An Efficient Direct Solver for a Class of Mixed Finite Element Problems

B.M. Brown[1], P.K. Jimack[2], M.D. Mihajlović[1]

Abstract. In this paper we present an efficient, stable and parallelizable direct method for the solution of the (indefinite) linear algebraic systems that arise in the solution of fourth order partial differential equations (PDEs) using mixed finite element approximations. The method is intended particularly for use when multiple right-hand sides occur, and when high accuracy is required in these solutions. The algorithm is described in some detail and its performance is illustrated through the numerical solution of a biharmonic eigenvalue problem: where the smallest eigenpair is approximated using inverse iteration after discretization via the Ciarlet-Raviart mixed finite element method.

Key words: sparse Gaussian elimination, mixed finite element method, biharmonic eigenproblem.

AMS Subject Classification: 31A30, 65F50, 65N30

1 Introduction

Fourth order partial differential operators play a significant role in modeling a wide variety of physical processes. For example, the clamped plate problem and the buckling plate problem, which arise in the theory of plates and shells [34] are important equations used in the design of structures. However, in comparison with second order operators, a relatively small amount of literature is dedicated to such problems. Nevertheless, a variety of numerical procedures for solving fourth order problems have been investigated, usually based upon the Rayleigh–Ritz method, using either high-order polynomial approximations or different finite element formulations (see, for example, [5],[7],[22],[28],[30],[31],[37]).

When high-order polynomials are used, the resulting linear algebra typically requires the solution of dense systems of equations for which standard Gaussian elimination schemes are appropriate. We do not consider such cases in this paper. Instead we focus on the use of finite element techniques. When using conforming elements these techniques fall into essentially two distinct classes: either a standard weak form of the differential equation may be obtained (by two applications of the divergence theorem) yielding a problem whose solution lies in H^2, or a mixed method may be used in which a secondary variable is used to directly approximate the second derivative of the primary variable. This latter problem has a solution in $H^1 \times H^1$ which means that a finite element solution may be sought based upon straightforward C^0 Lagrangian elements [14],[33],[38]. The former problem requires the use of C^1 finite elements (e.g. [2],[6],[14]) which are not so simple to work with in more than one dimension. Since, for this paper, we are concerned with problems in greater than one dimension (mainly two dimensions in fact), we now restrict attention to a mixed finite element method based upon C^0 elements. The purpose of the paper is to propose an efficient

---

[1]Department of Computer Science, Cardiff University of Wales, P.O. Box 916, Cardiff CF2 3XF, Wales, UK.
[2]School of Computer Studies, University of Leeds, Leeds LS2 9JT, UK.

direct solver for the resulting algebraic equations which makes use of the underlying structure of this mixed finite element formulation.

In the next section we provide some motivation for wishing to use a direct, rather than an iterative, linear solver by describing a family of biharmonic eigenproblems. The use of inverse iteration (for example) leads to a sequence of linear systems to be solved: each with the same indefinite matrix but a different right-hand side. Moreover, it is demonstrated that very high accuracy is required in the solution of these systems in order to get an accurate representation of the fundamental eigenfunction. Section 3 then outlines the particular finite element discretization that we wish to consider, which is originally due to Ciarlet and Raviart [15]. Despite the fact that the biharmonic problem is self-adjoint and elliptic, this mixed finite element formulation is shown to give rise to an indefinite algebraic system which is known to be highly ill-conditioned [23]. Recently, a number of authors have considered the solution of these linear systems using iterative techniques based upon efficient multigrid algorithms (see [23],[29],[35] for example). These methods appear to perform well in terms of their speed and low storage requirements, however, they do require the existence (or construction) of a multilevel mesh hierarchy which is not always convenient. Moreover, when multiple right-hand sides must be solved for, and very high accuracy is required, there are clearly potential advantages to be gained from using a direct, rather than an iterative, technique.

Having provided some motivation in Sections 2 and 3, Section 4 introduces details of the direct method that we propose for this class of indefinite system. It is based upon permuting the initial system to a block tridiagonal form, and then applying block Gaussian elimination on this system. The memory requirements are minimized by exploiting the sparsity pattern of the coefficient matrix and, because of the efficient re-use of data obtained in the process of factorization, the method is suitable for problems with multiple right-hand sides. The proposed method shows high accuracy when tested on a range of problems, as demonstrated in Section 5, and performs more efficiently than the use of sparse direct solvers on the original linear systems (also shown in Section 5). The paper concludes with a brief discussion of the algorithm presented, including issues such as its efficient parallelization (using the standard LAPACK libraries for example, see [1]) and the possible use of fast Laplacian solvers on regular structured grids.

2   A fourth order eigenvalue problem

This section is intended to provide the motivation for the rest of the paper by introducing some practical problems for which the discretized biharmonic equation must be solved on unstructured meshes, with multiple right-hand sides and to a very high level of accuracy. There are likely to be other situations where the same requirements arise, such as in iterative design processes for example, but we do not consider these here.

Consider the following two biharmonic eigenvalue problems:

$$\Delta^2 u = \lambda u \quad \text{in } \Omega, \qquad u = \frac{\partial u}{\partial n} = 0 \quad \text{on } \partial \Omega, \tag{1}$$

which is usually referred to as the clamped plate eigenproblem, and

$$\Delta^2 u = -\lambda \Delta u \quad \text{in } \Omega, \qquad u = \frac{\partial u}{\partial n} = 0 \quad \text{on } \partial \Omega, \tag{2}$$

which is the buckling plate eigenproblem [34]. These problems have some interesting features. In particular, the eigenfunctions corresponding to each of the eigenvalues in these problems may be shown to possess extreme oscillatory behaviour near to any sufficiently small corners of the domain Ω. The first numerical evidence of this behaviour dates from 1972 when Bauer and Reiss [4] reported this effect for the eigenfunction corresponding to the smallest eigenvalue of the clamped plate problem on the unit square. However, because this effect only occurs in areas where the eigenfunction values are exceptionally small (relative to values near the centre of the domain), the result was thought to be surprising and was questioned at that time. Nevertheless, later theoretical results (e.g. [8],[16],[17],[24],[25]), as well as more accurate computations ([37]), confirmed this initial work to be correct.

For a full discussion of the oscillatory structure of the solutions to (1) near a corner the reader is referred to [16], which builds upon the earlier papers [8] and [24] (see also [25]). By considering asymptotic expansions for the eigenfunctions, u, of (1) along the symmetry line of a corner of the domain (with internal angle θ), a number of results may be derived. In particular, if r_n is the distance along the symmetry line to a local extremal value of u and s_n is the distance to a zero of this function, where n increases with decreasing distance from the corner, then it may be shown that

$$\frac{r_n}{r_{n+1}} \sim \frac{s_n}{s_{n+1}} \sim \exp(\pi/\beta), \tag{3}$$

as $n \to \infty$. Here β is the imaginary part of the complex solution $p = \alpha + i\beta$ of the transcendental equation

$$p - 1 + \frac{\sin((p-1)\theta)}{\sin\theta} = 0 \tag{4}$$

which has the smallest real part. In addition the ratio of the magnitudes of the consecutive extrema can also be derived asymptotically as

$$\frac{t_n}{t_{n+1}} \sim \left(\frac{r_n}{r_{n+1}}\right)^\alpha \sim \exp(\alpha\pi/\beta), \tag{5}$$

as $n \to \infty$. (Note that (4) has only real solutions if $\theta > 0.8128\pi = 146°30'$, from which it may be concluded that the oscillations are only present for angles smaller than this critical value.)

