SIAM J. MATRIX ANAL. APPL.
Vol. 28, No. 1, pp. 170–189

IMPLICIT-FACTORIZATION PRECONDITIONING AND ITERATIVE SOLVERS FOR REGULARIZED SADDLE-POINT SYSTEMS*

H. SUE DOLLAR†, NICHOLAS I. M. GOULD‡, WIL H. A. SCHILDERS§, AND ANDREW J. WATHEN†

Abstract. We consider conjugate-gradient like methods for solving block symmetric indefinite linear systems that arise from saddle-point problems or, in particular, regularizations thereof. Such methods require preconditioners that preserve certain sub-blocks from the original systems but allow considerable flexibility for the remaining blocks. We construct a number of families of implicit factorizations that are capable of reproducing the required sub-blocks and (some) of the remainder. These generalize known implicit factorizations for the unregularized case. Improved eigenvalue clustering is possible if additionally some of the noncrucial blocks are reproduced. Numerical experiments confirm that these implicit-factorization preconditioners can be very effective in practice.

Key words. regularized saddle-point systems, implicit-factorization preconditioners

AMS subject classifications. 15A23, 65F10, 90C20

DOI. 10.1137/05063427X

1. Introduction.

Given a symmetric n by n matrix H, a symmetric m by m ($m \leq n$) matrix C and a full-rank m ($\leq n$) by n matrix A, we are interested in solving structured linear systems of equations

$$\begin{pmatrix} H & A^T \\ A & -C \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = - \begin{pmatrix} g \\ 0 \end{pmatrix} \tag{1.1}$$

by iterative methods, in which preconditioners of the form

$$M_G = \begin{pmatrix} G & A^T \\ A & -C \end{pmatrix} \tag{1.2}$$

are used to accelerate the iteration for some suitable symmetric G. We denote the coefficient matrix in (1.1) by M_H. There is little loss of generality in assuming the right-hand side of (1.1) has the form given rather than with the more general

$$\begin{pmatrix} H & A^T \\ A & -C \end{pmatrix} \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix}. \tag{1.3}$$

For, so long as we have some mechanism for finding an initial (x_0, y_0) for which $Ax_0 - Cy_0 = c$, linearity of (1.1) implies that $(\bar{x}, \bar{y}) = (x_0 - x, y_0 - y)$ solves (1.3)

*Received by the editors June 22, 2005; accepted for publication (in revised form) by V. Simoncini October 31, 2005; published electronically March 17, 2006.
http://www.siam.org/journals/simax/28-1/63427.html

†Oxford University Computing Laboratory, Numerical Analysis Group, Wolfson Building, Parks Road, Oxford, OX1 3QD, England, UK (Sue.Dollar@comlab.ox.ac.uk, Andy.Wathen@comlab.ox.ac.uk).

‡Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, England, UK (n.i.m.gould@rl.ac.uk). This work was supported by the EPSRC grant GR/S42170.

§Philips Research Laboratories, Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands (wil.schilders@philips.com). Also, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, PO Box 513, 5600 MB Eindhoven, The Netherlands.

when $b = g + Hx_0 + A^T y_0$. In particular, since we intend to use the preconditioner (1.2), solving

$$\begin{pmatrix} G & A^T \\ A & -C \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} 0 \\ c \end{pmatrix} \text{ or } = \begin{pmatrix} b \\ c \end{pmatrix} \tag{1.4}$$

to find suitable (x_0, y_0) are distinct possibilities.

When $C = 0$, (1.2) is commonly known as a constraint preconditioner [35] and in this case systems of the form (1.1) arise as stationarity (KKT) conditions for equality-constrained optimization [40, section 18.1], in mixed finite-element approximation of elliptic problems [6], including, in particular, problems of elasticity [41] and incompressible flow [23], as well as other areas. In practice C is often positive semi-definite (and frequently diagonal)—such systems frequently arise in interior-point and regularization methods in optimization, the simulation of electronic circuits [47] and other related areas; see [3] for an encyclopedic review of (regularized) saddle-point systems. Although such problems may involve m by n A with $m > n$, this is not a restriction, for in this case we might equally solve

$$\begin{pmatrix} C & A \\ A^T & -H \end{pmatrix} \begin{pmatrix} y \\ -x \end{pmatrix} = \begin{pmatrix} 0 \\ g \end{pmatrix},$$

for which A^T has more columns than rows. We place no restrictions on H, although we recognize that in some applications H may be positive (semi-) definite.

Notation. Let I be the (appropriately dimensioned) identity matrix. Given a symmetric matrix M with, respectively, m_+, m_- and m_0 positive, negative and zero eigenvalues, we denote its inertia by $\text{In}(M) = (m_+, m_-, m_0)$.

2. Suitable iterative methods. While it would be perfectly possible to apply general preconditioned iterative methods like GMRES [45] or the symmetric QMR method [25] to (1.1) with the indefinite preconditioner (1.2), the specific form of (1.2) allows the use of the more efficient preconditioned conjugate-gradient (PCG) method [12] instead. Use of GMRES would have the disadvantage that storage and orthogonalization of a set of k vectors would be required at the kth iteration, so that the work per iteration increases at each iteration. The symmetric QMR method does not suffer from this difficulty, though it does not minimize a quantity of interest (such as the residual) unlike GMRES and PCG. Because PCG has such a minimizing property and requires a fixed amount of work per iteration, we shall focus on this approach in this paper. We thus need to derive conditions for which PCG is an appropriate method.

Suppose that C is of rank l, and that we find a decomposition

$$C = EDE^T, \tag{2.1}$$

where E is m by l and D is l by l and invertible—either a spectral decomposition or an LDL^T factorization with pivoting are suitable—but the exact form is not relevant. In this case, on defining additional variables

$$z = -DE^T y,$$

we may rewrite (1.1) as

$$\begin{pmatrix} H & 0 & A^T \\ 0 & D^{-1} & E^T \\ A & E & 0 \end{pmatrix} \begin{pmatrix} x \\ z \\ y \end{pmatrix} = \begin{pmatrix} g \\ 0 \\ 0 \end{pmatrix}. \tag{2.2}$$

Noting the trailing zero block in the coefficient matrix of (2.2), we see that the required (x, z) components of the solution lie in the nullspace of $(A \;\; E)$.

Let the columns of the matrix

$$N = \begin{pmatrix} N_1 \\ N_2 \end{pmatrix}$$

form a basis for this null space. Then

$$\begin{pmatrix} x \\ z \end{pmatrix} = \begin{pmatrix} N_1 \\ N_2 \end{pmatrix} w \tag{2.3}$$

for some w, and (2.2) implies

$$H_N w = N_1^T g, \tag{2.4}$$

where

$$H_N \stackrel{\text{def}}{=} N_1^T H N_1 + N_2^T D^{-1} N_2. \tag{2.5}$$

Since we would like to apply PCG to solve (2.4), our fundamental assumption is then that

A1: the matrix H_N is positive definite.

Fortunately assumption **A1** is often easy to verify.

THEOREM 2.1. *Suppose that the coefficient matrix M_H of (1.1) is nonsingular and has m_{H-} negative eigenvalues and that C has c_- negative ones, then* **A1** *holds if and only if*

$$m_{H-} + c_- = m. \tag{2.6}$$

Proof. It is well known [29, Thm. 2.1] that under assumption **A1** the coefficient matrix E_H of (2.2) has inertia $(n + l, m, 0)$. The result then follows directly from Sylvester's law of inertia, since then $\text{In}(E_H) = \text{In}(D^{-1}) + \text{In}(M_H)$ and D^{-1} has as many negative eigenvalues as C has. ∎

Under assumption **A1**, we may apply the PCG method to find w, and hence recover (x, z) from (2.3). Notice that such an approach does not determine y, and additional calculations may need to be performed to recover it if it is required.

More importantly, it has been shown [8, 11, 31, 43] that rather than computing the iterates explicitly within the nullspace via (2.3), it is possible to perform the iteration in the original (x, z) space so long as the preconditioner is chosen carefully. Specifically, let G be any symmetric matrix for which

A2: the matrix

$$G_N \stackrel{\text{def}}{=} N_1^T G N_1 + N_2^T D^{-1} N_2 \tag{2.7}$$

is positive definite,

which we can check using Theorem 2.1. Then the appropriate projected preconditioned conjugate-gradient (PPCG) algorithm follows [31].

