code `mgd2` from Mudpack) as a function of the problem size and the number of V(1,1) multigrid cycles. $\varepsilon = 10^{-7}$.

mesh		128×128	256×256	512×512	1024×1024	2048×2048
V_n	N_p	32770	131074	524290	2097154	8388610
1	1	1.000	1.000	1.000	1.000	1.000
	2	0.668	0.666	0.665	0.678	0.681
	4	0.401	0.398	0.408	0.412	0.417
3	1	1.000	1.000	1.000	1.000	1.000
	2	0.718	0.767	0.772	0.736	0.761
	4	0.492	0.516	0.526	0.484	0.520
5	1	1.000	1.000	1.000	1.000	1.000
	2	0.780	0.811	0.811	0.815	0.807
	4	0.556	0.598	0.597	0.605	0.602

Finally, for completeness we give timing results of a direct sparse solver from the HSL library (HSL_MP42). This solver is based on the same principles as HSL_MP62, but can be applied to the solution of non-symmetric or indefinite sparse linear systems (see [20]). Here we examine the execution time and scalability of the direct solver, as a function of the problem size and the domain decomposition. While experimenting with the HSL_MP42 code, we found that the performance was crucially dependent upon an additional parameter controlling the number of unknowns that are simultaneously eliminated. We experimented with a range of values for this parameter (varying it from 32 to 160 by the increments of 16) and the timing results that we report here are the best ones. The results are summarised in Table 14. The numbers in brackets are the execution times of the factorisation phase (symbolic and numerical, see [20]). Note that the memory requirements prevent its application to very large problems. (Both HSL_MP42 and HSL_MP62 have the capability of storing the matrix factors out-of-core (see [11]), but since this badly impinges on performance due to the increased data traffic between the memory levels, this was not tested by us.)

Comparing the execution times from Table 14 with the our preconditioned solver (Table 7 and Table 8 for the exact case, and Table 11 and Table 12 for an inexact case) we see that iterative strategy is almost always faster (especially when moderate accuracy is required, as is usually the case in many engineering applications). The iterative strategy also requires less memory than the direct solver (especially the inexact case), thus it is possible to solve much larger linear systems.

6 Conclusions

A linear system is defined here to be "small" if it has less than 10^5 equations, "medium" if it has up to 10^6 equations, and "large" if it contains more than 10^6 equations. The timing results reported in the previous section suggest the following conclusions. For small systems, application of a direct solver is the most feasible option, especially if the coefficient matrix does not change and there are many different right hand sides. The constraint limiting the application of direct solvers is that sufficient RAM needs to be available. For the linear systems of medium size, when only moderate level of accuracy is required, one should consider an iterative method with an exact version of the indefinite constraint preconditioner. This alternative is particularly attractive if a parallel architecture with a sufficient amount of RAM is available. This method can also be appealing when the domain and discretisation do not change (e.g. if solving an eigenvalue problem), as the factorisation of K_I is only performed once. In addition, since the factorisation time is

Table 14: Timing results of the HSL_MP42 code when applied to the solution of the discrete biharmonic problem (10).

a) piecewise linear elements

mesh		96×96	144×144	192×192	288×288	384×384
part	N_p	18434	41474	73730	165890	294914
4×4	1	8.88 (8.65)	35.10 (34.38)	77.23 (75.07)	379.74 (374.74)	1146.85 (1132.37)
	2	5.65 (5.47)	21.18 (20.71)	45.70 (44.63)	227.63 (224.23)	625.95 (618.58)
	4	4.01 (3.88)	13.91 (13.61)	29.39 (28.78)	147.24(145.35)	401.00 (397.28)
8×8	1	8.09 (7.67)	21.86 (20.93)	45.53 (43.75)	184.12 (178.95)	681.30 (667.54)
	2	5.57 (5.37)	16.05 (15.51)	30.98 (30.01)	123.17 (119.98)	404.78 (397.09)
	4	4.38 (4.21)	13.12 (12.75)	23.39 (22.77)	89.35 (87.68)	270.49 (266.94)

b) piecewise quadratic elements

mesh		48×48	72×72	96×96	144×144	192×192
part	N_p	18434	41474	73730	165890	294914
4×4	1	6.19 (5.97)	21.15 (20.57)	60.37 (59.06)	265.52 (260.71)	735.19 (722.44)
	2	4.03 (3.87)	12.95 (12.53)	35.35 (34.46)	151.40 (148.27)	410.99 (403.22)
	4	2.90 (2.79)	8.73 (8.47)	22.60 (22.05)	92.48 (90.92)	247.40 (243.15)
8×8	1	5.93 (5.58)	18.75 (17.94)	34.74 (33.07)	139.01 (134.14)	347.71 (336.02)
	2	4.18 (3.97)	12.48 (12.03)	22.91 (21.98)	90.86 (88.05)	227.40 (220.52)
	4	3.27 (3.11)	9.33 (9.00)	17.10 (16.50)	69.91 (68.39)	163.86 (160.05)

c) piecewise cubic elements

mesh		32×32	48×48	64×64	96×96	128×128
part	N_p	18434	41474	73730	165890	294914
4×4	1	5.81 (5.52)	22.94 (22.11)	52.76 (51.00)	240.66 (234.10)	596.94 (580.96)
	2	3.86 (3.66)	14.84 (14.30)	32.40 (31.29)	139.50 (135.53)	336.36 (327.04)
	4	2.82 (2.69)	10.55 (10.21)	21.01 (20.37)	87.68 (85.70)	207.90 (202.83)
8×8	1	5.84 (5.43)	18.14 (17.19)	39.10 (36.91)	131.61 (125.67)	311.17 (297.07)
	2	4.18 (3.96)	12.39 (11.88)	26.24 (25.07)	85.68 (82.46)	206.85 (198.99)
	4	3.35 (3.19)	9.47 (9.12)	19.66 (18.94)	65.28 (63.44)	157.94 (153.54)