In recent years many authors have tried to verify numerically these theoretical results using a variety of computational techniques (see, for example, [5],[12],[22],[37]). Since the oscillatory features occur very close to the corners and are damped out very quickly, most of these attempts have, due to discretization errors and numerical inaccuracy, failed to find more than one sign change. Clearly this is not sufficient to verify equation (3). A notable exception to this is the recent paper of Bjørstad and Tjøstheim ([7]) in which the authors report five correct sign changes for the principal eigenfunction. For this work a spectral Legendre–Galerkin method is used and computations are performed using quadruple precision arithmetic.

The above considerations demonstrate that the solutions to eigenproblems of the form (1) or (2) on domains which contain corners of sufficiently small internal angles ($\theta < 0.8128\pi$ in two dimensions for example) can be highly oscillatory in the neighbourhood of these corners. Moreover, as one approaches such a corner these sign changes become closer together and the magnitude of the function gets smaller. If a numerical method based upon the use of finite elements is to be able to resolve some of these oscillations therefore, it is clear that a very fine mesh is required close to these corners. Furthermore, the algebraic equations which result from such a discretization will need to be solved very accurately if the relative error in the magnitudes of the first few oscillations

3

is to be acceptable (since the size of the solution at these local extrema is many orders of magnitude smaller than it is away from the corners).

Before introducing our proposed direct method for the solution of these algebraic equations, which is designed to satisfy the above requirements, the next section briefly describes the finite element discretization. This is based upon the use of C^0 Lagrange finite elements and so may be applied on an unstructured mesh on an arbitrary polygonal domain in two dimensions.

3   The Ciarlet–Raviart mixed formulation

Let us consider the biharmonic problem

$$\Delta^2 u = f, \tag{6}$$

defined on some domain Ω in R^2, subject to the Dirichlet boundary conditions

$$u = \frac{\partial u}{\partial n} = 0 \qquad \text{on } \partial\Omega. \tag{7}$$

In (6) the right-hand side function f can represent either the usual load vector if we solve the simple biharmonic equation, or it can be

$$f = \lambda u \qquad \text{or} \qquad f = -\lambda \Delta u \tag{8}$$

for the clamped plate and the buckling plate eigenproblems, respectively. To simplify the notation in what follows we shall use the symbol f for all of these cases of the right-hand side vector throughout this paper, unless explicitly stated otherwise. In this section we consider the Ciarlet–Raviart mixed method for solving the biharmonic problem in the plane. Many authors have considered this technique as a starting point in developing methods for solving problem (6)–(7) (see, for example, [9],[23],[29],[30],[35]). Analysis of mixed methods using mesh-dependent norms is given in [3].

One can construct a variety of weak formulations of the biharmonic Dirichlet eigenproblem that are of the following abstract form.

Problem 3.1 *Given real Banach spaces V and W, and bilinear forms $a(\cdot,\cdot)$, $b(\cdot,\cdot)$, $c(\cdot,\cdot)$ on $V \times V$, $V \times W$ and $W \times W$ respectively, find a pair $\{v, u\} \in V \times W$ such that*

$$\begin{aligned}
a(v, w_1) + b(w_1, u) &= 0 & \forall w_1 \in V, \\
b(v, w_2) - c(w_2, u) &= (f, w_2) & \forall w_2 \in W.
\end{aligned}$$

The mixed method of Ciarlet and Raviart [15] is a finite element method for one such formulation. Introducing an auxiliary function v such that

$$\begin{aligned}
\Delta u &= v, \\
\Delta v &= f
\end{aligned} \tag{9}$$

and using Green's formula we may obtain a weak formulation of the original problem (6)–(7) in the same form as Problem 3.1 with $c(\cdot,\cdot) \equiv 0$.

Problem 3.2 *Find* $\{v, u\} \in H^1(\Omega) \times H_0^1(\Omega)$ *such that*

$$
\begin{aligned}
(v, w_1) + b(w_1, u) &= 0 && \forall w_1 \in H^1(\Omega), \\
b(v, w_2) &= -(f, w_2) && \forall w_2 \in H_0^1(\Omega),
\end{aligned}
$$

where the quadratic form $b(\cdot, \cdot)$ and the L^2–inner product are given by

$$
b(\rho, \psi) = \int_\Omega \nabla \rho \cdot \nabla \psi \, d\underline{x}, \qquad (\rho, \psi) = \int_\Omega \rho \psi \, d\underline{x}. \tag{10}
$$

As usual, $H^1(\Omega)$ represents the Sobolev space of $L^2(\Omega)$ functions whose first partial derivatives are all in $L^2(\Omega)$, and

$$
H_0^1(\Omega) = \left\{ \psi \in H^1(\Omega) : \psi|_{\partial\Omega} = \left.\frac{\partial\psi}{\partial n}\right|_{\partial\Omega} = 0 \right\}.
$$

This formulation is studied by Ciarlet and Raviart in [15]. They show that if u and $\partial\Omega$ are sufficiently smooth Problem 3.2 is uniquely solvable, and that the part of the solution u in the pair $\{v, u\} \in H^1(\Omega) \times H_0^1(\Omega)$ also solves the original problem (6)–(7).

If we consider a family of regular and quasi-uniform triangulations, $\mathcal{T}^h$ (where h is the diameter of the largest triangle), of the domain Ω (see, for example, Ciarlet [14]), then we may define the space

$$
\mathcal{S}_m^h = \{p \in C^0(\bar\Omega) : p|_T \in P_m(T), \ \forall T \in \mathcal{T}^h\},
$$

where $P_m(T)$ is the space of polynomials of degree at most m over triangle T. From this we may define the following finite-dimensional subspaces of $H^1(\Omega)$: $V^h = \mathcal{S}_m^h$ and $W^h = \mathcal{S}_m^h \cap H_0^1(\Omega)$. Then the Ciarlet–Raviart mixed finite element method approximates the solution $\{v, u\}$ of Problem 3.2 with the solution of the following problem.

Problem 3.3 *Find* $\{v^h, u^h\} \in V^h \times W^h$ *such that*

$$
\begin{aligned}
(v^h, w_1) + b(w_1, u^h) &= 0 && \forall w_1 \in V^h, \\
b(v^h, w_2) &= -(f, w_2) && \forall w_2 \in W^h.
\end{aligned}
$$

Moreover, if the Brezzi stability conditions are satisfied (see [10],[30] for example), then the unique solvability of these discrete equations can be established. Indeed, the following error estimate may be derived ([30]):

$$
\|\Delta u - v^h\|_{H^1(\Omega)} + \|u - u^h\|_{H_0^1(\Omega)} \le C\left(\inf_{w_1 \in V^h} \|\Delta u - w_1^h\|_{H^1(\Omega)} + \inf_{w_2 \in W^h} \|u - w_2^h\|_{H_0^1(\Omega)} \right),
$$

where C is independent of h.

Consider the choice of bases $\{\phi_i\}_{i=1}^{n_b + n_e}$ for V^h and $\{\phi_i\}_{i=1}^{n_b}$ for W^h, where ϕ_i are the usual Lagrange "hat" functions of degree m and the degrees of freedom are enumerated so that those that lie on the boundary of Ω are ordered last (from $n_b + 1$ to $n_b + n_e$). In [30], for example, it is shown that for $m \ge 2$ the Brezzi stability conditions are satisfied for these spaces and in [15] the following asymptotic error estimate is given:

$$
\|u - u^h\|_{H^1(\Omega)} + \|\Delta u - v^h\|_{L^2(\Omega)} \le c\|u\|_{H^{m+2}(\Omega)} h^{m-1}. \tag{11}
$$

(Note that one may also use spaces of differing polynomial degree for V^h and W^h but for simplicity we do not discuss such a generalization here.) It is possible to rewrite Problem 3.3 as the following linear algebraic system (in block matrix form):

$$\begin{bmatrix} M & K^t \\ K & O \end{bmatrix} \begin{bmatrix} \underline{v} \\ \underline{u} \end{bmatrix} = - \begin{bmatrix} \underline{0} \\ \underline{f} \end{bmatrix}, \tag{12}$$

where $[K]_{ij} = b(\phi_j, \phi_i)$, $[M]_{ij} = (\phi_j, \phi_i)$, K^t is the transpose of K and $[\underline{f}]_j = (f, \phi_j)$. The vectors $\underline{v} \in \Re^{n_b+n_e}$ and $\underline{u} \in \Re^{n_b}$ denote the coefficients of v^h and u^h respectively when they are expanded in terms of their basis functions. That is

$$v^h = \sum_{i=1}^{n_b+n_e} v_i \phi_i \qquad \text{and} \qquad u^h = \sum_{i=1}^{n_b} u_i \phi_i.$$

The coefficient matrix in (12) is indefinite and is clearly not diagonally dominant. Moreover, this matrix becomes extremely ill-conditioned as the mesh parameter $h \to 0$. Hence the fast, stable and accurate solution of (12) is a task that presents significant difficulties when the mesh becomes very fine.