Projected Preconditioned Conjugate Gradients (variant 1):
Given $x = 0$, $z = 0$ and $h = 0$, solve

$$\begin{pmatrix} G & 0 & A^T \\ 0 & D^{-1} & E^T \\ A & E & 0 \end{pmatrix} \begin{pmatrix} r \\ d \\ u \end{pmatrix} = \begin{pmatrix} g \\ h \\ 0 \end{pmatrix}, \tag{2.8}$$

and set $(p, v) = -(r, d)$ and $\sigma = g^T r + h^T d$.
Iterate until convergence:

$$\begin{array}{lrcl} \text{Form } & Hp \text{ and } D^{-1}v. & & \\ \text{Set} & \alpha & = & \sigma/(p^T Hp + v^T D^{-1} v). \\ \text{Update} & x & \leftarrow & x + \alpha p, \\ & z & \leftarrow & z + \alpha v, \\ & g & \leftarrow & g + \alpha Hp \\ \text{and} & h & \leftarrow & h + \alpha D^{-1} v. \\ \text{Given } g \text{ and } h, & \text{solve (2.8) to find } r \text{ and } d. & & \\ \text{Set} & \sigma_{\text{new}} & = & g^T r + h^T d \\ \text{and} & \beta & = & \sigma_{\text{new}}/\sigma. \\ \text{Update} & \sigma & \leftarrow & \sigma_{\text{new}}, \\ & p & \leftarrow & -r + \beta p \\ \text{and} & v & \leftarrow & -d + \beta v. \end{array}$$

We note in passing that the algorithm above may be generalized by replacing D in the preconditioning step (2.8) by any nonsingular T for which $N_1^T G N_1 + N_2^T T^{-1} N_2$ is positive definite. The scalar σ gives an appropriate optimality measure [31], and a realistic termination rule is to stop when σ is small relative to its original value.

While this method is acceptable when a decomposition (2.1) of C is known, it is preferable to be able to work directly with C. To this end, suppose that at each iteration

$$h = -E^T a, \quad v = -DE^T q \text{ and } d = -DE^T t$$

for unknown vectors a, q and t—this is clearly the case at the start of the algorithm. Then, letting $w = Ca$, it is straightforward to show that $t = u + a$, and that we can replace our previous algorithm with the following equivalent one.

Projected Preconditioned Conjugate Gradients (variant 2):
Given $x = 0$, and $a = w = 0$, solve

$$\begin{pmatrix} G & A^T \\ A & -C \end{pmatrix} \begin{pmatrix} r \\ u \end{pmatrix} = \begin{pmatrix} g \\ w \end{pmatrix}, \tag{2.9}$$

and set $p = -r$, $q = -u$ and $\sigma = g^T r$.
Iterate until convergence:

Form Hp and Cq.
Set $\alpha = \sigma/(p^T Hp + q^T Cq)$.
Update $x \leftarrow x + \alpha p$,
$a \leftarrow a + \alpha q$,
$g \leftarrow g + \alpha Hp$
and $w \leftarrow w + \alpha Cq$.
Given g and w, solve (2.9) to find r and u.
Set $t = a + u$,
$\sigma_{\text{new}} = g^T r + t^T w$
and $\beta = \sigma_{\text{new}}/\sigma$.
Update $\sigma \leftarrow \sigma_{\text{new}}$,
$p \leftarrow -r + \beta p$
and $q \leftarrow -t + \beta q$.

Notice now that z no longer appears, and that the preconditioning is carried out using the matrix M_G mentioned in the introduction. Also note that although this variant involves two more vectors than its predecessor, t is simply used as temporary storage and may be omitted if necessary, while w may also be replaced by Ca if storage is tight.

When $C = 0$, this is essentially the algorithm given by [31], but for this case the updates for v and w are unnecessary and may be discarded. At the other extreme, when C is nonsingular the algorithm is precisely that proposed by [30, Alg. 2.3], and is equivalent to applying PCG to the system

$$(H + A^T C^{-1} A)x = g,$$

using a preconditioner of the form $G + A^T C^{-1} A$.

Which of the two variants is preferable depends on whether we have a decomposition (2.1) and whether l is small relative to m: the vectors h and v in the first variant are of length l, while the corresponding a and q in the second are of length m. Notice also that although the preconditioning steps in the first variant require that we solve (2.8) this is entirely equivalent to solving (2.9), where $w = -EDh$, and recovering

$$d = D(h - E^T v).$$

Thus our remaining task is to consider how to build suitable and effective preconditioners of the form (1.2). We recall that it is the distribution of the generalized eigenvalues λ for which

$$H_N \bar{v} = \lambda G_N \bar{v} \tag{2.10}$$

that determines the convergence of the preceding PPCG algorithms, and thus we will be particularly interested in preconditioners which cluster these eigenvalues. In particular, if we can efficiently compute G_N so that there are few distinct eigenvalues λ in (2.10), then PPCG convergence (termination) will be rapid.

3. Eigenvalue considerations. We first consider the spectral implications of preconditioning (1.1) by (1.2).

THEOREM 3.1 (see [16, Thm. 3.1] or, in special circumstances, [4, 44]). *Suppose that M_H is the coefficient matrix of* (1.1). *Then $M_G^{-1} M_H$ has m unit eigenvalues, and the remaining n eigenvalues satisfy*

$$(H - \lambda G)v = (\lambda - 1)A^T w, \quad \textit{where } Av - Cw = 0.$$

If C is invertible, the nonunit eigenvalues satisfy

$$(H + A^T C^{-1} A)v = \lambda(G + A^T C^{-1} A)v. \tag{3.1}$$

Our goal in this section is to improve upon this result in the general case $C \neq 0$. For the special case in which $C = 0$, results are already known [18] concerning the eigenvalues of $K_G^{-1} K_H$ for the pair of matrices

$$K_H = \begin{pmatrix} H & A^T \\ A & 0 \end{pmatrix} \text{ and } K_G = \begin{pmatrix} G & A^T \\ A & 0 \end{pmatrix}.$$

These results refer to a partitioning of A as

$$A = (A_1 \;\; A_2), \tag{3.2}$$

so that its leading m by m submatrix

A3: A_1 is nonsingular;

and a similar partitioning of G and H as

$$G = \begin{pmatrix} G_{11} & G_{21}^T \\ G_{21} & G_{22} \end{pmatrix} \text{ and } H = \begin{pmatrix} H_{11} & H_{21}^T \\ H_{21} & H_{22} \end{pmatrix}, \tag{3.3}$$

where G_{11} and H_{11} are, respectively, the leading m by m submatrices of G and H.

In practice, the partitioning of A to ensure **A3** may involve column permutations, but without loss of generality we simply assume here that any required permutations have already been carried out. Given **A3**, we shall be particularly concerned with the *reduced-space* basis matrix

$$N = \begin{pmatrix} R \\ I \end{pmatrix}, \text{ where } R = -A_1^{-1} A_2. \tag{3.4}$$

Such basis matrices play vital roles in simplex (pivoting)-type methods for linear programming [2, 24], and more generally in active-set methods for nonlinear optimization [27, 38, 39].

THEOREM 3.2 (see [18, Thm. 2.3]). *Suppose that G and H are as in* (3.3)*, that* **A3** *holds and that*

$$G_{22} = H_{22}, \;\; \textit{but } G_{11} = 0 \;\; \textit{and } G_{21} = 0. \tag{3.5}$$

Suppose furthermore that H_{22} is positive definite, and let

$$\begin{aligned} \rho \stackrel{\text{def}}{=} \; & \min\left[\operatorname{rank}(A_2), \operatorname{rank}(H_{21})\right] \\ & + \; \min\left[\operatorname{rank}(A_2), \operatorname{rank}(H_{21}) + \min[\operatorname{rank}(A_2), \operatorname{rank}(H_{11})]\right]. \end{aligned}$$

Then $K_G^{-1} K_H$ has at most

$$\operatorname{rank}(R^T H_{21}^T + H_{21} R + R^T H_{11} R) + 1 \leq \min(\rho, n - m) + 1 \leq \min(2m, n - m) + 1$$

distinct eigenvalues.

The restriction that H_{22} be positive definite is not as severe as it might first seem because the problem may be reformulated to use $H_{22} + A_2^T \Delta A_2$ for any symmetric

positive definite weight matrix Δ instead [18, Thm. 2.2]—this corresponds to the so-called augmented Lagrangian approach [33].

