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

B.M. Brown[1], P.K. Jimack[2], M.D. Mihajlović[3]

Abstract. In this paper we present an efficient, accurate 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 [36], 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],[23],[30],[32],[33],[39]).

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],[35],[41]. 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

[1]Department of Computer Science, University of Wales, Cardiff, P.O. Box 916, Cardiff CF24 3XF, Wales, UK.
[2]School of Computer Studies, University of Leeds, Leeds LS2 9JT, UK.
[3]Centre for Novel Computing, Department of Computer Science, The University of Manchester, Oxford Road, Manchester M13 9PL, England.

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 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 [24]. Recently, a number of authors have considered the solution of these linear systems using iterative techniques based upon efficient multigrid algorithms (see [24],[31],[37] 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 [36]. 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. Later theoretical results (e.g. [8],[16],[17],[26],[27]), as well as more accurate computations ([39]), confirmed this initial work.

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 [26] (see also [27]). 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],[23],[39]). 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 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 subject to the Dirichlet boundary conditions

$$\Delta^2 u = f \quad \text{on } \Omega, \qquad u = \frac{\partial u}{\partial n} = 0 \qquad \text{on } \partial\Omega. \tag{6}$$

defined on some domain Ω in R^2 bounded by the smooth boundary $\partial\Omega$. 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

$$f = \lambda u \qquad \text{or} \qquad f = -\lambda\Delta u \tag{7}$$

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) (see, for example, [9],[24],[31],[32],[37]). Analysis of mixed methods using mesh-dependent norms is given in [3]. Introducing an auxiliary function v such that

$$\Delta u = v, \qquad \Delta v = f \tag{8}$$

and using Green's formula we may obtain a weak formulation of the original problem (6).

Problem 3.1 *Find $\{v, u\} \in H^1(\Omega) \times H^1_0(\Omega)$ such that*

$$\begin{aligned} (v, w_1) + b(w_1, u) &= 0 & \forall w_1 \in H^1(\Omega), \\ b(v, w_2) &= -(f, w_2) & \forall w_2 \in H^1_0(\Omega), \end{aligned}$$

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

$$b(\rho, \psi) = \int_\Omega \nabla\rho \cdot \nabla\psi \, d\underline{x} \quad \text{and} \quad (\rho, \psi) = \int_\Omega \rho\psi \, d\underline{x}.$$

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 $H^1_0(\Omega) = \{\psi \in H^1(\Omega) : \psi|_{\partial\Omega} = 0\}$. This formulation is studied by Ciarlet and Raviart in [15]. They show that if u and $\partial\Omega$ are sufficiently smooth Problem 3.1 is uniquely solvable, and that the part of the solution u in the pair $\{v, u\} \in H^1(\Omega) \times H^1_0(\Omega)$ also solves the original problem (6).

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

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

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 3.1 with the solution of the following problem.

Problem 3.2 *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],[32] for example), then the unique solvability of these discrete equations can be established.

Consider the choice of bases $\{\phi_i\}_{i=1}^{n_i+n_b}$ for V^h and $\{\phi_i\}_{i=1}^{n_i}$ for W^h, where ϕ_i are the usual Lagrange "hat" functions of degree m, n_i denotes the number of nodes in triangulation $\mathcal{T}^h$ which lie in the interior of the domain Ω, and n_b denotes the number of nodes in $\mathcal{T}^h$ which lie on the boundary $\partial\Omega$ (note that the degrees of freedom are enumerated so that those that lie on the boundary of Ω are ordered last from $n_i + 1$ to $n_i + n_b$). In [32], for example, it is shown that for $m \geq 2$ the Brezzi stability conditions are satisfied for these spaces and in [15] the following asymptotic error estimate is given:

$$\|u - u^h\|_{H^1(\Omega)} + \|\Delta u - v^h\|_{L^2(\Omega)} \leq c\|u\|_{H^{m+2}(\Omega)} h^{m-1} \ . \tag{9}$$

It is possible to rewrite Problem 3.2 as the following linear algebraic system (in block matrix form):

$$\begin{bmatrix} M & K^t \\ K & O \end{bmatrix} \begin{bmatrix} \underline{v} \\ \underline{u} \end{bmatrix} = - \begin{bmatrix} \underline{0} \\ \underline{f} \end{bmatrix}, \tag{10}$$

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 $\underline{v} \in \Re^{n_i+n_b}$ and $\underline{u} \in \Re^{n_i}$ denote the coefficients of v^h and u^h respectively when they are expanded in terms of their basis functions. That is

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

The coefficient matrix in (10) 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 and accurate solution of (10) 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 (7). Hence (10) 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, [22]). In either case it will be necessary to solve a system with the same coefficient matrix (the indefinite matrix of (10)) 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. [22],[34], [38]). 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.