Recall that when solving a biharmonic eigenproblem f takes one of the forms given in (8). Hence (12) is really an algebraic eigenvalue problem. Assuming that the solutions of interest correspond to the smallest eigenvalues in magnitude then it is appropriate to apply an iterative technique such as the inverse power method or a Lanczos algorithm (see, for example, [21]). In either case it will be necessary to solve a system with the same coefficient matrix (the indefinite matrix of (12)) at each iteration. Since this matrix is sparse one possible approach would be to make use of iterative techniques, such as Krylov subspace methods for example (e.g. [21],[36]). However, when very high accuracy is required and the system is highly ill-conditioned these algorithms rely almost entirely on the quality of the preconditioner used. Good preconditioners for other mixed finite element problems, for solving Stokes' equations or the linear elasticity equations for example, have been successfully developed in recent years (see [32], [36]) but these clearly do not apply to systems of the form (12) which have very different spectral properties.

Another technique that may be considered is to solve the Schur complement system, in which (12) is reduced to

$$KM^{-1}K^t\underline{u} = \underline{f}. \tag{13}$$

Now the coefficient matrix is symmetric and positive-definite however it is still extremely ill-conditioned (with a condition number of $O(h^{-4})$ as $h \to 0$, [23]). Hence, for an iterative solver, the same difficulties of finding an appropriate preconditioner arise here as for the original saddle-point problem. Moreover, lack of sparsity in the matrix $KM^{-1}K^t$, which is of dimension $n_b \times n_b$, prevents the use of a direct solver for (13) when the number of unknowns is large.

Probably the most successful approach that has been used to solve systems of the form (12) and (13) are multigrid algorithms. Details of how the multigrid approach may be applied to such problems can be found in [23],[30] for example. These algorithms are capable of producing very accurate solutions however they are most efficient when only moderate accuracy is required (in which case fewer cycles need to be completed). Their major disadvantage however is the need for a hierarchy of grids to be present, which can be a major complication (and significant overhead) when non-uniform meshes are required on complex geometries. Also, like other iterative approaches, the multigrid solvers in [23],[30] do not use any information from the first solution of a problem to improve the efficiency of subsequent solves using the same matrix with a different right-hand vector.

This therefore leads us to consider the use of direct methods for the solution of multiple systems of the form (12). Since the coefficient matrix is not diagonally dominant the use of sparse Gaussian elimination (or LU factorization) will require a pivoting strategy that ensures stability as well as a small amount of fill-in. There are a number of software tools that are designed for this type of problem (e.g. [18],[27]) however, of those that we considered here, we found the code called SuperLU ([18]) to be the most efficient. When compared with iterative techniques on the finite element grids described in Section 5 below, this code not only executed fastest but also provided results with a greater number of significant digits of accuracy.

We have not explicitly contrasted the performance of SuperLU against the multigrid approach of [23] since our objective is to consider methods appropriate for arbitrary unstructured grids which are as straightforward as possible to use. Nevertheless we have considered the asymptotic cost of this sparse direct method on systems of the form (12) as the mesh size h is decreased uniformly. Table 1 shows the number of floating point operations required by SuperLU for various uniform meshes when using piecewise polynomials of degree 3. As can be seen the solver appears to have a complexity of approximately $O(n^{1.8})$ for this class of problem (where n is the dimension of the linear system and the average value for the exponent is calculated from the three finest meshes), with a complexity of approximately $O(n^{1.3})$ for subsequent right-hand sides. This second asymptotic rate is similar to that obtained by the multigrid approach in [23] however the cost of the sparse factorization is somewhat greater. This factorization cost may be contrasted with the number of operations required to solve finite element discretizations of Laplace's equation on similar grids using SuperLU. These figures are given for piecewise cubic polynomial approximations in Table 2. For these problems we observe an approximate cost of $O(n^{1.6})$ for the sparse factorization of a matrix of dimension n. It is this disparity that motivates the direct method that is described in the following section.

grid	32×32	40×40	48×48	56×56	64×64	72×72	80×80	88×88	96×96
n	18434	28802	41474	56450	73730	93314	115202	139394	165890
fact.	1.196E9	2.668E9	5.205E9	7.812E9	1.274E10	1.887E10	2.776E10	3.816E10	5.369E10
solve	1.004E7	1.823E7	2.961E7	4.240E7	5.976E7	8.078E7	1.058E8	1.438E8	1.796E8
f + s	1.206E9	2.686E9	5.235E9	7.854E9	1.280E10	1.895E10	2.787E10	3.830E10	5.387E10

grid ratio	32/40	40/48	48/56	56/64	64/72	72/80	80/88	88/96	Ave.
α_f	1.798	1.833	1.317	1.831	1.668	1.832	1.669	1.962	**1.821**
α_s	1.337	1.330	1.165	1.285	1.279	1.280	1.610	1.277	**1.389**

Table 1: The number of floating point operations required to factorize and solve (forward and backward substitution) $n \times n$ systems of the form (12) resulting from piecewise cubic finite element discretizations of (6) on uniform grids. Here α_f and α_s are the exponents in work estimates of the form Cn^α for the factorization and the solution steps respectively.

4   A new direct method

In this section we introduce an efficient direct method for solving the system (12). The approach is related to a technique which has been used in constrained optimization algorithms, (see [20] for example), and is essentially based upon a block Gaussian elimination in a manner which allows us to exploit the sparsity patterns arising from our particular application.

grid	32×32	40×40	48×48	56×56	64×64	72×72	80×80	88×88	96×96
n	9025	14161	20449	27889	36481	46225	57121	69169	82369
fact.	4.055E7	8.513E7	1.567E8	2.690E8	4.226E8	6.214E8	8.912E8	1.169E9	1.564E9
solve	1.112E6	1.942E6	3.055E6	4.494E6	6.242E6	8.320E6	1.076E7	1.345E7	1.659E7
N_{SLU}	4.166E7	8.707E7	1.598E8	2.735E8	4.288E8	6.297E8	9.020E8	1.182E9	1.581E9

grid ratio	32/40	40/48	48/56	56/64	64/72	72/80	80/88	88/96	Ave.
α_f	1.646	1.661	1.740	1.683	1.629	1.740	1.419	1.666	**1.608**
α_s	1.238	1.233	1.244	1.224	1.214	1.216	1.166	1.199	**1.194**

Table 2: The number of floating point operations required to factorize and solve (forward and backward substitution) $n \times n$ linear systems resulting from piecewise cubic finite element discretizations of Laplace's equation on uniform grids. Here α_f and α_s are the exponents in work estimates of the form Cn^α for the factorization and the solution steps respectively.