THEOREM 3.3 (see [18, Thm. 2.4]). *Suppose that G and H are as in* (3.3)*, that* **A3** *holds and that*

$$G_{22} = H_{22} \text{ and } G_{11} = H_{11} \text{ but } G_{21} = 0. \tag{3.6}$$

Suppose furthermore that $H_{22} + R^T H_{11}^T R$ is positive definite, and that

$$\nu \stackrel{\text{def}}{=} 2\min\left[\operatorname{rank}(A_2), \operatorname{rank}(H_{21})\right].$$

Then $K_G^{-1} K_H$ has at most

$$\operatorname{rank}(R^T H_{21}^T + H_{21} R) + 1 \leq \nu + 1 \leq \min(2m, n-m) + 1$$

distinct eigenvalues.

THEOREM 3.4 (see [18, Thm. 2.5]). *Suppose that G and H are as in* (3.3)*, that* **A3** *holds and that*

$$G_{22} = H_{22} \text{ and } G_{21} = H_{21} \text{ but } G_{11} = 0. \tag{3.7}$$

Suppose furthermore that $H_{22} + R^T H_{21}^T + H_{21} R$ is positive definite, and that

$$\mu \stackrel{\text{def}}{=} \min\left[\operatorname{rank}(A_2), \operatorname{rank}(H_{11})\right].$$

Then $K_G^{-1} K_H$ has at most

$$\operatorname{rank}(R^T H_{11} R) + 1 \leq \mu + 1 \leq \min(m, n-m) + 1$$

distinct eigenvalues.

Turning to the general case of $C \neq 0$, denote the coefficient matrices of the systems (2.2) and (2.8) by

$$\bar{K}_H \stackrel{\text{def}}{=} \begin{pmatrix} H & 0 & A^T \\ 0 & D^{-1} & E^T \\ A & E & 0 \end{pmatrix} \text{ and } \bar{K}_G \stackrel{\text{def}}{=} \begin{pmatrix} G & 0 & A^T \\ 0 & D^{-1} & E^T \\ A & E & 0 \end{pmatrix},$$

respectively. Recalling the definitions (2.5) and (2.7) of H_N and G_N, the following result is a direct consequence of [35, Thm. 2.1].

THEOREM 3.5. *Suppose that N is any (n by $n+l-m$) basis matrix for the null space of $(A \;\; E)$. Then $\bar{K}_G^{-1} \bar{K}_H$ has $2m$ unit eigenvalues, and the remaining $n+l-m$ eigenvalues are those of the generalized eigenproblem* (2.10).

We may improve on Theorem 3.5 by applying Theorems 3.2–3.4 in our more general setting. To do so, let

$$\bar{R} = -A_1^{-1}(A_2 \;\; E),$$

and note that (3.2) implies the partitioning

$$\bar{K}_H = \left(\begin{array}{c|ccc} H_{11} & H_{21}^T & 0 & A_1^T \\ \hline H_{21} & H_{22}^T & 0 & A_2^T \\ 0 & 0 & D^{-1} & E^T \\ \hline A_1 & A_2 & E & 0 \end{array}\right) \text{ and } \bar{K}_G = \left(\begin{array}{c|ccc} G_{11} & G_{21}^T & 0 & A_1^T \\ \hline G_{21} & G_{22}^T & 0 & A_2^T \\ 0 & 0 & D^{-1} & E^T \\ \hline A_1 & A_2 & E & 0 \end{array}\right).$$

We then have the following immediate consequences.

COROLLARY 3.6. *Suppose that G and H are as in* (3.3) *and that* (3.5) *and* **A3** *hold. Suppose furthermore that*

$$\begin{pmatrix} H_{22} & 0 \\ 0 & D^{-1} \end{pmatrix} \tag{3.8}$$

is positive definite, and let

$$\bar{\rho} = \min\left[\eta, \operatorname{rank}(H_{21})\right] + \min\left[\eta, \operatorname{rank}(H_{21}) + \min[\eta, \operatorname{rank}(H_{11})]\right],$$

where $\eta = \operatorname{rank}(A_2 \;\; E)$. *Then* $\bar{K}_G^{-1}\bar{K}_H$ *has at most*

$$\operatorname{rank}(\bar{R}^T H_{21}^T + H_{21}\bar{R} + \bar{R}^T H_{11}\bar{R}) + 1 \leq \min(\bar{\rho}, n + l - m) + 1 \leq \min(2m, n + l - m) + 1$$

distinct eigenvalues.

COROLLARY 3.7. *Suppose that G and H are as in* (3.3) *and that* (3.6) *and* **A3** *hold. Suppose furthermore that*

$$\begin{pmatrix} H_{22} & 0 \\ 0 & D^{-1} \end{pmatrix} + \bar{R}^T H_{11}^T \bar{R} \tag{3.9}$$

is positive definite, and that

$$\bar{\nu} = 2\min\left[\eta, \operatorname{rank}(H_{21})\right],$$

where $\eta = \operatorname{rank}(A_2 \;\; E)$. *Then* $\bar{K}_G^{-1}\bar{K}_H$ *has at most*

$$\operatorname{rank}(\bar{R}^T H_{21}^T + H_{21}\bar{R}) + 1 \leq \bar{\nu} + 1 \leq \min(2m, n + l - m) + 1$$

distinct eigenvalues.

COROLLARY 3.8. *Suppose that G and H are as in* (3.3) *and that* (3.7) *and* **A3** *hold. Suppose furthermore that*

$$\begin{pmatrix} H_{22} & 0 \\ 0 & D^{-1} \end{pmatrix} + \bar{R}^T H_{21}^T + H_{21}\bar{R} \tag{3.10}$$

is positive definite, and that

$$\bar{\mu} = \min\left[\eta, \operatorname{rank}(H_{11})\right],$$

where $\eta = \operatorname{rank}(A_2 \;\; E)$. *Then* $\bar{K}_G^{-1}\bar{K}_H$ *has at most*

$$\operatorname{rank}(\bar{R}^T H_{11}\bar{R}) + 1 \leq \bar{\mu} + 1 \leq \min(m, n + l - m) + 1$$

distinct eigenvalues.

While the requirements that (3.8)–(3.10) be positive definite may at first seem strong assumptions, as in the case $C = 0$ we can also apply a so-called augmented Lagrangian approach for the general case $C \neq 0$.

THEOREM 3.9. *The inertial requirement* (2.6) *holds for a given H if and only if there exists a positive semi-definite matrix $\bar{\Delta}$ such that*

$$\begin{pmatrix} H & 0 \\ 0 & D^{-1} \end{pmatrix} + \begin{pmatrix} A^T \\ E^T \end{pmatrix} \Delta (A \;\; E)$$

is positive definite for all Δ *for which* $\Delta - \bar{\Delta}$ *is positive semi-definite. In particular, if* (2.6) *holds,* $H + A^T\Delta A$ *and* $E^T\Delta E + D^{-1}$ *are positive definite for all such* Δ.

Proof. This follows immediately by applying [18, Thm. 2.2] to $\bar{K}_H$. ∎

Because $Ax + Ez = 0$ we may rewrite (2.2) as the equivalent system

$$\begin{pmatrix} H + A^T\Delta A & A^T\Delta E & A^T \\ E^T\Delta A & E^T\Delta E + D^{-1} & E^T \\ A & E & 0 \end{pmatrix} \begin{pmatrix} x \\ z \\ y \end{pmatrix} = \begin{pmatrix} g \\ 0 \\ 0 \end{pmatrix}.$$

Eliminating the variable z, we find that

$$\begin{pmatrix} H + A^T\Delta A & A^T P^T \\ PA & -W \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = - \begin{pmatrix} g \\ 0 \end{pmatrix},$$

where

$$P = I - \Delta W \ \text{ and } \ W = E(E^T\Delta E + D^{-1})^{-1}E^T.$$

Hence

$$\begin{pmatrix} H + A^T\Delta A & A^T \\ A & -\bar{C} \end{pmatrix} \begin{pmatrix} x \\ \bar{y} \end{pmatrix} = - \begin{pmatrix} g \\ 0 \end{pmatrix}, \tag{3.11}$$

where

$$\bar{C} = P^{-1}WP^{-T} = (I - \Delta W)^{-1}W(I - W\Delta)^{-1} \ \text{ and } \ \bar{y} = P^T y. \tag{3.12}$$

Thus it follows from Theorem 3.9 that we may rewrite (2.2) so that its trailing and leading diagonal blocks are, respectively, negative semi- and positive definite. In doing so, any underlying structure (such as sparsity) may be compromised. For the sparse case, if we are prepared to tolerate fill-in in these blocks, requirements (3.8)–(3.10) then seem more reasonable.