Another technique that may be considered is to solve the Schur complement system, in which (10) is reduced to

$$KM^{-1}K^t\underline{u} = \underline{f}. \tag{11}$$

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 $h \to 0$, [24]). 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 $n_i \times n_i$, prevents the use of a direct solver for (11) when the number of unknowns is large.

Probably the most successful approach that has been used to solve systems of the form (10) and (11) are multigrid algorithms. Details of how the multigrid approach may be applied to such problems can be found in [24],[32] for example. However, the major disadvantage of the multigrid approach 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 [24],[32] 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 (10). 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. [19],[29]). However, of those that we considered here, we found the code called SuperLU ([19]) 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 [24] 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 (10) 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 $O(n^{1.4})$ for subsequent right-hand sides. This second asymptotic rate is similar to that obtained by the multigrid approach in [24] (which is also a little worse than $O(n)$) 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.

4 A new direct method

In this section we introduce an efficient direct method for solving the system (10). The approach is related to a technique which has been used in constrained optimization algorithms, (see [21] for

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
$N_{\rm SLU}$	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 (10) resulting from piecewise cubic finite element discretizations of (6) on uniform grids using SuperLU with the best available ordering. 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.

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
$N_{\rm SLU}$	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 using SuperLU with the best available ordering. 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.

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.

As in the previous section, denote by n_i the number of interior nodes in the triangulation of Ω, and by n_b the number of nodes on the boundary $\partial\Omega$. By splitting the blocks M and K in (10) according to whether the corresponding degree of freedom is in the interior of Ω or on its boundary we can define a new 3×3 block decomposition. This modified splitting of the coefficient matrix enables us to better exploit the sparsity of the matrix, as demonstrated in the description of a new algorithm below. This modified 3×3 block system takes the form

$$\begin{bmatrix} M_i & M_c^t & K_i^t \\ M_c & M_b & K_c^t \\ K_i & K_c & O \end{bmatrix} \begin{bmatrix} \underline{v}_i \\ \underline{v}_b \\ \underline{u} \end{bmatrix} = - \begin{bmatrix} \underline{0} \\ \underline{0} \\ \underline{f} \end{bmatrix}, \tag{12}$$

where M_i is of size $n_i \times n_i$, M_b is of size $n_b \times n_b$, K_i is of size $n_i \times n_i$ and the interface rectangular blocks, M_c and K_c, have sizes $n_b \times n_i$ and $n_i \times n_b$ respectively. The unknown vector and the right-hand side are partitioned accordingly. Note that for the two-dimensional problems considered in this work $n_i \sim n_b^2$ as $h \to 0$. Furthermore, the use of Dirichlet conditions in (1), (2) and (6) ensure that the right-hand side vector in (12) always takes the form $[\underline{0}^t \;\; \underline{0}^t \;\; \underline{f}^t]^t$. However, for other types of boundary condition (e.g. mixed conditions, as described in [35] for example) the right-hand side can have the more general form $[\underline{d}_i^t \;\; \underline{d}_b^t \;\; \underline{f}^t]^t$ (see also [24]). We shall consider this more general case in the following derivation of a new algorithm despite the fact that for homogeneous Dirichlet problems we know that $\underline{d}_i = \underline{d}_b = \underline{0}$.

The system (12), with the general right hand side $[\underline{d}_i^t \;\; \underline{d}_b^t \;\; \underline{f}^t]^t$, 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

$$\begin{array}{lccl} K_i\underline{v}_i + & K_c\underline{v}_b & = & -\underline{f} \\ M_c\underline{v}_i + & M_b\underline{v}_b & +K_c^t\underline{u} = & \underline{d}_b \\ & (M_c^t - M_iK_i^{-1}K_c)\underline{v}_b & +K_i^t\underline{u} = & \underline{d}_i + M_iK_i^{-1}\underline{f}\,. \end{array} \tag{13}$$