Let us consider more carefully the sparsity pattern of the coefficient matrix of the system (12) where, for simplicity, only piecewise linear approximations are made to u and v in (9) in the first instance. As in the previous section, denote by n_b the number of interior nodes in the triangulation of Ω, and by n_e the number of nodes on the boundary $\partial\Omega$. The sparsity pattern of the coefficient matrix corresponding to a 4 element by 4 element uniform mesh (hence $n_b = 9$ and $n_e = 16$) is shown on the Fig. 1. The boundaries between the blocks denoted in (12) are marked by the second row and column of tick marks (after row and column 25). In addition another row and column of tick marks have been added to define a new block decomposition. This is created by splitting the blocks M and K in (12) according to whether the corresponding degree of freedom is in the interior of Ω or on the boundary $\partial\Omega$. In this way a 3×3 block system of the form

$$\left[\begin{array}{ccc} M_i & M_c^t & K_i^t \\ M_c & M_b & K_c^t \\ K_i & K_c & O \end{array}\right] \left[\begin{array}{c} \underline{v}_i \\ \underline{v}_b \\ \underline{u} \end{array}\right] = - \left[\begin{array}{c} \underline{0} \\ \underline{0} \\ \underline{f} \end{array}\right] \tag{14}$$

is obtained. Observe that in (14) the block M_i is of size $n_b \times n_b$, M_b is of size $n_e \times n_e$, K_i is of size $n_b \times n_b$ and the interface rectangular blocks M_c and K_c have sizes $n_e \times n_b$ and $n_b \times n_e$ respectively. The unknown vector and the right-hand side are partitioned accordingly. (Note that when the mesh is large, unlike for Fig. 1 below, the number of nodes inside the body, n_b, will be much greater than the number of nodes satisfying the essential boundary conditions, n_e. In fact, for a two-dimensional problem, it will generally be the case that $n_b \sim n_e^2$ as $h \to 0$.)

The system (14) may be simply rearranged by interchanging the first and third row blocks and then eliminating the unknowns $\underline{v}_i$ from this new third block of equations, to obtain

$$\begin{array}{llll} K_i\underline{v}_i + & K_c\underline{v}_b & = & -\underline{f} \\ M_c\underline{v}_i + & M_b\underline{v}_b & +K_c^t\underline{u} = & \underline{0} \\ & (M_c^t - M_iK_i^{-1}K_c)\underline{v}_b & +K_i^t\underline{u} = & M_iK_i^{-1}\underline{f} \,. \end{array} \tag{15}$$

The system is now in block tridiagonal form and may be solved by means of Gaussian elimination (sometimes referred to in this special case of a tridiagonal (or block-tridiagonal) system as the Thomas method). This yields the following equations:

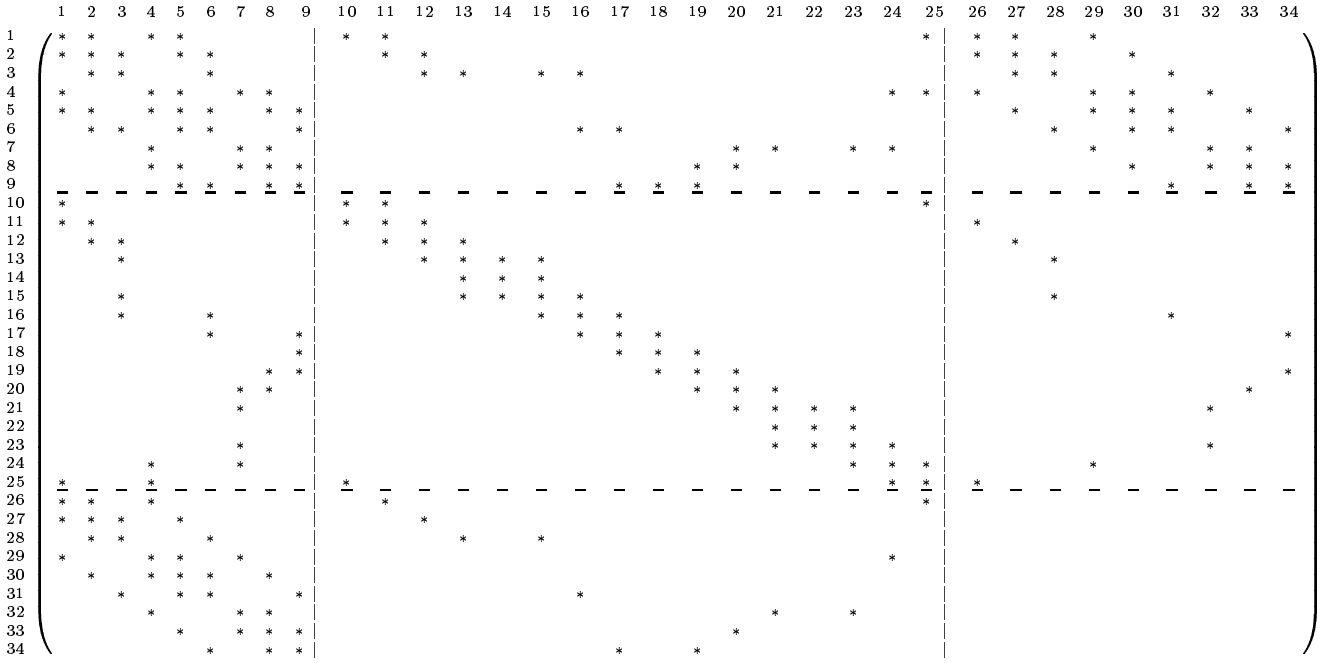$$B_1 = -K_i^{-1}K_c, \quad Z_1 = -K^{-1}\underline{f}, \tag{16}$$

8

Figure 1: The sparsity pattern of the matrix in (12) corresponding to a uniform 4 by 4 grid.

$$B_2 = -(M_b + M_c B_1)^{-1} K_c^t, \quad Z_2 = -(M_b + M_c B_1)^{-1} M_c Z_1 \tag{17}$$

$$Z_3 = \left(K_i^t + \left(M_c^t - M_i K_i^{-1} K_c \right) B_2 \right)^{-1} \left(M_i K_i^{-1} \underline{f} - \left(M_c^t - M_i K_i^{-1} K_c \right) Z_2 \right). \tag{18}$$

Note that the unknown vector $\underline{u}$ must now be equal to Z_3. Consequently, the algorithm for solving the linear system (14) can be organized in 3 macro steps as follows.

Step 1. The equations (16) may be rewritten as

$$K_i B_1 = -K_c, \quad K_i Z_1 = -\underline{f}. \tag{19}$$

This is really a single system with $n_e + 1$ right-hand sides which may be expressed as

$$K_i[B_1 \mid Z_1] = -[K_c \mid \underline{f}]. \tag{20}$$

The sparse matrix K_i is symmetric and positive definite. In order to solve the systems (20), we may therefore use sparse Cholesky factorization of the coefficient matrix K_i with any ordering of the unknowns (e.g. for minimum fill-in or bandedness, etc.), followed by a forward and back-substitution phase.

It should be noted that when it is necessary to solve the biharmonic problem (1) repeatedly for different right-hand side vectors, the coefficient matrix in (14) remains the same. This will enable us to re-use a significant amount of data calculated in the first solve. At this stage, the sparse Cholesky factorization of the block K_i and the matrix B_1 will remain unchanged, hence for subsequent calculations this step reduces to a single forward and back-substitution phase to obtain the vector Z_1.