Although (3.12) may appear complicated for general C, $\bar{C}$ is diagonal whenever C is. More generally, if $E = I$, $\bar{C} = D + D\Delta D$ and we may recover $y = (I + \Delta D)\bar{y}$.

4. Suitable preconditioners. It has long been common practice (at least in optimization circles) [4, 7, 13, 26, 36, 49] to use *explicit-factorization* preconditioners of the form (1.2) by specifying G and factorizing M_G using a suitable symmetric, indefinite package such as `MA27` [21] or `MA57` [20]. Given G, an alternative used commonly by the PDE community (see, for example, [1, 22, 36, 42, 44, 14, 48] and the many references in [3]) is to use the explicit block decomposition

$$M_G = \begin{pmatrix} I & 0 \\ AG^{-1} & I \end{pmatrix} \begin{pmatrix} G & 0 \\ 0 & -C - AG^{-1}A^T \end{pmatrix} \begin{pmatrix} I & G^{-1}A^T \\ 0 & I \end{pmatrix} \tag{4.1}$$

to solve (1.2) via factorizations of G and the Schur complement $C + AG^{-1}A^T$ (or an approximation to the latter) if these are viable. While such techniques for choosing G have often been successful, they have usually been rather ad hoc, with little attempt to improve upon the eigenvalue distributions beyond those suggested by Theorem 3.1. In this section we investigate an implicit-factorization alternative.

4.1. Implicit-factorization preconditioners. Recently Dollar and Wathen [19] proposed a class of incomplete factorizations for saddle-point problems ($C = 0$), based upon earlier work by Schilders [46]. They consider preconditioners of the form

$$M_G = PBP^T, \tag{4.2}$$

where solutions with each of the matrices P, B and P^T are easily obtained. In particular, rather than obtaining P and B from a given M_G, M_G *is derived from specially chosen* P *and* B. In this section, we examine a broad class of methods of this form.

In order for the methods we propose to be effective, we shall require that **A3** holds. Since there is considerable flexibility in choosing the "basis" A_1 from the rectangular matrix A by suitable column interchanges, **A3** is often easily, and sometimes trivially, satisfied. Even though, theoretically, there is a lot of choice, the actual A_1 that is used for practical computation can have a significant effect on the overall effectiveness of the preconditioning strategies described in this paper. The problem of determining the "sparsest" A_1 is NP hard, [9, 10], while numerical considerations must be given to ensure that A_1 is not badly conditioned if at all possible [27]. More generally, we do not necessarily assume that A_1 is sparse or structured nor that it has a sparse (or other) factorization, merely that there are effective ways to solve systems involving A_1 and A_1^T. For example, for many problems involving constraints arising from the discretization of partial differential equations, there are highly effective *iterative* methods for such systems [5].

Suppose that

$$P = \begin{pmatrix} P_{11} & P_{12} & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & P_{32} & P_{33} \end{pmatrix} \text{ and } B = \begin{pmatrix} B_{11} & B_{21}^T & B_{31}^T \\ B_{21} & B_{22} & B_{32}^T \\ B_{31} & B_{32} & B_{33} \end{pmatrix}. \tag{4.3}$$

Our goal is to ensure that

$$(M_G)_{31} = A_1, \tag{4.4a}$$

$$(M_G)_{32} = A_2 \tag{4.4b}$$

$$\text{and} \quad (M_G)_{33} = -C, \tag{4.4c}$$

whenever $M_G = PBP^T$. Pragmatically, though, we are only interested in the case where one of the three possibilities

$$P_{11} = 0, \ P_{12} = 0 \text{ and } P_{32} = 0, \tag{4.5a}$$

$$\text{or } P_{11} = 0, \ P_{12} = 0 \text{ and } P_{21} = 0, \tag{4.5b}$$

$$\text{or } P_{12} = 0, \ P_{32} = 0 \text{ and } P_{33} = 0 \tag{4.5c}$$

(as well as nonsingular P_{31} and P_{22}) hold, since only then will P be easily block-invertible. Likewise, we restrict ourselves to the three general cases

(4.6a) $B_{21} = 0, \ B_{31} = 0$ and $B_{32} = 0$ with easily invertible B_{11}, B_{22} and B_{33},

(4.6b) $B_{32} = 0$ and $B_{33} = 0$ with easily invertible B_{31} and B_{22}, or

(4.6c) $B_{11} = 0$ and $B_{21} = 0$ with easily invertible B_{31} and B_{22},

so that B is block-invertible. B is also easily block-invertible if

$$B_{21} = 0 \text{ and } B_{32} = 0 \text{ with easily invertible } \begin{pmatrix} B_{11} & B_{31}^T \\ B_{31} & B_{33} \end{pmatrix} \text{ and } B_{22}, \tag{4.7}$$

and we will also consider this possibility.

TABLE 4.1

Possible implicit factors for the preconditioner (1.2). *We give the P and B factors and any necessary restrictions on their entries. We also associate a family number with each class of implicit factors. Full derivations are given in* [17, Appendix A].

Family/ reference	P	B	Conditions
1.	$\begin{pmatrix} 0 & 0 & A_1^T \\ 0 & P_{22} & A_2^T \\ P_{31} & 0 & P_{33} \end{pmatrix}$	$\begin{pmatrix} B_{11} & 0 & 0 \\ 0 & B_{22} & 0 \\ 0 & 0 & B_{33} \end{pmatrix}$	$B_{11} = -P_{31}^{-1}(C + P_{33})P_{31}^{-T}$ $B_{33} = P_{33}^{-1}$
2.	$\begin{pmatrix} 0 & 0 & A_1^T \\ 0 & P_{22} & A_2^T \\ P_{31} & 0 & P_{33} \end{pmatrix}$	$\begin{pmatrix} B_{11} & 0 & B_{31}^T \\ 0 & B_{22} & 0 \\ B_{31} & 0 & 0 \end{pmatrix}$	$P_{31} = B_{31}^{-T}$ $P_{33} + P_{33}^T + P_{31}B_{11}P_{31}^T = -C$
3.	$\begin{pmatrix} 0 & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & -C \end{pmatrix}$	$\begin{pmatrix} B_{11} & 0 & B_{31}^T \\ 0 & B_{22} & 0 \\ B_{31} & 0 & 0 \end{pmatrix}$	$B_{31} = P_{31}^{-T}$ $B_{11} = P_{31}^{-1}CP_{31}^{-T}$
4.	$\begin{pmatrix} 0 & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & P_{33} \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & B_{31}^T \\ 0 & B_{22} & B_{32}^T \\ B_{31} & B_{32} & 0 \end{pmatrix}$	$P_{21} = -P_{22}B_{32}^TB_{31}^{-T}$ $P_{31} = B_{31}^{-T}$ $P_{33} + P_{33}^T = -C$
5.	$\begin{pmatrix} 0 & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & P_{33} \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & B_{31}^T \\ 0 & B_{22} & B_{32}^T \\ B_{31} & B_{32} & B_{33} \end{pmatrix}$	$-C = P_{33} + P_{33}^T - P_{33}B_{33}P_{33}^T$ $B_{31} = (I - B_{33}P_{33}^T)P_{31}^{-T}$ $B_{32} = -B_{31}P_{21}^TP_{22}^{-T}$
6.	$\begin{pmatrix} 0 & 0 & A_1^T \\ 0 & P_{22} & A_2^T \\ P_{31} & P_{32} & P_{33} \end{pmatrix}$	$\begin{pmatrix} B_{11} & B_{21}^T & B_{31}^T \\ B_{21} & B_{22} & 0 \\ B_{31} & 0 & 0 \end{pmatrix}$	$P_{31} = B_{31}^{-T}$ $P_{32} = -P_{31}B_{21}^TB_{22}^{-1}$ $P_{33} + P_{33}^T$ $= -C - P_{31}(B_{11} - B_{21}^TB_{22}^{-1}B_{21})P_{31}^T$
7.	$\begin{pmatrix} 0 & 0 & A_1^T \\ 0 & P_{22} & A_2^T \\ P_{31} & P_{32} & P_{33} \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & B_{31}^T \\ 0 & B_{22} & B_{32}^T \\ B_{31} & B_{32} & B_{33} \end{pmatrix}$	$P_{33} + P_{33}^T + P_{33}(B_{33} - B_{32}B_{22}^{-1}B_{32}^T)P_{33}^T$ $= -C$ $P_{32} = -P_{33}B_{32}B_{22}^{-1}$ $P_{31} = (I - P_{32}B_{32}^T - P_{33}B_{33}^T)B_{31}^{-T}$

We consider all of these possibilities in detail in [17, Appendix A], and summarize our findings in Tables 4.1 and 4.2. We have identified eleven possible classes of easily invertible factors that are capable of reproducing the A and C blocks of M_G, a further two which may be useful when C is diagonal, and one that is only applicable if $C = 0$.