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:

$$B_1 = -K_i^{-1}K_c, \quad \underline{z}_1 = -K_i^{-1}\underline{f}, \tag{14}$$

$$B_2 = -(M_b + M_cB_1)^{-1}K_c^t, \quad \underline{z}_2 = (M_b + M_cB_1)^{-1}(\underline{d}_b - M_c\underline{z}_1) \tag{15}$$

$$\underline{z}_3 = \left(K_i^t + \left(M_c^t + M_iB_1\right)B_2\right)^{-1}\left(\underline{d}_i + M_iK_i^{-1}\underline{f} - \left(M_c^t + M_iB_1\right)\underline{z}_2\right). \tag{16}$$

Note that the unknown vector $\underline{u}$ must now be equal to $\underline{z}_3$. The unknown vectors $\underline{v}_i$ and $\underline{v}_b$ (if required) can also be obtained by a single application of the back-substitution formulas

$$\underline{v}_b = B_2\underline{z}_3 + \underline{z}_2 \quad \text{and} \quad \underline{v}_i = B_1\underline{v}_b + \underline{z}_1. \tag{17}$$

Consequently, the algorithm for solving the linear system (12) can be organized into four macro steps as follows.

Step 1. The equations (14) may be rewritten as

$$K_iB_1 = -K_c, \quad K_i\underline{z}_1 = -\underline{f}. \tag{18}$$

This is really a single system of order n_i with $n_b + 1$ right-hand sides which may be expressed as

$$K_i[B_1 \mid \underline{z}_1] = -[K_c \mid \underline{f}]. \tag{19}$$

The sparse matrix K_i is symmetric and positive definite (it is a discrete Dirichlet Laplacian in fact). Therefore we may use sparse Cholesky factorization of the coefficient matrix K_i (step 1.1) with any ordering of the unknowns (e.g. for minimum fill-in or bandedness, etc.), followed by a forward and back-substitution phase (steps 1.2 and 1.3).

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 (12) remains the same. This enables us to re-use a significant amount of data calculated in the first solve since both the sparse Cholesky factorization of the block K_i and the matrix B_1 remain unchanged. Hence for subsequent calculations this step reduces to a single forward and back-substitution phase to obtain the vector z_1 (step 1.3).

Step 2. We now consider equations (15). These may be rewritten as

$$(M_b + M_c B_1) B_2 = -K_c^t, \quad (M_b + M_c B_1) \underline{z}_2 = \underline{d}_b - M_c \underline{z}_1,$$

or equivalently,

$$(M_b + M_c B_1)[B_2 \mid \underline{z}_2] = -[K_c^t \mid \underline{d}_b - M_c \underline{z}_1]. \tag{20}$$

This is a linear system with $n_i + 1$ right-hand sides and the matrix, of order n_b, is dense and non-symmetric. The conventional approach to solving (20) would therefore be to perform a full LU factorization (with pivoting as necessary) followed by $n_i + 1$ forward and backward substitutions. Since $n_i \gg n_b$ the forward and backward substitution phase is actually more time-consuming than the factorization (since it has complexity $O(n_i n_b^2) > O(n_b^3)$). Given that the matrix K_c^t on the right-hand side of (20) is sparse (containing $O(n_b)$ non-zero entries), an alternative is to compute an explicit inverse of $M_b + M_c B_1$ in $O(n_b^3)$ operations (steps 2.1 and 2.2) followed by multiplication with the sparse augmented matrix $[K_c^t | M_c \underline{z}_1]$, also in $O(n_b^3)$ 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 $\underline{z}_2$ at this step (steps 2.3 and 2.4).

We again observe 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, the inverse of the coefficient matrix in (20) 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 to calculate the vector $\underline{z}_2$ (steps 2.3 and 2.4).

Step 3. Next consider the third block of equations (16), which may be rewritten in the form

$$(K_i^t + (M_c^t + M_i B_1) B_2) \underline{z}_3 = -(-\underline{d}_i + M_i \underline{z}_1 + (M_c^t + M_i B_1) \underline{z}_2). \tag{21}$$

In (21) it is necessary to solve a single system of n_i equations. At first sight this appears to be prohibitively expensive since the coefficient matrix in (21) is both dense and non-symmetric, thus requiring $O(n_i^3)$ operations (and $O(n_i^2)$ memory locations) for a solution. Fortunately however a simple modification of formula (16) 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 (16) we may apply the Sherman-Morrison-Woodbury formula (see [22] 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 $n \gg k$ 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

$$(A + UV^t)^{-1} = A^{-1} - A^{-1} U (I + V^t A^{-1} U)^{-1} V^t A^{-1}. \tag{22}$$

