SIAM J. MATRIX ANAL. APPL.
Vol. 13, No. 1, pp. 292–311, January 1992

021

PRECONDITIONERS FOR INDEFINITE SYSTEMS ARISING IN OPTIMIZATION*

PHILIP E. GILL†, WALTER MURRAY‡, DULCE B. PONCELEÓN§, AND MICHAEL A. SAUNDERS‡

Dedicated to Gene Golub on the occasion of his 60th birthday

Abstract. Methods are discussed for the solution of sparse linear equations $Ky = z$, where K is symmetric and indefinite. Since exact solutions are not always required, direct and iterative methods are both of interest. An important *direct* method is the Bunch–Parlett factorization $K = U^TDU$, where U is triangular and D is block-diagonal. A sparse implementation exists in the form of the Harwell code MA27. An appropriate *iterative* method is the conjugate-gradient–like algorithm SYMMLQ, which solves indefinite systems with the aid of a positive-definite preconditioner.

For any indefinite matrix K, it is shown that the U^TDU factorization can be modified at nominal cost to provide an "exact" preconditioner for SYMMLQ. Code is given for overwriting the block-diagonal matrix D produced by MA27.

The KKT systems arising in barrier methods for linear and nonlinear programming are studied, and preconditioners for use with SYMMLQ are derived.

For nonlinear programs a preconditioner is derived from the "smaller" KKT system associated with variables that are not near a bound. For linear programs several preconditioners are proposed, based on a square nonsingular matrix B that is analogous to the basis matrix in the simplex method. The aim is to facilitate solution of full KKT systems rather than equations of the form $AD^2A^T\Delta\pi = r$ when the latter become excessively ill conditioned.

Key words. indefinite systems, preconditioners, linear programming, nonlinear programming, numerical optimization, barrier methods, interior-point methods

AMS(MOS) subject classifications. 65F05, 65F10, 65F50, 65K05, 90C05

1. Introduction.

Symmetric indefinite systems of linear equations arise in many areas of scientific computation. We will discuss the solution of *sparse* indefinite systems $Ky = z$ by direct and iterative means.

The direct method we have in mind is the Bunch–Parlett factorization $K = U^TDU$, where U is triangular and D is block-diagonal with blocks of dimension 1 or 2 that may be indefinite. Such a factorization exists for any symmetric matrix K [BP71]. (We shall refer to it as the Bunch–Parlett factorization, while noting that the Bunch–Kaufman pivoting strategy is preferred in practice [BK77]. The principal sparse implementation to date is due to Duff and Reid [DR82], [DR83] in the Harwell code MA27. See also [DGRST89].)

The iterative method to be discussed is the Paige–Saunders algorithm SYMMLQ [PS75]. This is a conjugate-gradient–like method for indefinite systems that can make use of a positive-definite preconditioner.

1.1. Preconditioning indefinite systems.

One of our aims is to present a new and simple result that shows how to use the Bunch–Parlett factorization of an

* Received by the editors January 4, 1991; accepted for publication (in revised form) July 26, 1991. This paper was presented at the Second Asilomar Workshop on Progress in Mathematical Programming, February 1990. This research was supported in part by National Science Foundation grant DDM-8715153 and Office of Naval Research grant N00014-90-J-1242.

† Department of Mathematics, University of California at San Diego, La Jolla, California 92093 (peg@optimal.ucsd.edu).

‡ Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, California 94305-4022 (mike@sol-michael.stanford.edu and walter@sol-walter.stanford.edu).

§ Apple Computer, Inc., 20525 Mariani Avenue, Cupertino, California 95014 (dulce@apple.com).

indefinite matrix to construct an exact preconditioner for an iterative method such as SYMMLQ. The intended use is as follows.

Given an indefinite system $Ky = z$ and a related indefinite matrix $\tilde{K}$, we expect that the Bunch–Parlett factorization $\tilde{K} = U^T D U$ will be computed, or will already be available. We show that D can be changed cheaply to provide a positive-definite matrix $M = U^T \bar{D} U$, such that SYMMLQ (with preconditioner M) will solve $\tilde{K}y = z$ in at most two iterations. Hence, M should be a good preconditioner for the original system involving K.

1.2. Optimization. As a source of indefinite systems, we are interested in *barrier methods* or *interior-point methods* for solving linear and nonlinear programs in the following standard form:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & c^T x \\ \text{subject to} & Ax = b, \qquad l \le x \le u, \end{array} \tag{1}$$

where $A \in \Re^{m \times n}$, and

$$\begin{array}{ll} \underset{x}{\text{minimize}} & F(x) \\ \text{subject to} & c(x) = 0, \qquad l \le x \le u, \end{array} \tag{2}$$

where $F(x)$ and $c(x)$ have continuous first and second derivatives. We assume that an optimal solution (x^*, π^*) exists, where π^* is a set of Lagrange multipliers for the constraints $Ax = b$ or $c(x) = 0$.

1.3. KKT systems. When barrier or interior-point methods are applied to these optimization problems, the Karush–Kuhn–Tucker optimality conditions lead to a set of equations of the form

$$\begin{pmatrix} H & A^T \\ A & \end{pmatrix} \begin{pmatrix} \Delta x \\ -\Delta \pi \end{pmatrix} = \begin{pmatrix} -g \\ r \end{pmatrix}, \qquad K \equiv \begin{pmatrix} H & A^T \\ A & \end{pmatrix}, \tag{3}$$

whose solution usually dominates the total computation. The vectors Δx and $\Delta \pi$ are used to update the estimates of x^* and π^*.

For quadratic programs or general nonlinear programs, H is typically a general sparse matrix like A, and it is natural to solve the KKT system as it stands. The Harwell code MA27 has been used in this context by several authors, including Gill et al. [GMSTW86] and Turner [Tur87], [Tur90] for sparse linear programs, by Ponceleón [Pon90] for sparse linear and quadratic programs, and by Burchett [Bur88] for some large nonlinear programs arising in the electric power industry.