Notice that aside from invertibility, there are *never* restrictions on P_{22} and B_{22}.

4.2. Reproducing H. Having described families of preconditioners which are capable of reproducing the required components A and C of M_G, we now examine what form the resulting G takes. In particular, we consider which submatrices of G can be defined to completely reproduce the associated submatrix of H; we say that a

TABLE 4.2
Possible implicit factors for the preconditioner (1.2) (*cont.*). *We give the P and B factors and any necessary restrictions on their entries. We also associate a family number with each class of implicit factors. Full derivations are given in* [17, Appendix A].

Family/ reference	P	B	Conditions
8.	$\begin{pmatrix} A_1^T & 0 & A_1^T \\ A_2^T & P_{22} & A_2^T \\ -C & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} -C^{-1} & 0 & 0 \\ 0 & B_{22} & 0 \\ 0 & 0 & B_{33} \end{pmatrix}$	C invertible
9.	$\begin{pmatrix} P_{11} & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} B_{11} & B_{21}^T & B_{31}^T \\ B_{21} & B_{22} & 0 \\ B_{31} & 0 & 0 \end{pmatrix}$	$B_{11} = -P_{31}^{-1} C P_{31}^{-T}$ $B_{31} = P_{31}^{-T} - M B_{11}$ $B_{21} = P_{22}^{-1}(P_{21} - A_2^T M) B_{11}$ $P_{11} = A_1^T M$ for some invertible M
10.	$\begin{pmatrix} P_{11} & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & B_{31}^T \\ 0 & B_{22} & B_{32}^T \\ B_{31} & B_{32} & B_{33} \end{pmatrix}$	$C = 0$ $P_{31} = B_{31}^{-T}$
11.	$\begin{pmatrix} 0 & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & -C \end{pmatrix}$	$\begin{pmatrix} B_{11} & 0 & B_{31}^T \\ 0 & B_{22} & 0 \\ B_{31} & 0 & B_{33} \end{pmatrix}$	C invertible $P_{31}^T = B_{11}^{-1} B_{31}^T C$ $B_{33} = (B_{31} P_{31}^T - I) C^{-1}$
12.	$\begin{pmatrix} 0 & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & -C \end{pmatrix}$	$\begin{pmatrix} B_{11} & 0 & B_{31}^T \\ 0 & B_{22} & 0 \\ B_{31} & 0 & B_{33} \end{pmatrix}$	$B_{11} = P_{31}^{-1} C P_{31}^{-T}$ $B_{31} = P_{31}^{-T}$, where $B_{33} C = 0$
13.	$\begin{pmatrix} 0 & 0 & A_1^T \\ 0 & P_{22} & A_2^T \\ P_{31} & 0 & P_{33} \end{pmatrix}$	$\begin{pmatrix} B_{11} & 0 & B_{31}^T \\ 0 & B_{22} & 0 \\ B_{31} & 0 & B_{33} \end{pmatrix}$	$P_{31} = (I - P_{33} B_{33}) B_{31}^{-T}$ $B_{11} = P_{31}^{-1} \left(P_{33} B_{33} P_{33}^T - C - P_{33} - P_{33}^T \right) P_{31}^{-T}$
14.	$\begin{pmatrix} P_{11} & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} B_{11} & 0 & B_{31}^T \\ 0 & B_{22} & 0 \\ B_{31} & 0 & B_{33} \end{pmatrix}$	$B_{11} = -P_{31}^{-1} C P_{31}^{-T}$ $B_{31} = P_{31}^{-T} - M B_{11}$ $P_{11} = A_1^T M$ $P_{21} = A_2^T M$ for some invertible M

component G_{ij}, $i, j \in \{1, 2\}$, is *complete* if it is possible to choose it so that $G_{ij} = H_{ij}$. We give the details in [17, Appendix B], and summarize our findings for each of the 14 families from section 4.1 in Table 4.3.

Some of the submatrices in the factors P and B can be arbitrarily chosen without changing the completeness of the family. We shall call these "free blocks." For example, consider family 2 from Table 4.1. The matrix G produced by this family always satisfies $G_{11} = 0$, $G_{21} = 0$, and $G_{22} = P_{22} B_{22} P_{22}^T$. Hence, P_{22} can be defined as any nonsingular matrix of suitable dimension, and B_{22}^T can be subsequently chosen so that $G_{22} = H_{22}$. The simplest choice for P_{22} is the identity matrix. We observe that

TABLE 4.3

Blocks of G for the families of preconditioners given in Tables 4.1 and 4.2. The superscript 1 indicates that the value of G_{21} is dependent on the choice of G_{11}. If G_{ij}, $i,j \in \{1,2\}$, is a zero matrix, then a superscript 2 is used. The superscript 3 means that G_{21} is dependent on the choice of G_{11} when $C = 0$, but complete otherwise, while the superscript 4 indicates that G_{11} is only guaranteed to be complete when $C = 0$.

Family	Completeness G_{11}	G_{21}	G_{22}	Conditions on C	Feasible to use	Comments
1.	✓	$\times^1$	✓	any C	✓	
2.	$\times^2$	$\times^2$	✓	any C	✓	
3.	$\times^2$	✓	✓	any C	✓	
4.	$\times^2$	$\times^2$	✓	any C	✓	Simplest choice of free blocks is the same as that for family 2.
5.	✓	$\times^1$	✓	any C	$C = 0$	
6.	$\times^2$	$\times^2$	✓	any C	✓	Simplest choice of free blocks is the same as that for family 2.
7.	✓	$\checkmark^3$	✓	any C	$C = 0$	If $C = 0$ using simplest choice of free blocks, then same as that for family 5 with $C = 0$.
8.	✓	$\times^1$	✓	nonsingular	✓	
9.	✓	✓	✓	any C	$C = 0$	
10.	✓	✓	✓	$C = 0$	✓	Generalization of factorization suggested by Schilders [19, 46]; See also [37].
11.	✓	✓	✓	nonsingular	✓	
12.	$\checkmark^4$	✓	✓	any C	diagonal C	$C = 0$ gives example of family 10. C nonsingular gives family 3.
13.	✓	$\times^1$	✓	any C	✓	
14.	✓	$\times^1$	✓	any C	✓	$C = 0$ gives example of family 10.

the choice of the remaining submatrices in P and B will not affect the completeness of the factorization, and are only required to satisfy the conditions given in Table 4.1. The simplest choices for these submatrices will be $P_{31} = I$, and $B_{11} = 0$, giving $P_{33} = -\frac{1}{2}C$, and $B_{31} = I$. Using these simple choices we obtain:

$$P = \begin{pmatrix} 0 & 0 & A_1^T \\ 0 & I & A_2^T \\ I & 0 & -\frac{1}{2}C \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 0 & 0 & I \\ 0 & B_{22} & 0 \\ I & 0 & 0 \end{pmatrix}.$$

The simplest choice of the free blocks may result in some of the families having the same factors as other families. This is indicated in the "comments" column of the table. Table 4.3 also gives the conditions that C must satisfy to use the family, and whether the family is feasible to use, i.e., are the conditions on the blocks given in Tables 4.1 and 4.2 easily satisfied?

Table 4.4 gives some guidance towards which families from Tables 4.1 and 4.2 should be used in the various cases of G given in section 3. We also suggest simple choices for the free blocks. In our view, although Table 4.3 indicates that it is theoret-

TABLE 4.4
Guidance towards which family to use to generate the various choices of G given in section 3.

Sub-blocks of G	Conditions on C	Family	Free block choices
$G_{22} = H_{22}$, $G_{11} = 0$, $G_{21} = 0$	any C	2	$P_{22} = I$, $P_{31} = I$, $B_{11} = 0$
$G_{22} = H_{22}$, $G_{11} = H_{11}$, $G_{21} = 0$	$C = 0$	10	$B_{21} = 0$, $P_{22} = I$, $P_{31} = I$
$G_{22} = H_{22}$, $G_{11} = H_{11}$, $G_{21} = 0$	C nonsingular	11	$P_{22} = I$, $P_{31} = I$
$G_{22} = H_{22}$, $G_{21} = H_{21}$, $G_{11} = 0$	any C	3	$P_{22} = I$, $P_{31} = I$

ically possible to reproduce all of H using, e.g., family 9, in practice this is unviable because structure, such as sparsity, could be severely compromised.