Step 2. We now consider equations (17). These may be rewritten as

$$(M_b + M_c B_1)B_2 = -K_c^t, \quad (M_b + M_c B_1)Z_2 = -M_c Z_1,$$

or equivalently,
$$(M_b + M_c B_1)[B_2 \mid Z_2] = -[K_c^t \mid M_c Z_1]. \tag{21}$$
The sizes of the matrices involved in (21) are as follows: $M_b + M_c B_1$ is of size $n_e \times n_e$, K_c^t is $n_e \times n_b$ and $M_c Z_1$ is $n_e \times 1$. Consequently, the unknown matrices, B_2 and Z_2, are of size $n_e \times n_b$ and $n_e \times 1$ respectively. Hence (21) may be considered as $n_b + 1$ systems of n_e equations, each with the same coefficient matrix. This coefficient matrix, $M_b + M_c B_1$, is both dense and non-symmetric. The conventional approach to solving (21) would therefore be to perform a full LU factorization of the coefficient matrix (with pivoting as necessary) followed by $n_b + 1$ forward and backward substitutions. It may be observed however that since $n_b \gg n_e$ the forward and backward substitution phase is more time-consuming than the factorization (since it has complexity $O(n_b n_e^2)$ which is greater than $O(n_e^3)$). Given that the matrix K_c^t on the right-hand side of (21) is sparse (containing $O(n_e)$ non-zero entries), an alternative is to compute an explicit inverse of $M_b + M_c B_1$ in $O(n_e^3)$ operations followed by multiplication with the sparse augmented matrix $[K_c^t | M_c Z_1]$, also in $O(n_e^3)$ operations. Even this multiplication is not completely necessary however since it is advantageous to use $(M_b + M_c B_1)^{-1}$ and the sparse matrix K_c^t separately in Steps 3.3 and 3.6 below, rather than the dense block B_2. Hence, having computed $(M_b + M_c B_1)^{-1}$ it remains only to compute Z_2 at this step.

We may again note that many of the calculations in this step need only be performed once when the biharmonic equation is solved many times with different right-hand sides. In particular, since the coefficient matrix in (21) depends only upon the coefficient matrix of the original system, (12), the inverse need only be calculated once and than re-used. Consequently, on all subsequent applications of this algorithm, Step 2 reduces to just two matrix–vector multiplications in order to calculate the vector Z_2.

Step 3. Finally, we consider the third block of equations (18). These may be rewritten in the form
$$(K_i^t + (M_c^t + M_i B_1)B_2)Z_3 = -(M_i Z_1 + (M_c^t + M_i B_1)Z_2). \tag{22}$$
Here the matrix $M_c^t + M_i B_1$ is of size $n_b \times n_e$, B_2 is $n_e \times n_b$ and K_i^t is $n_b \times n_b$. The overall coefficient matrix is consequently of size $n_b \times n_b$. Similar considerations for the right-hand side show it to be an $n_b \times 1$ matrix. Hence it is necessary to solve a single system of n_b equations. At first sight this appears to be prohibitively expensive since the coefficient matrix in (22) is both dense and non-symmetric, thus requiring $O(n_b^3)$ operations (and $O(n_b^2)$ memory locations) for a solution. Fortunately however a simple modification of formula (18) can lead to a significant reduction in the complexity of this step.

To compute the action of the inverse matrix $(K_i^t + (M_c^t + M_i B_1)B_2)^{-1}$ in (18) we may apply the Sherman–Morrison–Woodbury formula (see [21] for example) which gives a closed expression for the inverse of a rank k correction of an $n \times n$ matrix. In the case where $k \ll n$ this formula can be very efficient. Given the matrices $A \in R^{n \times n}$ and $U, V^t \in R^{n \times k}$, we have
$$(A + UV^t)^{-1} = A^{-1} - A^{-1}U(I + V^t A^{-1}U)^{-1}V^t A^{-1}. \tag{23}$$
This means that it is possible to reduce the problem of inverting the original matrix $A + UV^t$ of size $n \times n$ to the inversion of the matrix A plus inversion of the matrix $I + V^t A^{-1}U$ of dimension $k \times k$. Note that in our case $n = n_b$ and $k = n_e$ (and recall that $n_e \ll n_b$). Moreover the matrix $A = K_i^t$ is sparse, symmetric, positive-definite and has already been factorized so this modification will be especially efficient here. Introducing the notation
$$U_2 = M_c^t + M_i B_1, \quad \underline{g} = -(M_i Z_1 + (M_c^t + M_i B_1)Z_2),$$

it is possible to write equation (18) in the form

$$Z_3 = (K_i^t + U_2 B_2)^{-1} \underline{g}. \tag{24}$$

Applying formula (23) to (24) then gives

$$Z_3 = (K_i^t)^{-1} \underline{g} - (K_i^t)^{-1} U_2 \left[I + B_2 (K_i^t)^{-1} U_2 \right]^{-1} B_2 (K_i^t)^{-1} \underline{g}, \tag{25}$$

which may in turn be written as

$$Z_3 = Z_3' - Z_3'',$$

where

$$Z_3' = (K_i^t)^{-1} \underline{g} \quad \text{and} \quad Z_3'' = (K_i^t)^{-1} U_2 \left[I + B_2 (K_i^t)^{-1} U_2 \right]^{-1} B_2 (K_i^t)^{-1} \underline{g}.$$

Calculation of Z_3 according to the formula (25) may now be organized as follows.

Step 3.1) Calculation of Z_3'. Rewrite the equation

$$Z_3' = (K_i^t)^{-1} \underline{g} \tag{26}$$

as

$$K_i^t Z_3' = \underline{g}.$$

This is a sparse, symmetric, positive-definite system of n_b equations. Moreover the Cholesky factorization of K_i^t has already been performed in Step 1 above.

Step 3.2) Calculation of Z_3''. To calculate the vector Z_3'' the following expression is used:

$$Z_3'' = (K_i^t)^{-1} U_2 \left[I + B_2 (K_i^t)^{-1} U_2 \right]^{-1} B_2 (K_i^t)^{-1} \underline{g}. \tag{27}$$

Recalling (26), this may be rewritten as

$$Z_3'' = (K_i^t)^{-1} U_2 \left[I + B_2 (K_i^t)^{-1} U_2 \right]^{-1} B_2 Z_3'.$$

We now split the calculation of Z_3'' into a number of substeps.

1) Calculate $X_1 = B_2 Z_3'$. Here we use the fact that $B_2 = (M_b + M_c B_1)^{-1}(-K_c^t)$, thus yielding

$$X_1 = (M_b + M_c B_1)^{-1}(-K_c^t Z_3').$$

This requires a sparse matrix–vector multiplication of complexity $O(n_e)$ followed by a dense matrix–vector multiplication of complexity $O(n_e^2)$. (If at Step 2 above we had calculated B_2 explicitly and then formed X_1 at this step by the dense matrix–vector multiplication $B_2 Z_3'$, this would have required $O(n_b n_e) = O(n_e^3)$ operations.)

2) Calculate $X_2 = \left[I + B_2 (K_i^t)^{-1} U_2 \right]^{-1} X_1$. This calculation may be performed in several phases. First, the above equation is rewritten as

$$[I + B_2 (K_i^t)^{-1} U_2] X_2 = X_1.$$

Here, the matrix $I + B_2 (K_i^t)^{-1} U_2$ is of size $n_e \times n_e$ and the vectors X_2 and X_1 are of size $n_e \times 1$. We now have

Phase 2a). Calculate

$$Y_1 = (K_i^t)^{-1} U_2,$$

that is,

$$K_i^t Y_1 = U_2.$$

This is a multiple right-hand side system (i.e. n_e systems) for which the coefficient matrix has already been factorized.

Phase 2b). Perform the matrix multiplication

$$Y_2 = B_2 Y_1.$$

Here we again benefit from not having computed B_2 explicitly since it is possible to express

$$Y_2 = (M_b + M_c B_1)^{-1}(-K_c^t Y_1).$$

This requires a sparse matrix multiplication with a dense matrix, of complexity $O(n_e^2)$, followed by a dense matrix multiplication of complexity $O(n_e^3)$. (Again we note the disadvantage of computing Y_2 by direct multiplication using the dense matrix B_2, which would have a complexity of $O(n_e^2 n_b) = O(n_e^4)$.)

Phase 2c). Perform the matrix addition

$$Y_3 = I + Y_2,$$

where I is the $n_e \times n_e$ identity matrix.