1.4. Avoiding AD^2A^T. If H is known to be nonsingular, it is common practice to use it as a block pivot and solve (3) according to the *range-space equations* of optimization:

$$AH^{-1}A^T \Delta \pi = AH^{-1}g + r, \qquad H\Delta x = A^T \Delta \pi - g.$$

For linear programs this is particularly attractive, since H is then a positive diagonal matrix. For example, in a typical primal barrier method, $H = \mu D^{-2}$ where D is diagonal and μ is the barrier parameter ($\mu > 0$) [GMSTW86]. The range-space equations reduce to

$$AD^2A^T \Delta \pi = AD^2 g + \mu r, \qquad \Delta x = \frac{1}{\mu} D^2 (A^T \Delta \pi - g), \tag{4}$$

and most of the work lies in solving the system involving AD^2A^T. When $r = 0$, the numerical properties may be improved by noting that the equation for $\Delta\pi$ reduces to the least-squares problem

$$\min_{\Delta\pi} \|Dg - DA^T\Delta\pi\|_2. \tag{5}$$

However, it is important to observe that *the range-space equations may not give a stable method for solving the* KKT *system if H is ill conditioned.*

1.5. Example. Let

$$A = \begin{pmatrix} 1 & 1 & & \\ 1 & & 1 & \\ 1 & & & 1 \end{pmatrix}, \qquad H = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \mu & \\ & & & \mu \end{pmatrix}, \qquad x = \begin{pmatrix} \sqrt{\mu} \\ \sqrt{\mu} \\ 1 \\ 1 \end{pmatrix},$$

where $\mu \ll 1$, and consider the KKT system (3). This would arise when a primal barrier method is applied to a 3×4 LP problem (1) having $l = 0$, $u = \infty$, when x is the current estimate of x^* and μ is the current barrier parameter. Thus $H = \mu D^{-2}$, where $D = \operatorname{diag}(x_j)$.

The condition numbers of interest are $\operatorname{cond}(K) \approx 6$ (independent of μ) and $\operatorname{cond}(AH^{-1}A^T) = \operatorname{cond}(AD^2A^T) \approx 0.5/\mu$.[1] The latter becomes increasingly large as a solution is approached ($\mu \to 0$), even though K and the original linear program are very well conditioned.

Similar examples are easily constructed. (Indeed, K can be well conditioned even if H is singular.) Thus, we advocate direct or iterative solution of the full KKT system (3) even for linear programs, rather than (4) or (5) according to current practice.

Gay [Gay89, pp. 16–17] has already drawn attention to the lurking numerical difficulties and suggests a middle ground of working with AD^2A^T as long as possible, then switching to a more robust alternative such as direct solves with K.

1.6. Iterative methods and preconditioning. The KKT systems we are concerned with arise when Newton's method is applied to the nonlinear equations defining optimality conditions for barrier subproblems; see §3. In this context, there are not many KKT systems to be solved (compared to those in active-set methods), the systems need not be solved exactly [DES82], and the KKT matrix eventually does not change significantly. It is therefore appropriate to consider iterative methods and preconditioners for the indefinite matrix K.

Previous work on preconditioning for interior-point methods has focused on the LP case and the Schur-complement matrix AD^2A^T. Most authors have used approximate Cholesky factors of AD^2A^T; see, for example, [GMSTW86], [Kar87], [KR88], [Meh89a]. Exact LU factors of DA^T have also been investigated [GMS89].

The success of preconditioned conjugate-gradient methods in this context lends added promise to our proposed use of the much better conditioned KKT systems, now that it is known how to precondition indefinite systems.

1.7. Summary. In §2 we consider general indefinite systems and derive a preconditioner from the Bunch–Parlett factorization. In §3 we consider barrier methods for nonlinear programs, and propose factorizing just part of the KKT system to obtain a preconditioner for the whole system.

[1] We use the spectral condition number, $\operatorname{cond}(K) = \|K^{-1}\|_2\|K\|_2$.

Sections 4 to 6 deal with the LP case. In §4 we propose three preconditioners based on LU factors of a square nonsingular matrix B (analogous to the basis in the simplex method). Section 5 discusses some practical difficulties. Section 6 gives numerical results on the condition numbers of K and AD^2A^T in a typical sequence of barrier subproblems, and compares the preconditioned systems $C^{-1}KC^{-T}$ for several preconditioners CC^T.

2. Preconditioning indefinite systems. Let K be any symmetric nonsingular matrix, and let M be a given positive-definite matrix. Also, let "products with K" mean matrix-vector products of the form $u = Kv$, and "solves with M" mean solution of linear systems of the form $Mx = y$.

The Paige–Saunders algorithm as implemented in SYMMLQ [PS75] may be used to solve $Ky = z$ even if K is indefinite. As with other conjugate-gradient–like algorithms, the matrix is represented by a procedure for computing products with K (those generated by the symmetric Lanczos process).

The first steps towards accelerating the convergence of this algorithm were taken by Szyld and Widlund [SW78], [SW79]. Given a positive-definite matrix M as preconditioner, their algorithm used solves with M in the normal way, but was unconventional in also requiring products with M.

Subsequently, a variant of SYMMLQ was developed that requires only solves with M [Sau79]. To solve $Ky = z$, this variant regards the preconditioner as having the form $M = CC^T$ and implicitly applies the Paige–Saunders algorithm to the system

$$C^{-1}KC^{-T}w = C^{-1}z,$$

accumulating approximations to the solution $y = C^{-T}w$ (without approximating w, which isn't needed). An implementation is available from the *misc* chapter of *netlib* [DG85].

2.1. Use of the Bunch–Parlett factorization. Given any symmetric nonsingular matrix K, there exists a factorization of the form

$$K = P^TU^TDUP,$$

where P is a permutation, U is upper triangular, and D is block-diagonal with blocks of dimension 1 or 2 [BP71]. If K is indefinite, some of the blocks of D will have negative eigenvalues. Let the eigensystem of D be

$$D = Q\Lambda Q^T, \qquad \Lambda = \operatorname{diag}(\lambda_j),$$

and let

$$\bar{D} = Q\bar{\Lambda}Q^T, \qquad \bar{\Lambda} = \operatorname{diag}(|\lambda_j|),$$

be a closely related positive-definite matrix that can be obtained at minimal cost. If we define $C = P^TU^T\bar{D}^{1/2}$, it is easily verified that

$$\bar{K} \equiv C^{-1}KC^{-T} = \operatorname{diag}(\lambda_j/|\lambda_j|) = \operatorname{diag}(\pm 1).$$

This means that the "perfect" preconditioner for K is the matrix

$$M = CC^T = P^TU^T\bar{D}UP,$$

since the "preconditioned" matrix $\bar{K}$ has at most two distinct eigenvalues and the Paige–Saunders algorithm converges in at most two iterations.