5. Numerical examples. In this section we examine how effective implicit-factorization preconditioners might be when compared with explicit-factorization ones. We consider problems generated using the complete set of quadratic programming examples from the CUTEr [32] test set used in our previous experiments for the $C = 0$ case [18]. All inequality constraints are converted to equations by adding slack variables, and a suitable "barrier" penalty term is added to the diagonal of the Hessian for each bounded or slack variable to simulate systems that might arise during an iteration of an interior-point method for such problems; in each of the test problems the value 1.1 is used—this sort of value would correspond to an intermediate stage of the outer (optimization) iteration. The resulting equality-constrained quadratic programs are then of the form

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad g^T x + \tfrac{1}{2} x^T H x \ \text{ subject to } \ Ax = 0. \tag{5.1}$$

Given this data H and A, two illustrative choices of diagonal C are considered, namely,

$$c_{ii} = 1 \ \text{ for } \ 1 \le i \le m, \tag{5.2}$$

and

$$c_{ii} = \begin{cases} 0 & \text{for } \ 1 \le i \le \lceil \frac{m}{2} \rceil \\ 1 & \text{for } \ \lceil \frac{m}{2} \rceil + 1 \le i \le m; \end{cases} \tag{5.3}$$

in practice such C may be thought of as regularization terms for some or all on the constraints in (5.1). Our aim is thus to solve for the primal variables x in the system (1.1) using a suitably preconditioned PPCG iteration.

Rather than present large tables of data (these can be found in [17, Appendix C]), here we use performance profiles [15] to illustrate our results. To explain the idea, let $\mathcal{P}$ represent the set of preconditioners that we wish to compare. Suppose that the run of PPCG using a given preconditioner $i \in \mathcal{P}$ reports the total CPU time $t_{ij} \ge 0$ when executed on example j from the test set $\mathcal{T}$. For all problems $j \in \mathcal{T}$, we want to compare the performance of algorithm i with the performance of the fastest algorithm in the set $\mathcal{P}$. For $j \in \mathcal{T}$, let $t_j^{\text{MIN}} = \min\{t_{ij}; i \in \mathcal{P}\}$. Then for $\alpha \ge 1$ and each $i \in \mathcal{P}$ we define

$$k(t_{ij}, t_j^{\text{MIN}}, \alpha) = \begin{cases} 1 & \text{if } \ t_{ij} \le \alpha t_j^{\text{MIN}} \\ 0 & \text{otherwise.} \end{cases}$$

The *performance profile* [15] of algorithm i is then given by the function

$$p_i(\alpha) = \frac{\sum_{j \in \mathcal{T}} k(t_{ij}, t_j^{\rm MIN}, \alpha)}{|\mathcal{T}|}, \quad \alpha \geq 1.$$

Thus $p_i(1)$ gives the fraction of the examples for which algorithm i is the most effective (according to the statistic t_{ij}), $p_i(2)$ gives the fraction for which algorithm i is within a factor of 2 of the best, and $\lim_{\alpha \to \infty} p_i(\alpha)$ gives the fraction for which the algorithm succeeded.

We consider two explicit factorization preconditioners, one using exact factors ($G = H$), and the other using a simple projection ($G = I$). A Matlab interface to the HSL [34] package `MA57` [20] (version 2.2.1) is used to factorize M_G and subsequently solve (1.4); as we have already mentioned, in some cases it would have been both possible and preferable to use instead the explicit block decomposition (4.1) when $G = I$ (or for easily invertible H), and interpretation of the results we present should keep this in mind. Three implicit factorizations of the form (4.2) with factors (4.3) are also considered. The first is from family 1 (Table 4.1), and aims for simplicity by choosing $P_{31} = I$, $P_{33} = I = B_{33}$ and $B_{22} = I = P_{22}$, and this leads $B_{11} = -(C + I)$; such a choice does not necessarily reproduce any of H, but is inexpensive to use. The remaining implicit factorizations are from family 2 (Table 4.1). The former (marked (a) in the following figures) selects $G_{22} = H_{22}$ while the latter (marked (b) in the figures) chooses $G_{22} = I$; for simplicity we chose $P_{31} = I = B_{31}$, $B_{11} = 0$, $P_{22} = I$ and $P_{33} = -\frac{1}{2}C$ (see section 4.2), and thus we merely require that $B_{22} = H_{22}$ for case (a) and $B_{22} = I$ for case (b)—we use `MA57` to factorize H_{22} in the former case.

Given A, a suitable basis matrix A_1 is found by finding a sparse LU factorization of A^T using the built-in Matlab function `lu`. An attempt to correctly identify the rank is controlled by tight threshold column pivoting, in which any pivot may not be smaller than a factor $\tau = 2$ of the largest entry in its (uneliminated) column [27, 28]. The rank is estimated as the number of pivots, $\rho(A)$, completed before the remaining uneliminated submatrix is judged to be numerically zero, and the indices of the $\rho(A)$ pivotal rows and columns of A define A_1—if $\rho(A) < m$, the remaining rows of A are judged to be dependent, and are discarded. Although such a strategy may not be as robust as, say, a singular-value decomposition or a QR factorization with pivoting, both our and others' experience [27] indicate it to be remarkably reliable and successful in practice. Having found A_1, the factors are discarded, and a fresh LU decomposition of A_1, with a looser threshold column pivoting factor $\tau = 100$, is computed using `lu` in order to try to encourage sparse factors.

All of our experiments were performed using a dual processor Intel Xeon 3.2GHz Workstation with hyperthreading and 2 Gbytes of RAM. Our codes were written and executed in Matlab 7.0 Service Pack 1.

In Figures 5.1–5.2, (see the tables in [17, Appendix C] for the raw data), we compare our five preconditioning strategies for (approximately) solving the problem (1.1) when C is given by (5.2) using the PPCG scheme (variant 2) described in section 2. We consider both low and high(er) accuracy solutions. For the former, we terminate as soon as the residual σ has been reduced more than 10^{-2} from its original value, while the latter requires a 10^{-8} reduction; these are intended to simulate the levels of accuracy that might be required within a nonlinear equation or optimization solver in early (global) and later (asymptotic) phases of the solution process.

We see that if low accuracy solutions suffice, the implicit factorizations appear to be significantly more effective at reducing the residual than their explicit counter-

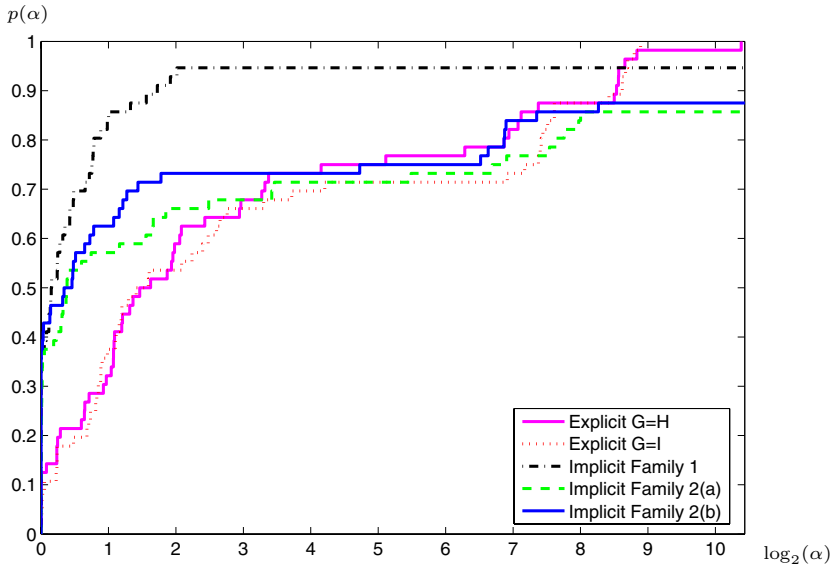

FIG. 5.1. *Performance profile, $p(\alpha)$: CPU time (seconds) to reduce relative residual by 10^{-2}, when C is given by* (5.2).