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 $n = n_i$ and $k = n_b$ (and recall that $n_i \gg n_b$). 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

$$U_2 = M_c^t + M_i B_1, \quad \underline{g} = -(-\underline{d}_i + M_i \underline{z}_1 + (M_c^t - M_i K_i^{-1} K_c) \underline{z}_2), \tag{23}$$

it is possible to write equation (16) in the form

$$\underline{z}_3 = (K_i^t + U_2 B_2)^{-1} \underline{g}. \tag{24}$$

Applying formula (22) to (24) then gives

$$\underline{z}_3 = (K_i^t)^{-1}\underline{g} - (K_i^t)^{-1}U_2 \left[I + B_2(K_i^t)^{-1}U_2\right]^{-1} B_2(K_i^t)^{-1}\underline{g}\ , \tag{25}$$

which may in turn be written as $\underline{u} = \underline{z}_3 = \underline{z}_3' - \underline{z}_3''$, where

$$\underline{z}_3' = (K_i^t)^{-1}\underline{g} \quad \text{and} \quad \underline{z}_3'' = (K_i^t)^{-1}U_2 \left[I + B_2(K_i^t)^{-1}U_2\right]^{-1} B_2(K_i^t)^{-1}\underline{g}.$$

Calculation of $\underline{z}_3$ according to the formula (25) may now be organized as follows.

First calculate the vector $\underline{g}$ using (23) (step 3.1) then solve the equation

$$K_i^t \underline{z}_3' = \underline{g} \tag{26}$$

using the Cholesky factorization of K_i^t performed in step 1.1 above (step 3.2). Note that in step 3.1 $\underline{g}$ is calculated using the formula given by (23) rather than explicitly using the matrix B_1 found in step 1.2. This is because, due to the sparsity of the block K_c, it is cheaper to perform a multiplication by K_c followed by a forward and backward substitution (using the known lower and upper triangular factors of K_1) than it is to perform a multiplication by B_1. The same saving is also made at steps 3.10 and 4.2 below.

To calculate the vector $\underline{z}_3''$ the following expression is used:

$$\underline{z}_3'' = (K_i^t)^{-1}U_2 \left[I + B_2(K_i^t)^{-1}U_2\right]^{-1} B_2\underline{z}_3'. \tag{27}$$

The evaluation of z_3'' is then further divided into a number of substeps.

Firstly, to calculate $\underline{x}_1 = B_2\underline{z}_3'$, we use the fact that $B_2 = (M_b + M_c B_1)^{-1}(-K_c^t)$, thus yielding

$$\underline{x}_1 = (M_b + M_c B_1)^{-1}(-K_c^t \underline{z}_3')\ .$$

This calculation of $\underline{x}_1$ (step 3.3) requires a sparse matrix-vector multiplication of complexity $O(n_b)$ followed by a dense matrix-vector multiplication of complexity $O(n_b^2)$. (If at step 2 above we had calculated B_2 explicitly and then formed $\underline{x}_1$ at this step by the dense matrix-vector multiplication $B_2\underline{z}_3'$, this would have required $O(n_i n_b) = O(n_b^3)$ operations.)

Next we calculate $\underline{x}_2 = [I + B_2(K_i^t)^{-1}U_2]^{-1}\,\underline{x}_1$, which may also be broken down into several phases. First, the above equation is rewritten as $[I+B_2(K_i^t)^{-1}U_2]\underline{x}_2 = \underline{x}_1$. We now have to evaluate U_2 from (23) (step 3.4) then solve $K_i^t Y_1 = U_2$ (step 3.5) (this is a linear system with n_b right-hand sides for which the coefficient matrix has already been factorized). Next, it is necessary to perform the matrix multiplication $Y_2 = B_2 Y_1$, and here we again benefit from not having computed B_2 explicitly since it is possible to write $Y_2 = (M_b + M_c B_1)^{-1}(-K_c^t Y_1)$ (step 3.6). This requires a sparse matrix multiplication with a dense matrix, of complexity $O(n_b^2)$, followed by a dense matrix multiplication of complexity $O(n_b^3)$. (Again we note the disadvantage of computing Y_2 by direct multiplication using the dense matrix B_2, which would have a complexity of $O(n_b^2 n_i) = O(n_b^4)$.)