In practice, M will be computed from the Bunch–Parlett factorization of an *approximation* to K.

2.2. Modification of D from MA27. The block-diagonal matrix D is packed in the MA27 data structure as a sequence of matrices of the form

$$\begin{pmatrix} \alpha \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \alpha & \beta \\ \beta & \gamma \end{pmatrix}.$$

In the 1×1 case, we do nothing if $\alpha > 0$; otherwise we reverse its sign. In the 2×2 case, we do nothing if $\alpha\gamma > \beta^2$; otherwise we compute the eigensystem in the form

$$\begin{pmatrix} \alpha & \beta \\ \beta & \gamma \end{pmatrix} = \begin{pmatrix} c & s \\ s & -c \end{pmatrix} \begin{pmatrix} \lambda_1 & \\ & \lambda_2 \end{pmatrix} \begin{pmatrix} c & s \\ s & -c \end{pmatrix},$$

where $c^2 + s^2 = 1$. We then form the positive-definite matrix

$$\begin{pmatrix} \bar{\alpha} & \bar{\beta} \\ \bar{\beta} & \bar{\gamma} \end{pmatrix} = \begin{pmatrix} c & s \\ s & -c \end{pmatrix} \begin{pmatrix} |\lambda_1| & \\ & |\lambda_2| \end{pmatrix} \begin{pmatrix} c & s \\ s & -c \end{pmatrix},$$

and overwrite the appropriate three locations of MA27's storage.

The techniques for computing the 2×2 eigensystem are central to Jacobi's method for the symmetric eigenvalue problem. They were developed by Rutishauser [Rut66]. We have followed the description in Golub and Van Loan [GV89, p. 446] with minor changes to work with symmetric plane rotations.

A subroutine for modifying the D computed by MA27 is given in the Appendix.

2.3. Aasen's method. In general, Aasen's tridiagonalization method [Aas71] is considered competitive with the Bunch–Kaufman approach [BK77] for solving dense indefinite systems. Aasen's method computes a factorization of the form $K = U^T T U$ where T is tridiagonal.

We do not know of a sparse implementation, but in any event we note that it would not be ideal for producing a preconditioner in the manner described above, since the eigensystem for T would involve far more work than for the block-diagonal D of the Bunch–Parlett factorization.

On the other hand, we could compute a (very special) Bunch–Parlett factorization of T and modify the associated D as described above.

3. Barrier subproblems. We return now to the optimization problems (1)–(2). In barrier methods, the bounds $l \le x \le u$ are absorbed into the objective function and we solve a sequence of perturbed subproblems, typically of the form

$$\begin{aligned} &\underset{x}{\text{minimize}} && F_\mu(x) = F(x) - \mu \sum_{j=1}^{n} (\ln(x_j - l_j) + \ln(u_j - x_j)) \\ &\text{subject to} && c(x) = 0, \end{aligned} \tag{6}$$

where the barrier parameter μ takes decreasing positive values that are eventually very small. If $l_j = -\infty$ or $u_j = \infty$ for some j, the corresponding terms $\ln(x_j - l_j)$ or $\ln(u_j - x_j)$ are omitted. If x_j has no bounds, both terms may be omitted.

The following quantities are needed:

$L_\mu(x, \pi) = F_\mu(x) - \pi^T c(x)$	the Lagrangian function,
$g_\mu(x) = \nabla F_\mu(x)$	the gradient of the barrier function,
$g_L(x, \pi) = g_\mu(x) - A(x)^T \pi$	the gradient of the Lagrangian,
$H_L(x, \pi) = \nabla^2 F_\mu(x) - \sum \pi_i \nabla^2 c_i(x)$	the Hessian of the Lagrangian, and
$A(x) = \nabla c(x)$	the Jacobian of the constraints.

For convenience we assume that $A(x) \in \Re^{m \times n}$ has full row rank m, and that the scaling of the problem is reasonable, so that $\|A(x)\| \approx 1$.

3.1. Newton's method and the KKT system. The optimality conditions for (6) are the nonlinear equations

$$g_L(x, \pi) = 0, \tag{7}$$

$$c(x) = 0. \tag{8}$$

Newton's method may be applied directly, or to some equivalent system. Given suitable initial values for the primal and dual variables (x, π), the key set of equations for generating a search direction is the KKT system

$$\begin{pmatrix} H_L & A^T \\ A & \end{pmatrix} \begin{pmatrix} \Delta x \\ -\Delta \pi \end{pmatrix} = - \begin{pmatrix} g_L \\ c \end{pmatrix}, \tag{9}$$

where the KKT matrix and right-hand side are evaluated at the current point (x, π). A positive steplength α is then chosen to reduce some measure of the size of the right-hand side (g_L, c), and the variables are updated according to $x \leftarrow x + \alpha \Delta x$, $\pi \leftarrow \pi + \alpha \Delta \pi$. (Sometimes a different α may be used for x and π.)

3.2. A preconditioner for K. In general, some of the variables converge to values near their upper or lower bounds. For such variables x_j, the Hessian H_L includes on its diagonal a term that becomes very large: $\mu/(x_j - l_j)^2$ or $\mu/(u_j - x_j)^2$, which are $O(1/\mu)$. Let the KKT matrix be partitioned accordingly:

$$K = \begin{pmatrix} K_1 & K_3^T \\ K_3 & K_2 \end{pmatrix}, \tag{10}$$

where K_1 is the part of H_L associated with variables near a bound, and K_2 looks like a smaller KKT system associated with the remaining variables. This partitioning is crucial to the sensitivity analysis in [Pon90]. Of course, the partition depends on the measure of closeness to a bound, but it is not critical here except that the dimension of K_1 should not exceed $n - m$.