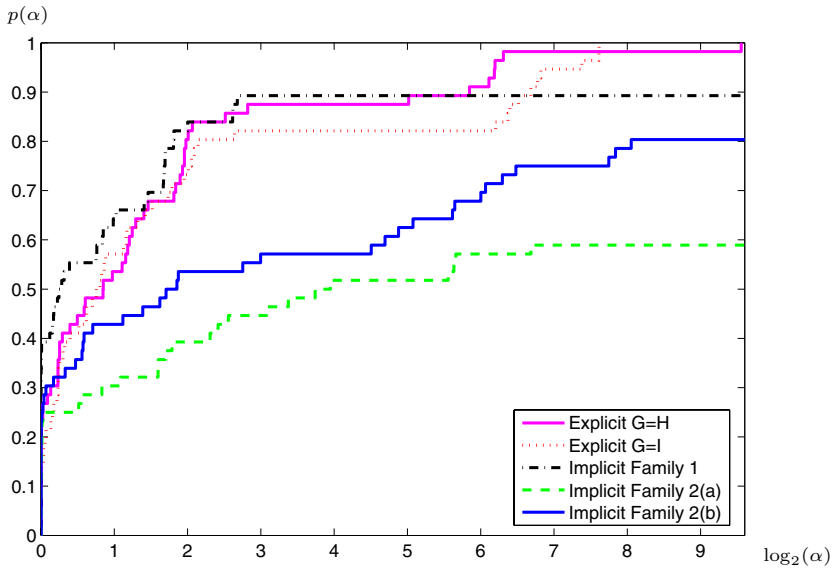

FIG. 5.2. *Performance profile, $p(\alpha)$: CPU time (seconds) to reduce relative residual by 10^{-8}, when C is given by* (5.2).

parts. In particular, the implicit factorization from family 1 seems to be the most effective. Of interest is that for family 2, the cost of applying the more accurate implicit factorization that reproduces H_{22} generally does not pay off relative to the cost of the cheaper implicit factorizations. For higher accuracy solutions, the leading implicit factorization still slightly outperforms the explicit factors, although now the remaining implicit factorizations are less effective.

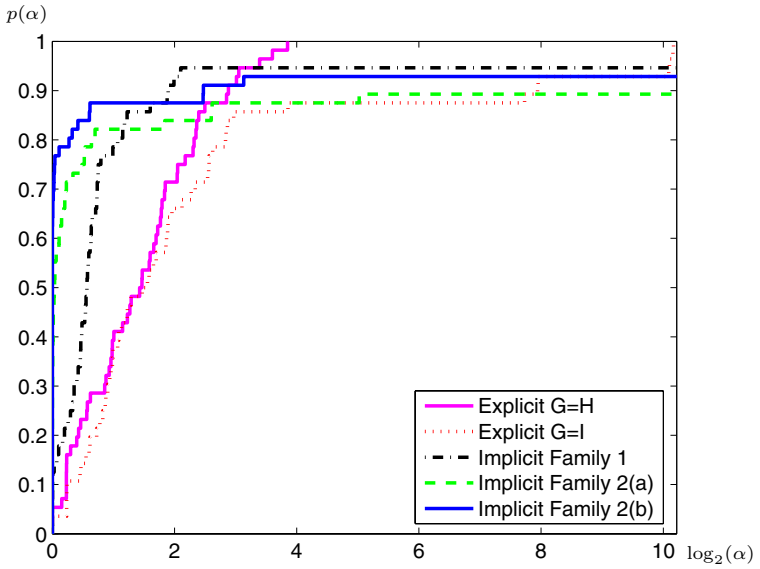

FIG. 5.3. *Performance profile, $p(\alpha)$: CPU time (seconds) to reduce relative residual by 10^{-2}, when C is given by* (5.3).

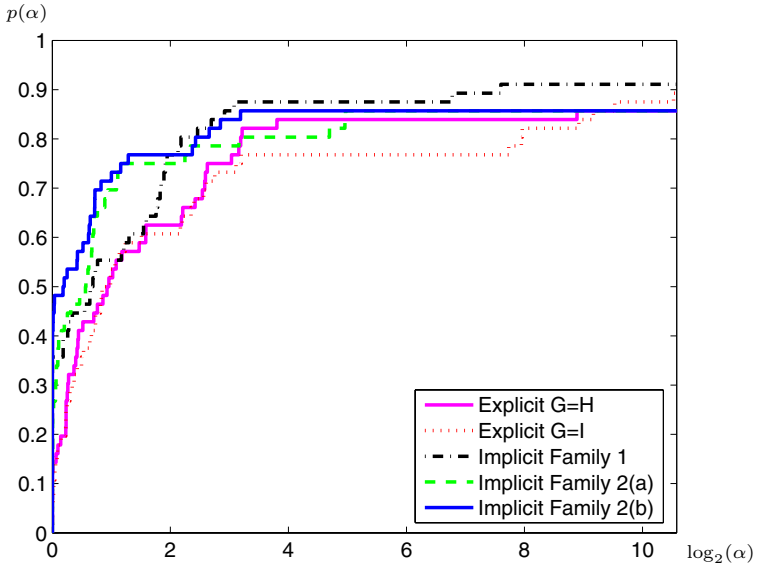

FIG. 5.4. *Performance profile, $p(\alpha)$: CPU time (seconds) to reduce relative residual by 10^{-8}, when C is given by* (5.3).

Figures 5.3–5.4 (see [17] for tables of the raw data) repeat the experiments when C is given by (5.3). Once again the implicit factorizations seem very effective, with a shift now to favor those from family 2, most especially the less sophisticated of these.

6. Comments and conclusions. In this paper we have considered conjugate-gradient like methods for block symmetric indefinite linear systems that arise from regularized saddle-point problems. Such methods require preconditioners that pre-

serve certain sub-blocks from the original systems but allow considerable flexibility for the remaining "noncrucial" blocks. To this end, we have constructed fourteen families of implicit factorizations that are capable of reproducing the required sub-blocks and (some) of the remainder. These generalize known implicit factorizations [18, 19] for the $C = 0$ case. Improved eigenvalue clustering is possible if additionally some of the "noncrucial" blocks are reproduced. We have shown numerically that these implicit-factorization preconditioners can be effective. However, further work is needed to see how these preconditioners compare against special-purpose ones based on (4.1) rather than generic ones using factors of (1.2).

A number of important issues remain. Firstly, we have made no effort to find the best preconditioner(s) from amongst our families, and indeed in most cases have not even tried them in practice. As always with preconditioning, there is a delicate balance between improving clustering of eigenvalues and the cost of doing so, especially since in many applications low accuracy estimates of the solution suffice. We expect promising candidates to emerge in due course, but feel it is beyond the scope of this paper to indicate more than that this is a promising approach.

Secondly, and as we pointed out in [18], the choice of the matrix A_1 is crucial, and considerations of both its stability and sparsity (or other structure), and of its effect on which of the "noncrucial" blocks may be reproduced, are vital. We have precisely defined the algorithm that we have used to select A_1 in the computations presented in this paper, but though this strategy seems to work reasonably across the wide range of test set problems we have computed, we make no claim to its relative quality. The most stringent practical requirement for computation with the preconditioners described in this paper is that there is an effective way to solve linear systems involving A_1.

Thirdly (and possibly related to the point above), when experimenting with family 3 (Table 4.1), we found that some very badly conditioned preconditioners were generated. Specifically, our aim had been to reproduce $G_{21} = H_{21}$, and for simplicity we had chosen $P_{31} = I = B_{31}$ and $B_{22} = I = P_{22}$, and this leads to $P_{21} = H_{21}A_1^{-1}$. Note that we did not try to impose additionally that $G_{22} = H_{22}$ as this would have lead to nontrivial B_{22}. Also notice that we did not need to form P_{21}, merely to operate with it (and its transpose) on given vectors. On examining the relevant spectrum for some small badly conditioned examples, the preconditioner appeared to have worsened rather than improved the range of the eigenvalues for these computations. Whether this is a consequence of requiring two solves with A_1 (and its transpose) when applying the preconditioner rather than the single solve required when not trying to reproduce H_{21}, and whether the same would be true for other families trying to do the same is simply conjecture at this stage. However, it is certainly a cautionary warning.

Acknowledgment. Thanks are due to Mario Arioli both for fruitful discussions on various aspects of this work and for providing us with a Matlab interface to `MA57`. We also thank two referees and the associate editor for their helpful comments.

REFERENCES

[1] O. Axelsson and M. Neytcheva, *Preconditioning methods for linear systems arising in constrained optimization problems*, Numer. Linear Algebra Appl., 10 (2003), pp. 3–31.

[2] R. H. Bartels and G. H. Golub, *The simplex method of linear programming using lu decompositions*, Comm. of the ACM, 12 (1969), pp. 266–268.

[3] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137.