To conclude the calculation of $\underline{x}_2$ we require a matrix addition to form $Y_3 = I + Y_2$ where I is the $n_b \times n_b$ identity matrix (step 3.7). We then solve the dense, non-symmetric system $Y_3\underline{x}_2 = \underline{x}_1$, using an LU factorization (possibly with pivoting for stability) followed by a forward and back-substitution (steps 3.8 and 3.9). 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 $\underline{z}_1$ and $\underline{z}_2$ (which themselves depend upon the right-hand side). Therefore, in the calculations that involve solving multiple systems with the same coefficient matrix, steps 3.4 through to 3.8 need to be performed only once. For subsequent solves all that needs to be done at this sub-step is a single forward and back-substitution to calculate $\underline{x}_2$ from the factorization of Y_3 (step 3.9).

Now, in order to complete the calculation of $\underline{z}_3''$ we must evaluate $\underline{x}_3 = U_2\underline{x}_2$ and then solve $K_i^t\underline{z}_3'' = \underline{x}_3$. Note however that the complexity of performing the dense matrix-vector multiplication $\underline{x}_3 = U_2\underline{x}_2$ is $O(n_i n_b) = O(n_b^3)$. 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

$$\underline{x}_3 = (M_c^t - M_i K_i^{-1} K_c)\underline{x}_2 = M_c^t \underline{x}_2 - M_i K_i^{-1}(K_c \underline{x}_2).$$

Given that the sparse factorization of K_i is already known, this may be computed with a complexity of $O(n_i) = O(n_b^2)$ 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 and these calculations together constitute step 3.10). Solution of the system $K_i^t\underline{z}_3'' = \underline{x}_3$ (step 3.11) requires only a single application of forward and back-substitution since the coefficient matrix has already been factorized.

Having successfully computed both $\underline{z}_3'$ and $\underline{z}_3''$ the final part of step 3 (step 3.12) simply requires a vector subtraction to compute the desired solution $\underline{u} = \underline{z}_3 = \underline{z}_3' - \underline{z}_3''$.

Step 4. This is an optional step which enables the unknown vector $\underline{v}^t = [\underline{v}_i^t \ \underline{v}_b^t]^t$ to be computed if it is required. Straightforward application of the formulas (17) would be of $O(n_i n_b)$ arithmetic complexity. However, this is unnecessarily expensive since we may rewrite (17) in the form

$$\underline{v}_b = -(M_b + M_c B_1)^{-1} K_c^t \underline{z}_3 + \underline{z}_2 \quad \text{and} \quad \underline{v}_i = -K_i^{-1} K_c \underline{v}_b + \underline{z}_1,$$

which allows us to make use of the sparsity of K_c along with our existing factorization of K_i and our representation of $(M_b + M_c B_1)^{-1}$.

The complete algorithm outlined above is presented in Figure 1. 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 $n_i \sim n_b^2$, the cost of a sparse Cholesky factorization of the $n_i \times n_i$ matrix K_i is $O(n_i^f)$ and the cost of a single forward and backward substitution is $O(n_i^s)$. Note that it is assumed that $s \geq 1$ when determining the dominant contributions to steps 3.1 and 3.10 in the figure.

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 $O(n_i^f)$ (step 1.1), $O(n_i^{s+1/2})$ (steps 1.2 and 3.5) or $O(n_i^{3/2})$ (steps 2.2, 3.4 and 3.8). Reference to Table 2 suggests that when SuperLU ([19]) 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 $O(n_i^{1.7})$. For second and subsequent right-hand sides the dominant terms will have a cost of $O(n_i^s) \approx O(n_i^{1.2})$. Inspection of Table 1 therefore 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.

STEP 1

1.1) Factorize $\boldsymbol{K_i = L_K L_K^t}$	$O(n_i^f)$
1.2) Solve $\boldsymbol{L_K L_K^t B_1 = -K_c}$	$O(n_i^{s+1/2})$
1.3) Solve $L_K L_K^t \underline{z}_1 = -\underline{f}$	$O(n_i^s)$

STEP 2

2.1) Form $\boldsymbol{M_b + M_c B_1}$	$O(n_b^2)$
2.2) Invert $\boldsymbol{M_b + M_c B_1}$	$O(n_b^3)$
2.3) Form $\underline{d}_b - M_c \underline{z}_1$	$O(n_b)$
2.4) Form $\underline{z}_2 = (M_b + M_c B_1)^{-1}(\underline{d}_b - M_c \underline{z}_1)$	$O(n_b^2)$