One possible approximation to K is

$$\begin{pmatrix} D_1 & \\ & K_2 \end{pmatrix}, \tag{11}$$

where D_1 is a diagonal matrix containing the diagonals of K_1, which by construction are large and positive. Applying the method of §2, we can now obtain a positive-definite preconditioner for K as follows:

$$K_2 = U_2^T D_2 U_2, \qquad M = \begin{pmatrix} D_1 & \\ & U_2^T \bar{D}_2 U_2 \end{pmatrix}, \tag{12}$$

where $\bar{D}_2$ is obtained from D_2 at nominal cost.

3.3. Discussion. In broad terms, we need to estimate which variables are going to be "free" (away from their bounds) at a solution. If $m \ll n$, the KKT system K_2 associated with the free variables may be much smaller than the whole of K, and the cost of the Bunch–Parlett factorization of K_2 may be acceptably low.

For the early iterations of Newton's method, the estimate of K_2 will usually be poor, and the diagonal term D_1 will not be particularly large. However, following the inexact Newton approach [DES82], only approximate solutions to the KKT system are needed, and the iterative solver need not perform many iterations.

As the Newton iterations converge and the partition (10) becomes more sharply defined, the preconditioner should become increasingly powerful and produce the increasingly accurate solutions required at an acceptable cost.

3.4. Performance of SYMMLQ with the MA27 preconditioner. The approach of §§2 and 3 has been tested by Burchett [Bur89] within a barrier algorithm for solving some large nonlinear problems in optimal power flow.

Normally, MA27 is used to factorize K at each iteration of the barrier algorithm, with H_L and A in (9) changing each time. For experimental purposes, the factors of K at iteration k were used to construct a preconditioner for iteration $k+1$ (via subroutine `syprec` in the Appendix). The dimension of K was 6000 for one problem and 16,000 for another.

Initially, SYMMLQ required about 30 iterations to solve the KKT systems to moderate accuracy. As the barrier algorithm (and A) converged, the number of iterations required by SYMMLQ fell to about 10. This performance seems very promising.

4. Preconditioners for linear programming. For linear programs the structure of the partitioned KKT system (10) can be investigated more closely, given that optimal solutions are often associated with a vertex of the feasible region. We partition the constraint matrix into the form $A = (\,N \;\; B\,)$, where B is square and nonsingular, and N in some sense corresponds to the $n-m$ variables that are closest to a bound.

The Hessian for the barrier function is a diagonal matrix H, which we partition as $H = \operatorname{diag}(H_N, H_B)$. The KKT system is then

$$K = \begin{pmatrix} H_N & & N^T \\ & H_B & B^T \\ N & B & \end{pmatrix}.$$

As convergence occurs, the diagonals of $H_N \to \infty$ (and in general $\operatorname{cond}(K) \to \infty$). In degenerate cases, some diagonals of H_B may also become very large.

In various primal, dual, and primal-dual interior-point algorithms for LP, similar matrices K arise with varying definitions of H (e.g., [Meg86], [KMY88], [LMS89], [Meh89a], [Meh90]). The discussion hereafter applies to all such methods.

In the following sections we introduce a series of preconditioners of the form $M = CC^T$. To improve the convergence of SYMMLQ, the transformed matrices $\bar{K} = C^{-1}KC^{-T}$ should have a better condition than K or a more favorable distribution of eigenvalues (clustered near ± 1). We make use of the quantities

$$V = B^{-T}H_B B^{-1}, \qquad W = NH_N^{-1/2},$$

and are motivated by the fact that $V \to 0$ and $W \to 0$ in nondegenerate cases. The effects of degeneracy are discussed later.

4.1. The preconditioner M_1. The first preconditioner is diagonal and is intended to eliminate the large diagonals of K:

$$M_1 = C_1 C_1^T = \begin{pmatrix} H_N & & \\ & I & \\ & & I \end{pmatrix}, \tag{13}$$

$$\bar{K}_1 = C_1^{-1} K C_1^{-T} = \begin{pmatrix} I & & W^T \\ & H_B & B^T \\ W & B & \end{pmatrix}. \tag{14}$$

With diagonal preconditioning, there is no loss of precision in recovering solutions for the original system. Thus as H_N becomes large, the preconditioned matrix $\bar{K}_1$ tends to represent the true sensitivity of the KKT system with regard to solving linear equations.

We will use $\bar{K}_1$ later for comparing condition numbers.

4.2. The preconditioner M_2. The second preconditioner is block diagonal:

$$M_2 = C_2 C_2^T = \begin{pmatrix} H_N & & \\ & B^T B & \\ & & I \end{pmatrix}, \tag{15}$$

$$\bar{K}_2 = C_2^{-1} K C_2^{-T} = \begin{pmatrix} I & & W^T \\ & V & I \\ W & I & \end{pmatrix}. \tag{16}$$

Since V and W tend to become small, M_2 tends towards being an exact preconditioner for K. We see that a Bunch–Parlett factorization is no longer needed. In order to solve systems involving M_2, we may use any sparse factorization of B or B^T.

4.3. The preconditioner M_3. The third preconditioner is designed to eliminate the submatrix V in (16), for degenerate cases where V is not adequately small:

$$M_3 = C_3 C_3^T, \qquad C_3 = \begin{pmatrix} H_N^{1/2} & & \\ & B^T & \frac{1}{2} H_B B^{-1} \\ & & I \end{pmatrix}, \tag{17}$$

$$\bar{K}_3 = C_3^{-1} K C_3^{-T} = \begin{pmatrix} I & -\frac{1}{2} W^T V & W^T \\ -\frac{1}{2} VW & & I \\ W & I & \end{pmatrix}. \tag{18}$$

The off-diagonal term in (17) can be derived by observing that for a KKT matrix of the form

$$K = \begin{pmatrix} H & B^T \\ B & \end{pmatrix}, \qquad B \text{ square},$$

we would like $M = CC^T$ to satisfy

$$C^{-1} K C^{-T} = \begin{pmatrix} & I \\ I & \end{pmatrix} \equiv J,$$

or equivalently, $CJC^T = K$. Letting C be of the form

$$C = \begin{pmatrix} B^T & E \\ & I \end{pmatrix},$$

we find that E should satisfy $EB + B^T E^T = H$. The simplest choice is then to set $E = \frac{1}{2} H B^{-1}$.