[4] L. Bergamaschi, J. Gondzio, and G. Zilli, *Preconditioning indefinite systems in interior point methods for optimization*, Comput. Optim. Appl., 28 (2004), pp. 149–171.
[5] G. Biros and O. Ghattas, *A Lagrange-Newton-Krylov-Schur method for PDE-constrained optimization*, SIAG/OPT Views-and-News, 11 (2000), pp. 12–18.
[6] J. H. Bramble and J. E. Pasciak, *A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems*, Math. Comput., 50 (1988), pp. 1–17.
[7] R. H. Byrd, M. E. Hribar, and J. Nocedal, *An interior point algorithm for large scale nonlinear programming*, SIAM J. Optim., 9 (2000), pp. 877–900.
[8] T. F. Coleman, *Linearly constrained optimization and projected preconditioned conjugate gradients*, in Proceedings of the Fifth SIAM Conference on Applied Linear Algebra, J. Lewis, ed., SIAM Philadelphia, 1994, pp. 118–122.
[9] T. F. Coleman and A. Pothen, *The null space problem* I: *Complexity*, SIAM J. Algebraic Discrete Methods, 7 (1986), pp. 527–537.
[10] T. F. Coleman and A. Pothen, *The null space problem* II: *Algorithms*, SIAM J. Algebraic Discrete Methods, 8 (1987), pp. 544–563.
[11] T. F. Coleman and A. Verma, *A preconditioned conjugate gradient approach to linear equality constrained minimization*, Technical report, Department of Computer Sciences, Cornell University, Ithaca, New York, 1998.
[12] P. Concus, G. H. Golub, and D. P. O'Leary, *Numerical solution of nonlinear elliptic partial differential equations by a generalized conjugate gradient method*, in Sparse Matrix Computations, J. Bunch and D. Rose, eds., Academic Press, London, 1976, pp. 309–332,
[13] A. R. Conn, N. I. M. Gould, D. Orban, and Ph. L. Toint, *A primal-dual trust-region algorithm for non-convex nonlinear programming*, Math. Prog., 87 (2000), pp. 215–249.
[14] E. de Sturler and J. Liesen, *Block-diagonal and constraint preconditioners for nonsymmetric indefinite linear systems Part* I: *Theory*, SIAM J. Sci. Comput., 26 (2005), pp. 1598–1619.
[15] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Math. Prog., 91 (2002), pp. 201–213.
[16] H. S. Dollar, *Extending constraint preconditioners for saddle-point problems*, Technical report NA-05-02, Oxford University Computing Laboratory, Oxford, England, 2005. (Submitted to SIAM J. Matrix Anal. Appl.)
[17] H. S. Dollar, N. I. M. Gould, W. H. A. Schilders, and A. J. Wathen, *On iterative methods and implicit-factorization preconditioners for regularized saddle-point systems*, Technical report RAL-TR-2005-011, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2005.
[18] H. S. Dollar, N. I. M. Gould, and A. J. Wathen, *On implicit-factorization constraint preconditioners*, Nonconvex Optimization and Its Applications 83, Springer Verlag, 2006.
[19] H. S. Dollar and A. J. Wathen, *Incomplete factorization constraint preconditioners for saddle point problems*, Technical report 04-01, Oxford University Computing Laboratory, Oxford, England, 2004. (To appear in SIAM J. Sci. Comput.)
[20] I. S. Duff, *MA57 - a code for the solution of sparse symmetric definite and indefinite systems*, ACM Trans. Math. Software, 30 (2004), pp. 118–144.
[21] I. S. Duff and J. K. Reid, *The multifrontal solution of indefinite sparse symmetric linear equations*, ACM Trans. Math. Software, 9 (1983), pp. 302–325.
[22] C. Durazzi and V. Ruggiero, *Indefinitely constrained conjugate gradient method for large sparse equality and inequality constrained quadratic problems*, Numer. Linear Algebra Appl., 10 (2002), pp. 673–688.
[23] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite-Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*, Numerical Mathematics and Scientific Computation, Oxford University Press, London, 2005.
[24] J. J. H. Forrest and J. A. Tomlin, *Updating triangular factors of the basis to maintain sparsity in the product form simplex method*, Math. Prog., 2 (1972), pp. 263–278.
[25] R. W. Freund and N. M. Nachtigal, *A new Krylov-subspace method for symmetric indefinite linear systems*, in Proceedings of the 14th IMACS World Congress on Computational and Applied Mathematics, W. F. Ames, ed., IMACS, 1994, pp. 1253–1256.
[26] P. E. Gill, W. Murray, D. B. Ponceleón, and M. A. Saunders, *Preconditioners for indefinite systems arising in optimization*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 292–311.
[27] P. E. Gill, W. Murray, and M. A. Saunders, *SNOPT: An SQP algorithm for large-scale constrained optimization*, SIAM J. Optim., 12 (2002), pp. 979–1006.
[28] P. E. Gill and M. A. Saunders, *private communication*, 1999.
[29] N. I. M. Gould, *On practical conditions for the existence and uniqueness of solutions to the general equality quadratic-programming problem*, Math. Prog., 32 (1985), pp. 90–99.
[30] N. I. M. Gould, *Iterative methods for ill-conditioned linear systems from optimization*, in *Nonlinear Optimization and Related Topics*, G. Di Pillo and F. Giannessi, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999, pp. 123–142.

[31] N. I. M. Gould, M. E. Hribar, and J. Nocedal, *On the solution of equality constrained quadratic problems arising in optimization*, SIAM J. Sci. Comput., 23 (2001), pp. 1375–1394.

[32] N. I. M. Gould, D. Orban, and Ph. L. Toint, *CUTEr (and SifDec), a constrained and unconstrained testing environment, revisited*, ACM Trans. Math. Software, 29 (2003), pp. 373–394.

[33] C. Greif, G. H. Golub, and J. M. Varah, *Augmented Lagrangian techniques for solving saddle point linear systems*, Technical report, Computer Science Department, University of British Columbia, Vancouver, Canada, 2004.

[34] HSL, *A collection of Fortran codes for large-scale scientific computation*, see http://hsl.rl.ac.uk, 2004.

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

[36] L. Lukšan and J. Vlček, *Indefinitely preconditioned inexact Newton method for large sparse equality constrained nonlinear programming problems*, Numer. Linear Algebra Appl., 5 (1998), pp. 219–247.

[37] M. Mihajlovic and D. Silvester, *A black-box multigrid preconditioner for the biharmonic equation*, BIT, 44 (2004), pp. 151–163.

[38] B. A. Murtagh and M. A. Saunders, *Large-scale linearly constrained optimization*, Math. Prog., 14 (1978), pp. 41–72.

[39] B. A. Murtagh and M. A. Saunders, *A projected Lagrangian algorithm and its implementation for sparse non-linear constraints*, Math. Programming Stud., 16 (1982), pp. 84–117.

[40] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Series in Operations Research, Springer, New York, 1999.

[41] L. A. Pavarino, *Preconditioned mixed spectral finite-element methods for elasticity and Stokes problems*, SIAM J. Sci. Comput., 19 (1998), pp. 1941–1957.

[42] I. Perugia and V. Simoncini, *Block-diagonal and indefinite symmetric preconditioners for mixed finite element formulations*, Numer. Linear Algebra Appl., 7 (2000), pp. 585–616.

[43] B. T. Polyak, *The conjugate gradient method in extremal problems*, U.S.S.R. Comput. Math. Math. Phys., 9 (1969), pp. 94–112.

[44] M. Rozložník and V. Simoncini, *Krylov subspace methods for saddle point problems with indefinite preconditioning*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 368–391.

[45] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Comput., 7 (1986), pp. 856–869.

[46] W. Schilders, *A preconditioning technique for indefinite systems arising in electronic circuit simulation*, Talk at the one-day meeting on preconditioning methods for indefinite linear systems, Technische Universiteit, Eindhoven, The Netherlands, December 9, 2002.

[47] W. H. A. Schilders and E. J. W. ter Maten, *Numerical methods in electromagnetics*, in Handbook of Numerical Analysis Vol XIII, P. G. Ciarlet, ed., Elsevier, North Holland, Amsterdam, 2005.

[48] K. Toh, K. Phoon, and S. Chan, *Block preconditioners for symmetric indefinite linear systems*, Internat. J. Numer. Methods Engrg., 60 (2004), pp. 1361–1381.

[49] R. J. Vanderbei and D. F. Shanno, *An interior point algorithm for nonconvex nonlinear programming*, Comput. Optim. Appl., 13 (1999), pp. 231–252.