Phase 2d). Solve the system

$$Y_3 X_2 = X_1$$

where Y_3 is of size $n_e \times n_e$ and X_1 and X_2 are of size $n_e \times 1$. The matrix Y_3 is dense and non-symmetric so it is necessary to perform LU factorization (possibly with pivoting for stability) which requires $O(n_e^3)$ operations. This is then followed by forward and back-substitution.

Again it should be noted that the matrix Y_3 is dependent only upon matrices which do not include the right-hand side vector or the matrices Z_1 and Z_2 (which themselves depend upon the right-hand side). Therefore, in the calculations that involve solving multiple systems with the same coefficient matrix, phases 2a)–2d) need be performed only once, for the first right-hand side. For subsequent solves all that needs to be done at this substep is a single forward and back-substitution to calculate X_2 from the factorization of Y_3.

3) Calculate $X_3 = U_2 X_2$. The complexity of performing this dense matrix–vector multiplication is $O(n_b n_e) = O(n_e^3)$. However, by replacing U_2 by $M_c^t + M_i B_1$, which is in turn equal to $M_c^t - M_i K_i^{-1} K_c$, one obtains the expression

$$X_3 = (M_c^t - M_i K_i^{-1} K_c) X_2 = M_c^t X_2 - M_i K_i^{-1}(K_c X_2).$$

Given that the sparse factorization of K_i is already known, this may be computed with a complexity of $O(n_b) = O(n_e^2)$ plus the cost of one forward and backward substitution using the sparse factors of K_i. (In practice this forward and backward substitution is the dominant cost.)

4) Calculate $Z_3'' = (K_i^t)^{-1} X_3$. This may be expressed as

$$K_i^t Z_3'' = X_3,$$

which is a single system for which the coefficient matrix has already been factorized. Hence, only application of forward and back-substitution is necessary.

Step 3.3) Calculation of $Z_3 = Z_3' - Z_3''$. Simple vector subtraction now allows us to calculate the required vector Z_3 (which is in fact the unknown vector $\underline{u}$).

The complete algorithm outlined above is presented in Figure 2 (with $\underline{f}$ used to denote the right-hand side vector). In this figure the steps of the algorithm that need only be performed once when multiple right-hand sides are being solved for are displayed in bold. The column next to each step displays the asymptotic complexity of that step, where it is assumed that $n_b \sim n_e^2$, the cost of a sparse Cholesky factorization of the $n_b \times n_b$ matrix K_i is $O(n_b^f)$ and the cost of a single forward and backward substitution is $O(n_b^s)$. Note that it is assumed that $s \geq 1$ when determining the dominant contributions to steps 3.1 and 3.10 in the figure. (Also note that the steps in the figure have been renumbered from those above for further clarity.)

STEP 1

    1.1) Factorize $K_i = L_K L_K^t$ $O(n_b^f)$

    1.2) Solve $L_K L_K^t B_1 = -K_c$ $O(n_b^{s+1/2})$

    1.3) Solve $L_K L_K^t Z_1 = -\underline{f}$ $O(n_b^s)$

STEP 2

    2.1) Form $M_b + M_c B_1$ $O(n_e^2)$

    2.2) Invert $M_b + M_c B_1$ $O(n_e^3)$

    2.3) Form $-M_c Z_1$ $O(n_e)$

    2.4) Form $Z_2 = (M_b + M_c B_1)^{-1}(-M_c Z_1)$ $O(n_e^2)$

STEP 3

    3.1) Form $\underline{g} = -M_i Z_1 - M_c^t Z_2 + M_i K_i^{-1} K_c Z_2$ $O(n_b^s)$

    3.2) Solve $L_K L_K^t Z_3' = \underline{g}$ $O(n_b^s)$

    3.3) Form $X_1 = (M_b + M_c B_1)^{-1}(-K_c^t)Z_3'$ $O(n_e^2)$

    3.4) Form $M_c^t + M_i B_1$ $O(n_b n_e)$

    3.5) Solve $L_K L_K^t Y_1 = M_c^t + M_i B_1$ $O(n_b^{s+1/2})$

    3.6) Form $Y_2 = (M_b + M_c B_1)^{-1}(-K_c^t)Y_1$ $O(n_e^3)$

    3.7) Form $Y_3 = I + Y_2$ $O(n_e)$

    3.8) Factorize $Y_3 = L_3 U_3$ $O(n_e^3)$

    3.9) Solve $L_3 U_3 X_2 = X_1$ $O(n_e^2)$

    3.10) Form $X_3 = (M_c^t - M_i K_i^{-1} K_c)X_2$ $O(n_b^s)$

    3.11) Solve $L_K L_K^t Z_3'' = X_3$ $O(n_b^s)$

    3.12) Form $Z_3 = Z_3' - Z_3''$ $O(n_b)$

Figure 2: Summary of the algorithm for solving (14) along with the asymptotic complexity of each step (note that $n_b \sim n_e^2$). Those steps shown in bold need not be repeated when second or subsequent right-hand sides are solved for.

Although the derivation of this direct algorithm is described in terms of piecewise linear polynomial approximations, the use of higher order Lagrange polynomial finite element spaces has no significant effect on either the algorithm itself or the asymptotic costs presented. (Where n_b generalizes to be the total number of unknowns inside the body, or on a Neumann boundary, and n_e the total number of unknowns on the essential boundary.)

Clearly the overall computational cost of the algorithm will depend upon the performance characteristics of the sparse solver used for the systems involving the symmetric positive-definite matrix K_i. In particular the most expensive step of the first solve using this algorithm will have a complexity of either $O(n_b^f)$ (step 1.1), $O(n_b^{s+1/2})$ (steps 1.2 and 3.5) or $O(n_b^{3/2})$ (steps 2.2, 3.4 and 3.8). Reference to Table 2 suggests that when SuperLU ([18]) is used for the sparse solves $f \approx 1.6$ and $s \approx 1.2$, hence steps 1.2 and 3.5 are likely to dominate with an asymptotic cost of approximately $O(n_b^{1.7})$. For second and subsequent right-hand sides the dominant terms will have a cost of $O(n_b^s) \approx O(n_b^{1.2})$. Inspection of Table 1 suggests that there is likely to be a significant saving when using this new algorithm in comparison to the application of a general sparse direct solver to the original system.

5 Numerical results

In this section we report on the performance of the proposed algorithm when applied to linear systems of the form (14) obtained from the mixed finite element discretization of the biharmonic equation on unstructured grids (using SuperLU for the factorization and solution of systems involving K_i). In order to demonstrate the accuracy of the method we first present some results for the biharmonic eigenvalue problem (1) on a unit square domain $\Omega = (0,1)^2$ using piecewise cubic C^0 elements (since, by (11), these are more accurate than piecewise linears or piecewise quadratics). These results are contrasted with the known analytic expressions for the ratio of the distance between zeros as a corner is approached, (3), and with the numerical results in [7] obtained using a high order spectral Legendre-Galerkin method with quadruple precision arithmetic. Having established the accuracy of the solver its complexity is then contrasted with that of SuperLU ([18]) applied to the full system (14), which is the best alternative direct solver that we have been able to find. All comparisons are given in terms of the number of floating point operations required, as in Tables 1 and 2, so as to ensure that the results are neither platform nor implementation dependent.

When $\theta = \pi/2$ the solution of equation (4) with smallest real part may be shown to be $p = 3.739593284 + 1.11902448i$. Hence, by substituting $\alpha = 3.739593284$ and $\beta = 1.11902448$ into (3) and (5), we obtain

$$\frac{r_n}{r_{n+1}} \sim \frac{s_n}{s_{n+1}} \sim 16.567429848 \quad \text{and} \quad \frac{t_n}{t_{n+1}} \sim 36267.54987 \tag{28}$$

as $n \to \infty$. (Recall from section 2 that r_n is the distance along the bisector of the angle θ to the nth local extremal value of the eigenfunction u, t_n is the magnitude of this extremum, and s_n the distance to the nth zero.)