Though V has been eliminated, we have now introduced the term $-\frac{1}{2} VW$, and solves with M_3 cost twice as much as solves with M_2. The expected benefit is that $\frac{1}{2} VW$ should be smaller than V itself.

4.4. The preconditioner M_4. The fourth preconditioner also eliminates V, using the factorization $B^T = LU$, where we intend that L be well conditioned:

$$M_4 = C_4 C_4^T, \qquad C_4 = \begin{pmatrix} H_N^{1/2} & & \\ & L & \frac{1}{2}H_B L^{-T} \\ & & U^T \end{pmatrix}, \tag{19}$$

$$\bar{K}_4 = C_4^{-1} K C_4^{-T} = \begin{pmatrix} I & -\frac{1}{2}\bar{W}^T\bar{V} & \bar{W}^T \\ -\frac{1}{2}\bar{V}\bar{W} & & I \\ \bar{W} & I & \end{pmatrix}, \tag{20}$$

where

$$\bar{V} = L^{-1} H_B L^{-T}, \qquad \bar{W} = U^{-T} N H_N^{-1/2}.$$

As before, letting C be of the form

$$C = \begin{pmatrix} L & E \\ & U^T \end{pmatrix}$$

and requiring $CJC^T = K$, we find that $EL^T + LE^T = H$, and we take $E = \frac{1}{2}HL^{-T}$.

Solves with M_4 are cheaper than with M_3. Comparing (18) and (20), a further advantage is that $\bar{V}\bar{W} = UVW$ tends to be smaller than VW, although $\bar{W} = U^{-T}W$ is probably larger than W.

4.5. The preconditioner M_D. We mention one further diagonal preconditioner that has appeared implicitly in the literature for the case $H = \mu D^{-2}$ with D diagonal. It does not depend on the N-B partitioning, and gives a transformed system that does not involve μ:

$$M_D = C_D C_D^T = \begin{pmatrix} \mu D^{-2} & \\ & \mu^{-1} I \end{pmatrix}, \tag{21}$$

$$\bar{K}_D = C_D^{-1} K C_D^{-T} = \begin{pmatrix} I & DA^T \\ AD & \end{pmatrix}. \tag{22}$$

The matrix $\bar{K}_D$ is associated with weighted least-squares problems of the form (5), as discussed in [GMSTW86]. Turner [Tur87], [Tur90] has investigated the use of MA27 to obtain exact factors of both K and $\bar{K}_D$. An important practical observation was that MA27 produced much sparser factors for $\bar{K}_D$ than for K.

Unfortunately, the numerical examples in §6 show that $\bar{K}_D$ has essentially the same condition as AD^2A^T, which tends to be much more ill conditioned than $\bar{K}_1$ (14). We therefore cannot recommend the use of $\bar{K}_D$.

Indeed, when $\|AD\| \approx 1$ as we have here, it can be shown that $\text{cond}(\bar{K}_D) \approx \text{cond}(AD)^2$. To improve the condition of $\bar{K}_D$ we should use

$$M_D = C_D C_D^T = \begin{pmatrix} \alpha^{-1}\mu D^{-2} & \\ & \alpha\mu^{-1} I \end{pmatrix}, \tag{23}$$

$$\bar{K}_D = C_D^{-1} K C_D^{-T} = \begin{pmatrix} \alpha I & DA^T \\ AD & \end{pmatrix} \tag{24}$$

for some $\alpha \in (0, \|AD\|_2]$, since it is known that $\text{cond}(\bar{K}_D) \approx \text{cond}(AD)$ can be achieved if $\alpha \approx \sigma_{\min}$, the smallest singular value of AD (Björck [Bjo67], [Bjo91]). Experiments in this direction have been performed by Arioli, Duff, and De Rijk [ADR89] (who also give error analyses) and by Fourer and Mehrotra [FM91].

For the sake of both direct and iterative methods for solving KKT systems, it is hoped that further development of MA27 will result in greatly improved sparsity in the factors of K and/or $\bar{K}_1$. At the time of writing, a new code MA47 holds much promise (see Duff et al. [DGRST89]), as does an analogous code described in [FM91].

4.6. Regularizing K and AD^2A^T. Since A often does not have full row rank, it is important to include a regularization parameter $\delta > 0$ in the KKT system. Thus (3) becomes

$$\begin{pmatrix} H & A^T \\ A & -\delta I \end{pmatrix} \begin{pmatrix} \Delta x \\ -\Delta \pi \end{pmatrix} = \begin{pmatrix} -g \\ r \end{pmatrix}. \tag{25}$$

Systems of this type have been studied in the context of sequential quadratic programming by Murray [Mur69], Biggs [Big75], and Gould [Gou86].

In practice, a wide range of values of δ may be used without inhibiting convergence, particularly with methods that do not maintain primal feasibility ($A\Delta x = 0$). For example, we would recommend values in the range $10^{-8} \le \delta \le 10^{-4}$ on a machine with about 16 digits of precision, assuming $\|A\| \approx 1$.

Note that the corresponding system (4) becomes

$$(AD^2A^T + \mu\delta I)\Delta\pi = AD^2g + \mu r. \tag{26}$$

When μ is as small as 10^{-10} (say), one would have to choose a rather large δ (say, $\delta \ge 10^{-2}$) to achieve any degree of regularization of AD^2A^T. This constitutes a large perturbation to the underlying KKT system (25).

In other words, a much smaller δ is sufficient to regularize (25) than (26). Thus, KKT systems again show an advantage over AD^2A^T.

With regard to the preconditioners, δ introduces terms $-\delta I$, $-\delta I$, $-\delta U^{-T}U^{-1}$ into the bottom corner of $\bar{K}_2$, $\bar{K}_3$, $\bar{K}_4$, respectively. For $\bar{K}_4$ it appears that δ must be chosen quite small and that the choice of B must be flexible enough to prevent U from being excessively ill conditioned (see §5.3).