STEP 3

3.1) Form $\underline{g} = \underline{d}_i - M_i \underline{z}_1 - M_c^t \underline{z}_2 + M_i K_i^{-1} K_c \underline{z}_2$	$O(n_i^s)$
3.2) Solve $L_K L_K^t \underline{z}_3' = \underline{g}$	$O(n_i^s)$
3.3) Form $\underline{x}_1 = (M_b + M_c B_1)^{-1}(-K_c^t)\underline{z}_3'$	$O(n_b^2)$
3.4) Form $\boldsymbol{M_c^t + M_i B_1}$	$O(n_i n_b)$
3.5) Solve $\boldsymbol{L_K L_K^t Y_1 = M_c^t + M_i B_1}$	$O(n_i^{s+1/2})$
3.6) Form $\boldsymbol{Y_2 = (M_b + M_c B_1)^{-1}(-K_c^t) Y_1}$	$O(n_b^3)$
3.7) Form $\boldsymbol{Y_3 = I + Y_2}$	$O(n_b)$
3.8) Factorize $\boldsymbol{Y_3 = L_3 U_3}$	$O(n_b^3)$
3.9) Solve $L_3 U_3 \underline{x}_2 = \underline{x}_1$	$O(n_b^2)$
3.10) Form $\underline{x}_3 = (M_c^t - M_i K_i^{-1} K_c)\underline{x}_2$	$O(n_i^s)$
3.11) Solve $L_K L_K^t \underline{z}_3'' = \underline{x}_3$	$O(n_i^s)$
3.12) Form $\underline{u} = \underline{z}_3 = \underline{z}_3' - \underline{z}_3''$	$O(n_i)$

STEP 4

4.1) Form $\underline{v}_b = -(M_b + M_c B_1)^{-1} K_c^t \underline{z}_3 + \underline{z}_2$	$O(n_b^2)$
4.2) Form $\underline{v}_i = -K_i^{-1} K_c \underline{v}_b + \underline{z}_1$	$O(n_i^s)$

Figure 1: Summary of the algorithm for solving (12) along with the asymptotic complexity of each step (note that $n_i \sim n_b^2$). Those steps shown in bold need not be repeated when second or subsequent right-hand sides are solved for.

5 Numerical results

In this section we report on the performance of the proposed algorithm when applied to linear systems of the form (12) 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 (9), 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 ([19]) applied to the full system (12), which is the best alternative direct solver that we have been able to find. All comparisons are presented in terms of both the number of floating point operations taken and the overall execution times required to complete the algorithms (on a single processor of a Sun E4000 computer). The former ensures that the results are neither platform nor implementation dependent, whilst the latter demonstrates that the new method really can lead to faster solution times (on one particular architecture at least). The tests included in this paper are representative examples taken from a more comprehensive survey which may be found in [28].

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 n^{th} local extremal value of the eigenfunction u, t_n is the magnitude of this extremum, and s_n the distance to the n^{th} 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 (for more detailed mesh description and some example grids see [28]).

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	16.56742802
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.

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

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.

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 [28] 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 is 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 (12) directly. All tests were performed on a single processor of a Sun E4000 computer and SuperLU was tested with various column orderings prior to the factorization and for various values of the threshold parameter $u_p \in [0, 1]$ which controls the partial pivoting (for details, see [18]). Upon analysing the impact of initial ordering of the matrix columns on the performance of SuperLU it was apparent that for the original fourth-order problem (12) the MMD algorithm applied to the structure of A^tA (A being the coefficient matrix) together with $u_p = 1$ (classical partial pivoting) gave the best results both in terms of the operation count and execution time. It is these results that are used for the purposes of comparison in Tables 5 through to 8 therefore. In contrast to this, for the second-order problem (the factorization of K_i) it was observed that SuperLU performs best when the MMD preordering algorithm is applied to the structure of $A + A^t$, while, due to the positive definiteness of K_i, the performance was independent of the choice of u_p. For the steps in the new algorithm that do not involve the use of SuperLU vendor-optimized BLAS and LAPACK routines [1] were used where appropriate.

In Tables 5–8 performance results are presented for piecewise cubic approximations on two different sequences of meshes (those used to generate the accuracy results quoted in Tables 3 and 4). Two comparisons of performance are provided: the total number of floating point operations (Tables 5 and 7) and the overall execution time (Tables 6 and 8). In each of these tables the subscript f refers to the first solve of the system (12) and the subscript s refers to subsequent solves. "New" refers to the performance of the algorithm that we propose in Section 4 and "SLU" refers to the performance of SuperLU (with the best ordering strategy) applied directly to (12). Finally, two ratios are defined as

$$\text{Ratio}_f = \frac{\text{New}_f}{\text{SLU}_f} \quad \text{and} \quad \text{Ratio}_s = \frac{\text{New}_s}{\text{SLU}_s}.$$

These ratios allow comparisons to be made between the two solvers. Note also that in each table the grid sizes quoted indicate only the number of intervals on each boundary of the domain but, since the meshes are unstructured, the total number of degrees of freedom, N, probably provides a more reliable indication of the overall problem size.