In Table 3 we present some numerical results for the first of these ratios, obtained by solving (1) on the unit square using piecewise cubic C^0 elements on a sequence of unstructured meshes over a quarter of the domain (making use of symmetry at $x = 1/2$ and $y = 1/2$ and applying appropriate Neumann conditions at these boundaries). N denotes the dimension of the linear system resulting from each of the non-uniformly refined meshes, which are designed to allow greater resolution of the eigenfunction in the corner. Table 4 shows calculations of the same ratios for a different sequence of non-uniformly refined meshes, on an eighth of the domain (utilizing further symmetry along the bisector of the corner). In each case the calculations are based upon the approximation of the principal eigenfunction (i.e. that which corresponds to the smallest eigenvalue) using inverse

iteration, and the final column of the tables is used to present the best numerical results obtained in [7]. The non-uniform meshes used to produce the other columns of these tables have a mesh size, h, of $O(10^{-7})$ near the corner and $O(10^{-2})$ near the centre of the domain; with a gradual transition between the two. The precise details of the meshes depend upon the exact grid sizes. Moreover, for the calculations on an eighth of the domain the mesh is divided using polar rather than Cartesian coordinates.

N	42229	94177	166897	Results from [7]
s_1/s_2	16.5674314	16.5674538	16.5674561	16.56745701
s_2/s_3	16.5674172	16.5674281	16.5674285	16.56742775
s_3/s_4	16.5674192	16.5674263	16.5674282	16.5674282
s_4/s_5	16.5671211	16.5678311	16.5675317	16.58008884

Table 3: The ratio of distances between consecutive zeros calculated using piecewise cubic C^0 elements on a sequence of non-uniformly refined grids on a quarter of the unit square domain.

N	15052	56035	219136	Results from [7]
s_1/s_2	16.5674214	16.5674511	16.5674567	16.56745701
s_2/s_3	16.5674022	16.5674231	16.5674282	16.56742775
s_3/s_4	16.5674164	16.5674208	16.5674276	16.56742802
s_4/s_5	16.5672813	16.5672315	16.5676374	16.58008884

Table 4: The ratio of distances between consecutive zeros calculated using piecewise cubic C^0 elements on a sequence of non-uniformly refined grids on an eighth of the unit square domain.

Inspection of Tables 3 and 4 reveals a good correspondence between our double precision finite element calculations and the quadruple precision spectral results presented in [7]. Furthermore, the ratios obtained in each of these cases are remarkably close to the asymptotic limit of 16.56742776 for the continuous problem (as given in (28)). It may also be observed from (28) that the ratio of the magnitudes, t_n, of the consecutive extrema for this angle is comparatively large. Consequently the size of the oscillations decreases very rapidly as the corner is approached. For example, the magnitude of the fifth extremum from the centre is approximately 10^{-23} times the size of the eigenfunction at the centre. This implies that very high accuracy is required in both the discretization and the direct solver in order to have any significant digits of accuracy in this value. Similar results to those in Tables 3 and 4 may also be obtained for the buckling plate problem (2) and so a similar degree of accuracy is also required: see [26] for details.

Further demonstrations of the accuracy of the method come from its application to more general domains (indeed justification for the use of the finite element method, rather than spectral techniques such as that in [7] for example, stems from its applicability to quite general geometries). In [11] we present a number of applications of the solver described above on a variety of different domains (both convex and non-convex and with curved and straight boundaries). The common feature in each case being the requirement to solve systems with many right-hand sides to a very high numerical accuracy.

We now focus on the computational expense of the new direct solver that we propose. In particular, we wish to verify the claim made in the previous section that we expect to obtain better performance than by applying a sparse direct solver to the discrete fourth order system (14)

directly. In Tables 5 and 6 results are presented for piecewise cubic approximations on the same two different sequences of meshes used to generate the ratios of distances between consecutive zeros (quoted in Tables 3 and 4). In each case the subscript f is used to denote the number of operations required for the first solution of the system (14) and the subscript s denotes the number of operations required for each subsequent solve. "New" refers to the algorithm proposed in Section 4 (using SuperLU ([18]) to factor and solve the sparse positive-definite systems) and SLU refers to the application of SuperLU directly to (14). The ratios quoted allow comparison between the two approaches to be made. Note also that in each table the grid sizes quoted indicate only the number of intervals on each boundary of the domain, however, since the meshes are unstructured, the total number of degrees of freedom, N, gives a more accurate indication of the overall size of the problem.

grid	64×64	80×80	96×96	112×112	128×128	144×144	160×160	176×176	192×192
N	19093	29509	42229	57217	74545	94177	116113	140353	166897
New_f	7.314E8	1.455E9	2.596E9	4.216E9	6.273E9	9.199E9	1.230E10	1.698E10	2.146E10
New_s	5.917E6	1.027E7	1.601E7	2.360E7	3.239E7	4.314E7	5.504E7	6.972E7	8.554E7
SLU_f	1.250E9	2.663E9	4.698E9	7.705E9	1.302E10	1.893E10	2.859E10	3.795E10	5.100E10
SLU_s	1.035E7	1.844E7	2.889E7	4.241E7	6.036E7	8.106E7	1.072E8	1.346E8	1.667E8
Ratio_f	0.585	0.546	0.553	0.547	0.482	0.486	0.431	0.447	0.421
Ratio_s	0.562	0.557	0.554	0.556	0.537	0.532	0.513	0.518	0.513

Table 5: Computational costs, both absolute (measured in floating point multiplications) and relative (Ratio), of the proposed (New) algorithm and SuperLU (SLU) for the first (f) and subsequent (s) solution of the system (14) obtained from piecewise cubic C^0 elements on a sequence of non-uniformly refined grids on a quarter of the unit square domain.

grid	4×28	8×32	16×48	32×96	64×192
N	2086	4804	15052	56035	219136
New_f	2.250E7	8.445E7	5.447E8	4.750E9	4.536E10
New_s	2.774E5	9.792E5	4.880E6	2.787E7	1.569E8
SLU_f	3.824E6	4.930E7	7.934E8	1.014E10	1.227E11
SLU_s	2.116E5	1.089E6	7.612E6	4.991E7	3.111E8
Ratio_f	5.883	1.713	0.686	0.469	0.370
Ratio_s	1.311	0.899	0.641	0.558	0.504

Table 6: Computational costs, both absolute (measured in floating point multiplications) and relative (Ratio), of the proposed (New) algorithm and SuperLU (SLU) for the first (f) and subsequent (s) solution of the system (14) obtained from piecewise cubic C^0 elements on a sequence of non-uniformly refined grids on an eighth of the unit square domain.

Inspection of Tables 5 and 6 clearly shows that in each case the cost of subsequent solves is almost insignificant in comparison to the cost of the first solve. In addition, as the discrete systems become larger, the advantage of using the new algorithm becomes more and more significant. Moreover, these operation counts compare very favourably with the solution of (14) using other available software: either sparse direct ([27]) or iterative ([21],[32]).

6 Conclusions

In this paper we present a new efficient method for solving the linear algebraic system which arises from the mixed finite element approximation of the biharmonic problem. The method is based upon reducing the initial system to block tridiagonal form and then applying block Gaussian elimination. In this way we are able to exploit the sparsity of the blocks, allowing significant reductions in execution time and memory requirements in comparison with more general sparse direct solvers (although such solvers are used *within* the new algorithm). The approach is especially suitable for problems where the solution of multiple systems with the same coefficient matrix is required to a high level of accuracy, such as with biharmonic eigenvalue computations using inverse iteration or when solving the biharmonic equation with multiple load vectors.