5. Use of LU factors. For linear programs, the "small" KKT matrix in (10) is of the form

$$K_2 = \begin{pmatrix} H_B & B^T \\ B & \end{pmatrix}.$$

As in the general nonlinear case we could obtain a preconditioner from a Bunch–Parlett factorization of K_2, and in practice this may prove to be a good approach.

The preconditioners M_2, M_3, and M_4 were derived on the assumption that it should be cheaper to compute sparse factors of just the matrix B. We propose to use the package LUSOL [GMSW87] to obtain $B^T = LU$, where L is a permuted lower triangle with unit diagonals. A user-defined tolerance limits the size of the off-diagonals of L (typically to 5, 10, or 100), thereby limiting the condition of L as required.

5.1. Choice of B. One of the main practical difficulties will be in choosing a "good" square matrix B at each stage. The current values of x and/or the estimated reduced costs $z = c - A^T\pi$ should provide some guidance. For example, the diagonal matrix H is defined in terms of these quantities, and the smallest $m + s$ diagonals of H could be used to pinpoint a submatrix $\bar{A}$ of A (for some moderate $s \geq 0$). LUSOL could then be used to obtain a rectangular factorization $\bar{A}^T = LU$. The first m pivot rows and columns may suggest a suitable B.

Alternative approaches to choosing B have been suggested by Gay [Gay89], Tapia and Zhang [TZ89], Mehrotra [Meh89b], and others. These remain to be explored.

5.2. The effects of degeneracy on V and W. In general, primal degeneracy will mean that certain elements of H_B do not tend to zero, so that not all of V or $\bar{V}$ will become small. Similarly, dual degeneracy will mean that certain elements of H_N will not become large, and not all of W or $\bar{W}$ will become small.

The main effect is that the preconditioners will be less "exact." Either form of degeneracy is likely to increase the number of SYMMLQ iterations required.

5.3. Singular systems. Whatever the method for choosing a square B, it is probable that B will be singular (since in many practical cases, A does not have full row rank). At present we propose to rely on the fact that LUSOL will compute a stable singular factorization of the form

$$B^T = \begin{pmatrix} L_1 \\ L_2 \end{pmatrix} \begin{pmatrix} U_1 & U_2 \end{pmatrix},$$

and the solve procedures will treat this as if it were the factorization of a nonsingular matrix

$$\bar{B}^T = \begin{pmatrix} L_1 & \\ L_2 & I \end{pmatrix} \begin{pmatrix} U_1 & U_2 \\ & I \end{pmatrix}.$$

User-defined tolerances determine how ill conditioned U_1 is allowed to be (and hence determine its dimension).

Alternatively, we may use the factorization $B_1^T = L_1U_1$ to transform most of K as already described. Certain rows of A will not be transformed in the preferred way, and again the effect will be to increase the number of SYMMLQ iterations required.

6. Numerical examples for the LP case. Here we investigate the effect of the preconditioners described in §4. For test purposes we have used MATLAB™ [MLB87] to implement a primal-dual interior-point algorithm for the standard LP problem min c^Tx subject to $Ax = b$, $x \geq 0$. The linear system to be solved each iteration is

$$\begin{pmatrix} H & A^T \\ A & -\delta I \end{pmatrix} \begin{pmatrix} \Delta x \\ -\Delta\pi \end{pmatrix} = \begin{pmatrix} -g \\ r \end{pmatrix}, \tag{27}$$

where $H = X^{-1}Z$, $X = \operatorname{diag}(x_j)$, $Z = \operatorname{diag}(z_j)$, $r = b - Ax$, $g = c - A^T\pi - \mu X^{-1}e$, and e is the vector of ones. The search direction for z is $\Delta z = X^{-1}(\mu e - Z\Delta x) - z$.

The rows and columns of A were scaled to give $\|A\| \approx 1$. The starting values were $x = e$, $z = e$, $\pi = 0$ (so that $H = I$ initially), and δ was fixed at 10^{-8}, with the machine precision on a DEC VAX system being around 16 digits. The parameter μ was reduced every iteration according to the steplengths for x and z: $\mu \leftarrow \mu - \alpha_\mu\mu$, where $\alpha_\mu = \min(\alpha_x, \alpha_z, 0.99)$, and α_x, α_z were limited in the usual way to be at most

TABLE 1
Condition numbers for problem exp1.

k	μ	AD^2A^T	$\bar{K}_D$	K	$\bar{K}_1$	$\bar{K}_2$	$\bar{K}_3$	$\bar{K}_4$	B rank-def
1	1.9e-2	1.2e1	1.1e1	1.1e1	1.1e1	3.1e1	2.6e1	1.1e1	
2	4.5e-3	6.5e1	3.1e1	6.8e2	1.9e1	5.5e1	2.7e1	2.7e1	
3	4.8e-4	1.2e3	2.2e3	4.4e4	6.9e1	4.6e2	5.7e1	1.6e1	
4	1.3e-4	1.7e5	3.1e5	1.0e6	4.2e3	4.4e3	1.5e3	1.6e3	2
5	3.2e-5	2.9e5	5.3e5	1.3e6	7.7e2	2.3e3	5.4e2	5.6e2	1
6	1.5e-5	4.0e5	5.8e5	3.1e5	6.3e2	2.8e3	7.4e2	7.5e2	1
7	4.6e-6	4.0e5	5.8e5	1.6e5	1.2e2	3.0e2	1.1e2	1.1e2	1
8	1.6e-6	4.7e5	6.3e5	3.0e5	3.9e1	1.4e2	1.9e1	8.4e0	
9	1.4e-7	9.1e5	1.1e6	4.9e5	2.0e1	4.7e0	1.3e0	1.5e0	
10	7.8e-9	8.4e5	1.0e6	3.5e6	1.7e1	1.1e0	1.1e0	1.3e0	

1 or 0.99 times the step to the boundaries $x > 0$, $z > 0$, respectively. See [KMY88], [MMS89], [LMS89], and [Meh90] for related details.