a small proportion of the total execution time in general (providing that a suitable partitioning of the domain is adopted, cf. Table 6), this approach is also suitable for adaptive algorithms, when the mesh and/or the domain geometry change. In the case of large systems, the iterative approach with an inexact version of the preconditioner is the only feasible alternative, both in terms of the execution time and memory requirements. It is especially efficient in the case of piecewise linear approximation, where we can achieve virtually the same convergence rate as in the case of an exact preconditioner by tuning the number of multigrid V cycles. We conjecture that this should be possible in the case of higher-order approximations, providing that appropriate multigrid solvers (based on higher-order restriction and interpolation operators) are used.

References

[1] I. Babuška, J. Osborn, J. Pitkäranta: *Analysis of mixed methods using mesh dependent norms*, Math. Comp., **35**(1980), 1039–1062.

[2] A. Battermann, E.W. Sachs: *Block Preconditioners for KKT systems in PDE-Governed Optimal Control Problems*, Proceedings of Workshop on Fast Solutions of Discretised Optimisation Problems (R.H.W. Hoppe, K.-H. Hoffmann, V. Schulz, eds.), Birkhäuser Verlag, Berlin, 2001, 1–18.

[3] P.E. Bjørstad, B.P. Tjøstheim: *Efficient algorithms for solving a fourth order equation with the spectral Galerkin method,* SIAM J. Sci. Comp., **18**(1997), 621–632.

[4] P.E. Bjørstad, B.P. Tjøstheim: *High Precision Solutions of Two Fourth Order Eigenvalue Problems,* Computing **63**(1999), 97–107.

[5] F. Brezzi, M. Fortin: Mixed and Hybrid Finite Element Methods, Springer-Verlag, New York, 1991.

[6] B.M. Brown, P.K. Jimack, M.D. Mihajlović: *A New Direct Solver for a Class of Mixed Finite Element Problems*, Appl. Numer. Math., **38**(1-2)(2001), 1–20.

[7] B.M. Brown, E.B. Davies, P.K. Jimack, M.D. Mihajlović: *A numerical investigation of the solution of a class of fourth-order eigenvalue problems*, Proc. Roy. Soc. (London), **A 456** (2000), 1505–1521.

[8] J.J. Dongarra, I.S. Duff, D.C. Sorensen, H.A. van der Vorst: Numerical Linear Algebra for High-Performance Computers, SIAM, Philadelphia, 1998.

[9] R. Glowinski, O. Pironneau: *Numerical methods for the first biharmonic equation and for the two-dimensional Stokes problem*, SIAM Rev., **21**(1979), 167–212.

[10] N.I.M. Gould, M.E. Hribar, J. Nocedal: *On the solution of equality constrained quadratic programming problems arising in optimisation*, SIAM J. Sci. Comp., **23**(4)(2001), 1375–1394.

[11] HSL 2000: A collection of Fortran codes for large scale scientific computation. `http://www.numerical.rl.ac.uk`

[12] C. Keller, N.I.M. Gould, A.J. Wathen: *Constraint preconditioning for indefinite linear systems*, SIAM J. Matrix Anal. Appl., **21**(4)(2000), 1300–1317.

[13] V.A. Kozlov, V.A. Kondratev, V.G. Mazya: *On sign variation and the absence of "strong" zeros of solutions of elliptic equations*, Mat. USSR Izv., **34**(1990), 337–353.

[14] MG Net: `http://www.mgnet.org/mgnet-codes-bunner.html`

[15] K. Moffat: *Viscous and restrictive eddies near a sharp corner*, J. Fluid. Mech., **18**(1964), 1–18.

[16] `http://www.mpi-forum.org/docs/mpi-20-html/mpi2-report.html`

[17] Numerical Algorithms Group: NAG manual, Fortran Library Mark 20, 2002.

[18] P. Peisker: *A multilevel algorithm for the biharmonic problem,* Numer. Math., **46**(1985), 623–634.

[19] I. Perugia, V. Simoncini: *Block-diagonal and indefinite symmetric preconditioners for mixed finite element formulations,* Numer. Lin. Alg. Appl., **7**(7–8)(2000), 585–616.

[20] J.A. Scott: *A parallel frontal solver for finite element applications*, Int. J. Numer. Meth. Engng, **50**(2001), 1131–1144.

[21] D.J. Silvester, M.D. Mihajlović: *A black-box multigrid preconditioner for the biharmonic equation*, BIT, submitted.

[22] G.L.G. Sleijpen, D.R. Fokkema: *BiCGSTAB(ℓ) for linear equations involving unsymmetric matrices with complex spectrum*, Electron. Trans. Numer. Anal., **1**(1993), 11–32.

[23] K. Stüben: *A review of algebraic multigrid*, J. Comput. Appl. Maths **128**(2001), 281–309.

[24] A.J. Wathen: *Realistic eigenvalue bounds for the Galerkin Mass Matrix*, IMA J. Numer. Anal., **7**(1987), 449–457.

[25] P. Wesseling: *MGD1 – A robust and efficient multigrid method*, In: Lecture Notes in Mathematics **960**(1982), 614–630.