Although the method introduced here is designed with unstructured finite element grids in mind it appears to also have some potential for use with structured grids, where fast Laplacian solvers (based upon cyclic reduction for example (e.g. [21])) might be used. Another area in which the work may be extended is in the parallel implementation of the algorithm. All of the substeps shown in Fig. 2 may be implemented in parallel: the main requirements being a parallel sparse direct solver (steps 1.1, 1.2, 1.3, 3.1, 3.2, 3.5, 3.10 and 3.11) and a parallel dense solver (steps 2.2, 3.8 and 3.9), along with a number of matrix–vector and matrix–matrix multiplications. Use of parallel BLAS or ScaLAPACK (see [1],[13]) would provide efficient implementations of all of these operations except for the sparse factorization and solve. It may be noted however that SuperLU, for example, has also been implemented in parallel [19] (or a parallel banded solver could be used), and so a complete parallel algorithm appears to be achievable in an efficient manner.

Acknowledgements

References

[1] E. Anderson, Z. Bai, C.H. Bischof, J. Demmel, J.J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, D.C. Sorensen: *LAPACK Users' Guide Release 2.0,* SIAM, Philadelphia, 1995.

[2] J.H. Argyris, D.W. Scharpf: *The TUBA family of plate elements for the matrix displacement method,* Aeronautical J., **72**(1968), 701–709.

[3] I. Babuška, J. Osborn, J. Pitkäranta: *Analysis of Mixed Methods Using Mesh Dependent Norms,* Math. Comp., **35**(152), 1039–1062.

[4] L. Bauer, E. Reiss: *Block five diagonal matrices and the fast numerical computation of the biharmonic equation,* Math. Comp., **26**(1972), 311–326.

[5] H. Behnke: *A numerically rigorous proof of curve veering for a plate,* Second Gregynog Workshop on Computational and Analytic Problems in Spectral Theory, Cardiff University, 1996.

[6] K. Bell: *A refined triangular plate bending finite element,* Int. J. Numer. Meth. Eng., **1**(1969), 101–122.

[7] P.E. Bjørstad, B.P. Tjøstheim: *A note on high precision solutions of two fourth order eigenvalue problems,* preprint, 1998.

[8] H. Blum, R. Rannacher: *On the boundary value problem of the biharmonic operator on domains with angular corners,* Math. Meth. in Appl. Sci., **2**(1980), 556–581.

[9] D. Braess, P. Peisker: *On the Numerical Solution of the Biharmonic Equation and the Role of Squaring Matrices for Preconditioning,* IMA J. Numer. Anal., **6**(1986), 393–404.

[10] F. Brezzi: *On the existence, uniqueness and approximation of saddle point problems arising from Lagrangian multipliers,* RAIRO, **8**(1974), 129–151.

[11] B.M. Brown, E.B. Davies, P.K. Jimack, M.D. Mihajlović: *Solving the biharmonic eigenproblem by the finite element method,* in preparation.

[12] G. Chen, M. Coleman, J. Zhou: *Analysis of vibration eigenfrequencies of a thin plate by the Keller–Rubinow wave method I: clamped boundary conditions with rectangular or circular geometry,* SIAM J. Appl. Math., **51**(1991), 967–983.

[13] J. Choi, J.J. Dongarra,R. Pozo, D.W. Walker: *ScaLAPACK: A scalable linear algebra library for distributed memory concurrent computers,* Proceedings of the Fourth Symposium on the Frontiers of Massively Parallel Computation, IEEE Computer Society Press, 120–127.

[14] P.G. Ciarlet: *The Finite Element Method for Elliptic Problems,* Stud. Math. Appl., 4, North–Holland, Amsterdam, 1978.

[15] P.G. Ciarlet, P. Raviart: *A mixed finite element method for the biharmonic equation,* In: Mathematical Aspects of Finite Elements in Partial Differential Equations, Academic Press, New York, 1974, pp. 125–145.

[16] C.V. Coffman: *On the structure of solutions $\Delta^2 u = \lambda u$ which satisfy the clamped plate conditions on a right angle,* SIAM J. Math. Anal., **13**(1982), 746–757.

1969,

[17] E.B. Davies: *L^p spectral theory of higher-order elliptic differential operators,* Bull. London Math. Soc., **29**(1997), 513–546.

[18] J.W. Demmel, S.C. Eisenstat, J.R. Gilbert, X.S. Li and J.W.H. Liu: *A Supernodal Approach to Sparse Partial Pivoting,* To appear in SIAM J. Matrix Anal. Appl.

[19] J.W. Demmel, S.C. Eisenstat, J.R. Gilbert and X.S. Li *An Asynchronous Parallel Supernodal Algorithm for Sparse Gaussian Elimination,* To appear in SIAM J. Matrix Anal. Appl.

[20] P. E. Gill, W. Murray, M.A. Saunders and M.H. Wright: *A Schur-complement method for sparse quadrtic programming,* In: Reliable Numerical Computation, OUP, Oxford, 1990, pp. 113–138.

[21] G.H. Golub, C.F. Van Loan: *Matrix Computations,* Second Edition, The John Hopkins University Press, Baltimore, 1989.

[22] W. Hackbusch, G. Hoffman: *Results of the eigenvalue problem for the plate equation,* ZAMP, **31**(1980), 730–739.

[23] M.R. Hanisch: *Multigrid preconditioning for the biharmonic Dirichlet problem,* SIAM J. Numer. Anal., **30**(1)(1993), 184–214.

[24] V.A. Kondrat'ev: *Boundary problems for elliptic equations in domains with conical or angular points,* Trans. Moscow Math. Soc., **16**(1967), 227–313.

[25] V.A. Kozlov, V.A. Kondrat'ev, V.G. Maz'ya: *On sign variation and the absence of "strong" zeros of solutions of elliptic equations,* Mat. USSR Izv., **34**(1990), 337–353.

[26] M.D. Mihajlović: *A numerical investigation of some problems associated with the biharmonic operator,* Ph.D. Thesis, Department of Computer Science, Cardiff University, 1999.

[27] Numerical Analysis Group: *NAG manual, Fortran Library Mark 13,* Vol. 4, 1988.

[28] M.P. Owen: *Asymptotic first eigenvalue estimates for the biharmonic operator on a rectangle,* preprint 1996, PhD thesis, King's College, London.

[29] P. Peisker: *A Multilevel Algorithm for the Biharmonic Problem,* Numer. Math., **46**(1985), 623–634.

[30] V.V. Shaidurov: *Multigrid Methods for Finite Elements,* Kluwer, Dodrecht, 1995.

[31] J. Shen: *Efficient spectral Galerkin method I: direct solvers of second and fourth order equations using Legendre polynomials,* SIAM J. Sci. Comput., **15**(1991), 1440–1451.

[32] D. Silvester, A. Wathen: *Fast iterative solution of stabilised Stokes systems, part II: Using general block preconditioners,* SIAM J. Numer. Anal., **31**(5)(1994), 1352–1367.

[33] G. Strang, G.J. Fix: *An Analysis of the Finite Element Method,* Prentice-Hall, Englewood-Cliffs, 1973.

[34] S. Timoshenko, S. Woinowsky–Krieger: *Theory of plates and shells,* McGraw-Hill, New York, 1959.

[35] R. Verfürth: *A multilevel algorithm for mixed problems,* SIAM J. Numer. Anal., **21**(2) (1984), 264–271.

[36] A. Wathen, D. Silvester: *Fast iterative solution of stabilised Stokes systems, part I: Using simple diagonal preconditioners,* SIAM J. Numer. Anal., **30**(3)(1993), 630–649.

[37] C. Wieners: *A numerical existence proof of nodal lines for the first eigenfunction of the plate equation,* Arch. Math., **66**(1996), 420–427.

[38] O.C. Zienkiewicz: *The Finite Element Method,* Third edition, McGraw-Hill, London, 1986.