Condition numbers of various matrices were obtained using MATLAB's function *rcond*. The square matrices B for the preconditioners of §4 were obtained from the columns of A for which $H_{jj} \leq 20$. The diagonals H_{jj} were first sorted and up to $1.2m$ of the smallest were used to select a rectangular matrix $\bar{A}$ from A. In practice, a sparse LU factorization of $\bar{A}$ or $\bar{A}^T$ would extract a full-rank submatrix, but here we used MATLAB's function $qr(A)$ to elicit a full-rank set of columns (via a QR factorization with column interchanges), and a second QR factorization of part of $\bar{A}^T$ to pinpoint a full-rank set of rows. The dimension of the resulting matrix B is generally less than m. The "rank" was determined from the first QR factorization by requiring the diagonals of R to be greater than 10^{-4}.

6.1. A nondegenerate example. To illustrate ideal behavior of the preconditioners, we chose a nondegenerate problem *exp1* [Bla82] in which A is 10 by 17 (including 10 unit columns associated with slack variables). The lack of primal or dual degeneracy means that near a solution, $m = 10$ diagonals of H are substantially less than 1, and $n - m = 7$ diagonals are significantly greater than 1. The choice of B is ultimately clear cut.

Table 1 lists various condition numbers for each iteration of the primal-dual algorithm. For interest, we include AD^2A^T and $\bar{K}_D$, which were defined in terms of $D = X = \text{diag}(x_j)$ (see §4.5) and incorporated the same regularization δ (§4.6). It may be seen that both $AD^2A^T + \mu\delta I$ and $\bar{K}_D$ become increasingly ill conditioned in step with K, in contrast to the "meaningful" condition of K reflected by $\bar{K}_1$ (in which the large diagonals of H have been scaled to 1).

The preconditioned systems $\bar{K}_2$, $\bar{K}_3$, and $\bar{K}_4$ show an increasing, though apparently mild, improvement over $\bar{K}_1$. Their effectiveness depends on the choice of B and whether or not it has dimension m. The column labeled "B rank-def" records the corresponding rank-deficiency. The conditions of B, L, and U were less than 25, 7, and 40, respectively, for all iterations.

Low conditions are always a good sign, but high ones tell an incomplete story. Figure 1 shows more clearly the increasing improvement of the preconditioners M_2, M_3, M_4 in terms of the clustering of the eigenvalues of $\bar{K}_2$, $\bar{K}_3$, $\bar{K}_4$ around ± 1. The KKT systems have dimension $m+n = 27$. Eigenvalues in the range $(-5, 5)$ are plotted exactly; the remainder are compressed into the ranges $(-6, -5)$ and $(5, 6)$. Thus, $\bar{K}_2$

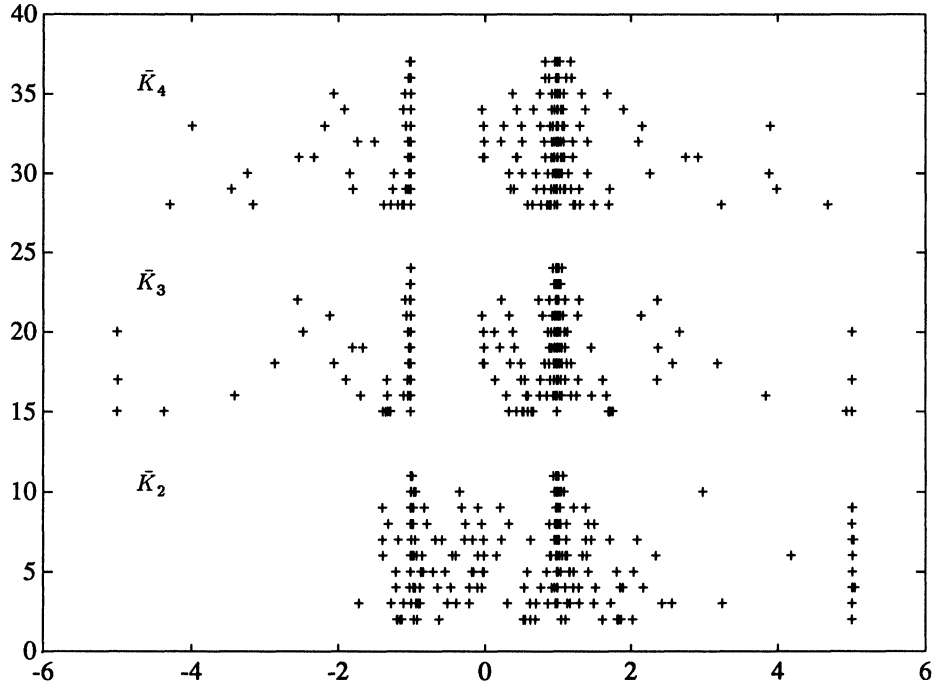

FIG. 1. *Eigenvalues for* $\bar{K}_2$, $\bar{K}_3$, $\bar{K}_4$ *for problem exp*1.

has one or two eigenvalues greater than 5 for the first eight iterations, whereas $\bar{K}_3$ has its eigenvalues inside $(-5, 5)$ at all times. (The vertical axis is "iteration number" shifted by 1 for $\bar{K}_2$, 14 for $\bar{K}_3$, and 27 for $\bar{K}_4$. Each horizontal line gives the spectrum of one of these matrices at the corresponding iteration.)

It is evident from Fig. 1 that $\bar{K}_3$ and $\bar{K}_4$ have more favorable eigenvalue distributions than $\bar{K}_2$, and that $\bar{K}_4$ is marginally better than $\bar{K}_3$, the main benefit being that it is more cheaply obtained. There is a striking absence of eigenvalues in the range $(-1+\beta, -\beta)$ for some small β, though we have no immediate explanation. This range broadens to $(-1+\beta, 1-\beta)$ for all systems at the final iteration, as we may expect.

6.2. A more typical example. Table 2 and Fig. 2 give similar results for the well-known problem *afiro* [Gay85]. The matrix A is 27 by 51, including 19 slack columns. We see that $AD^2A^T + \mu\delta I$ and $\bar{K}_D$ again become extremely ill conditioned in step with K.