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 (12) obtained from piecewise cubic C^0 elements on a sequence of non-uniformly refined grids on a quarter of the unit square domain.

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	19.9	38.5	65.5	100.5	150.3	204.7	315.6	468.4	658.7
New_s	0.3	0.7	1.0	1.3	1.7	2.1	2.8	3.8	4.9
SLU_f	27.7	54.3	88.6	137.7	220.3	306.4	501.7	730.7	1085.1
SLU_s	0.5	0.9	1.3	1.7	2.2	2.8	3.9	5.1	6.7
Ratio_f	0.718	0.709	0.739	0.730	0.682	0.668	0.629	0.641	0.607
Ratio_s	0.600	0.778	0.769	0.765	0.773	0.750	0.718	0.745	0.731

Table 6: Computational costs, both absolute (measured in execution time given in CPU seconds on a Sun E4000) and relative (Ratio), of the proposed (New) algorithm and SuperLU (SLU) for the first (f) and subsequent (s) solution of the system (12) obtained from piecewise cubic C^0 elements on a sequence of non-uniformly refined grids on a quarter of the unit square domain.

In terms of both metrics used, the operation count and the cpu time, Tables 5 to 8 clearly demonstrate that the new algorithm is superior to the best alternative for sufficiently fine meshes. For both sequences of grids the results in terms of operation count are more flattering to our new algorithm than the cpu times and this may, at least in part, be due to a lack of architecture-dependent optimization within our implementations. Nevertheless, even in terms of the cpu times taken, there is a significant gain due to using the new algorithm and, equally importantly, this gain increases as the size of the problem grows. It is also worth observing at this point that in each case presented the cost of subsequent solves is almost insignificant in comparison to the cost of the first solve.

We conclude this section with a further note on the stability and accuracy of our new solver. In order to demonstrate that it performs no worse than SuperLU we have applied both algorithms to the same sequence of systems (12) used to generate the data in Tables 3 and 4. Tables 9 and 10 report the results of these computations by presenting the normwise difference in the solutions obtained using these two solvers on (12) for each grid. (In the tables $\mathcal{A}$ represents the matrix

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 7: 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 (12) obtained from piecewise cubic C^0 elements on a sequence of non-uniformly refined grids on an eighth of the unit square domain.

grid	4×28	8×32	16×48	32×96	64×192
N	2086	4804	15052	56035	219136
New_f	0.7	2.5	16.2	126.1	1548.8
New_s	0.0	0.0	0.3	1.6	11.2
SLU_f	0.1	1.2	18.3	193.0	2726.7
SLU_s	0.0	0.0	0.3	2.1	15.1
Ratio_f	7.000	2.083	0.886	0.652	0.568
Ratio_s	—	—	1.000	0.762	0.741

Table 8: Computational costs, both absolute (measured in execution time given in CPU seconds on a Sun E4000) and relative (Ratio), of the proposed (New) algorithm and SuperLU (SLU) for the first (f) and subsequent (s) solution of the system (12) obtained from piecewise cubic C^0 elements on a sequence of non-uniformly refined grids on an eighth of the unit square domain.

in (12), $\underline{b}$ represents the right-hand side vector and $\underline{x}$ the solution vector (with an appropriate subscript to indicate the method of solution).) From these results it is clear that both algorithms have the same level of accuracy and that the maximal discrepancies between the computed solutions are in the area of machine precision. Furthermore, it is interesting to note that computed solution values at nodes close to a corner (where the solution is very small in magnitude) are the same to at least seven significant figures when using the two solvers (e.g. $-0.68461322 \times 10^{-22}$ and $-0.68461323 \times 10^{-22}$ at a particular point on the largest grid).

N	42229	94177	166897
$\|\underline{x}_{\text{NEW}} - \underline{x}_{\text{SLU}}\|_\infty$	6.7E-16	7.2E-16	4.8E-16
$\|\underline{b} - \mathcal{A}\underline{x}_{\text{NEW}}\|_\infty$	5.4E-16	9.1E-16	7.7E-16
$\|\underline{b} - \mathcal{A}\underline{x}_{\text{SLU}}\|_\infty$	6.3E-16	8.7E-16	7.9E-16

Table 9: Normwise discrepancies between the solutions of (12) computed by the new method and SuperLU and corresponding residuals for these systems, obtained from the use of piecewise cubic C^0 finite elements on the sequence of non-uniformly refined grids on a quarter of the unit square domain.

This empirical evidence, along with the robustness of the technique over a large variety of test problems (as reported in [11] and [28]) leads us to believe that our technique is both stable

N	15052	56035	219136
$\|\|\underline{x}_{\mathrm{NEW}} - \underline{x}_{\mathrm{SLU}}\|\|_\infty$	3.9E-16	3.8E-16	4.5E-16
$\|\|\underline{b} - \mathcal{A}\underline{x}_{\mathrm{NEW}}\|\|_\infty$	3.8E-16	1.0E-15	8.2E-16
$\|\|\underline{b} - \mathcal{A}\underline{x}_{\mathrm{SLU}}\|\|_\infty$	4.1E-16	7.9E-16	8.0E-16

Table 10: Normwise discrepancies between the solutions of (12) computed by the new method and SuperLU and corresponding residuals for these systems, obtained from the use of piecewise cubic C^0 finite elements on the sequence of non-uniformly refined grids on an eighth of the unit square domain.

and accurate. Since step 3 of the algorithm, as illustrated in Figure 1, involves the Sherman-Morrison-Woodbury formula it is difficult to be more rigorous than this however. This is because there are few results concerning general conditions which the matrix A in (22) must satisfy to guarantee stabilty. This issue is is addressed in [25] for example but conclusive general results have not so far been forthcoming. In [40] the author performs an analysis of (22) and states the result that the formula is stable if the matrix A is well conditioned. In particular, Yip [40] proves that $k(I + V^tA^{-1}U) \leq k(A)^2$, but this bound can be weak in some cases (recall that in our case $A = K_i$ is a symmetric positive definite matrix and that $k(A) \sim O(h^{-2})$). In the implementation of our algorithm we followed the main recommendations from [40] concerning the scaling of the appropriate blocks U and V and, in practice, computation of $(A + UV^t)^{-1}$ has never failed (due to $I + V^tA^{-1}U$ being singular or nearly singular for example).

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 [22]) 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. 1 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 [20] (or a parallel banded solver could be used), and so a complete parallel algorithm appears to be achievable in an efficient manner.

Acknowledgements

We thank Professor Brian Davies for his initial suggestions concerning the biharmonic eigenvalue problem and many helpful discussions during its investigation, and also Professor Philip Gill for his valuable comments. BMB and MDM would also like to acknowledge the EPRSC for support under grant GR/K84745 and PKJ acknowledges the support of the Leverhulme Trust.

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)(1980), 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,* University of Bergen, Norway, 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: *On the Accurate Finite Element Solution of a Class of Fourth Order Eigenvalue Problems,* Proc. Roy. Soc. (London), to appear.

[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] L.S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R.C. Whaley: *ScaLAPACK Users' Guide.* SIAM, Philadelphia, PA, 1997.

[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, J.R. Gilbert, and X.S. Li: *SuperLU User's Guide.* Technical Report UCB//CSD 97-944, Computer Science Division, University of California, Berkeley, CA, USA, November 1997.

[19] J.W. Demmel, S.C. Eisenstat, J.R. Gilbert, X.S. Li and J.W.H. Liu: *A Supernodal Approach to Sparse Partial Pivoting,* SIAM J. Matrix Anal. Appl., **20**(3)(1999), 720–755.

[20] J.W. Demmel, S.C. Eisenstat, J.R. Gilbert and X.S. Li *An Asynchronous Parallel Supernodal Algorithm for Sparse Gaussian Elimination,* SIAM J. Matrix Anal. Appl., **20**(4)(1999), 915–952.

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

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

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

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

[25] N.J. Higham: *Accuracy and Stability of Numerical Algorithms,* SIAM, Philadelphia, 1996.

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

[27] 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.

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

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

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

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

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

[33] 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.

[34] 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.

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

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

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

[38] 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.

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

[40] E.L. Yip: *A note on the stability of solving a rank-p modification of a linear system by the Sherman–Morrison–Woodbury formula,* SIAM J. Sci. Stat. Comput., **7**(2)(1986), 507–513.

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