The KKT systems have dimension 78. As before there is a clear division between large and small diagonals of H near a solution, but in this case only $m-5$ are substantially smaller than one. The rank of the corresponding columns of A is $m-7$, consistent with B's final rank-deficiency of 7. The conditions of B, L, and U were again low: less than 35, 13, and 34, respectively.

It is encouraging to observe that Fig. 2 is qualitatively similar to Fig. 1 in spite of the rank-deficiency in B. The main difference is two eigenvalues close to zero on the last iteration, in keeping with the difference between $m-5$ and $m-7$. Results

TABLE 2
Condition numbers for problem afiro.

k	μ	AD^2A^T	$\bar{K}_D$	K	$\bar{K}_1$	$\bar{K}_2$	$\bar{K}_3$	$\bar{K}_4$	B rank-def
1	2.6e-2	6.0e1	2.2e1	3.3e1	2.3e1	1.3e2	1.8e2	9.8e1	1
2	9.9e-3	3.1e2	2.0e2	1.8e3	1.2e2	7.0e2	1.1e2	4.4e1	1
3	1.8e-3	2.5e3	3.6e3	3.0e4	4.7e2	9.4e3	3.1e3	4.9e2	1
4	5.4e-4	1.7e4	3.4e4	1.1e6	2.2e3	4.1e4	1.1e4	7.0e3	1
5	2.9e-4	8.2e3	1.7e4	3.0e5	6.9e2	2.4e4	1.2e3	3.1e2	3
6	3.3e-5	8.5e3	1.8e4	4.5e4	1.0e2	1.1e3	4.6e2	2.9e2	
7	2.4e-5	2.0e4	3.9e4	1.4e6	3.8e2	3.0e3	6.5e2	4.8e2	1
8	8.5e-6	2.6e5	4.3e5	1.8e7	9.0e3	1.6e4	2.5e3	2.9e3	3
9	3.1e-6	1.7e7	2.7e7	1.1e8	7.8e4	5.3e3	2.5e3	2.2e3	6
10	4.3e-7	2.0e8	3.1e8	2.0e9	1.7e6	9.6e4	2.3e4	1.6e4	6
11	4.3e-9	2.e10	4.e10	4.e11	5.7e8	3.7e5	3.9e5	2.6e5	7

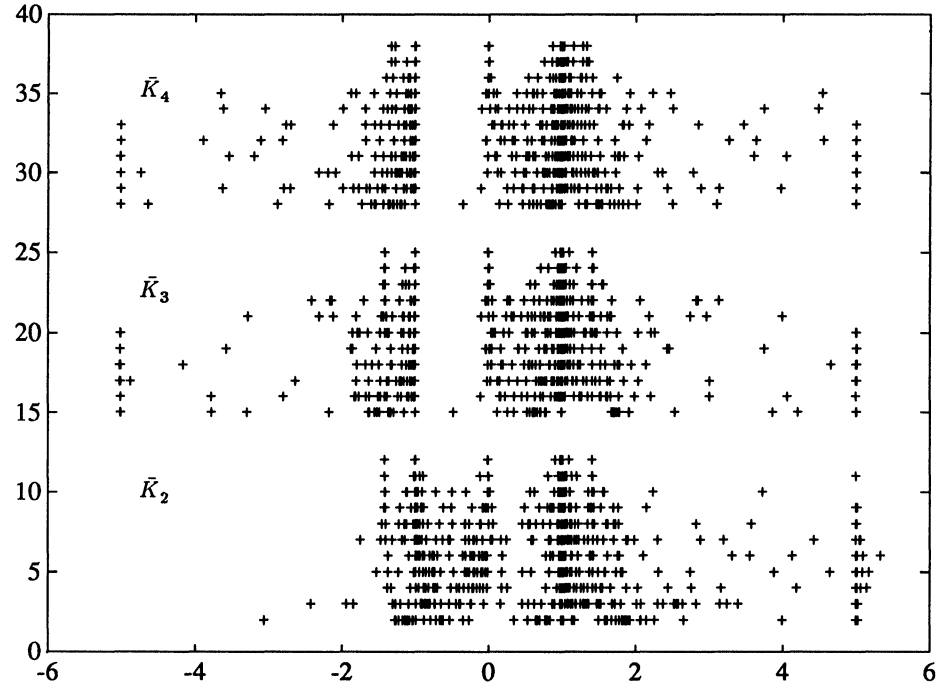

FIG. 2. *Eigenvalues for $\bar{K}_2$, $\bar{K}_3$, $\bar{K}_4$ for problem afiro.*

are similar for the second to last iteration. We can expect a low number of SYMMLQ iterations will be required as the barrier algorithm converges, as in the ideal nondegenerate case.

6.3. Performance of SYMMLQ with the LP preconditioners. As a further experiment we modified the barrier LP algorithm to solve (27) using SYMMLQ with the first four preconditioners M_1–M_4 of §4. We applied the LP algorithm to problem *exp*1 with and without scaling of the data. The iterations required by SYMMLQ

TABLE 3
SYMMLQ *iterations with various preconditioners on problem exp*1.

	Scaled				Unscaled			
k	M_1	M_2	M_3	M_4	M_1	M_2	M_3	M_4
1	17	21	24	17	44	21	25	82
2	28	25	21	21	72	80	68	98
3	29	27	19	19	95	76	72	91
4	31	29	17	21	88	73	69	83
5	31	23	18	22	55	51	52	57
6	27	19	14	16	51	36	29	42
7	25	14	12	14	42	27	25	31
8	24	12	9	11	37	20	20	26
9	23	8	8	9	35	12	12	18
10	23	6	6	7	32	9	9	13
11	21	4	4	4	29	6	6	7
12	19	3	3	3	28	4	4	6
13	18	3	3	3				
14	12	3	3	3				

are shown in Table 3 for each iteration k of the barrier algorithm. For simplicity the partition $A = (N \quad B)$ was chosen to be the same for all k, with B being the optimal nondegenerate basis determined by the simplex method. As expected, the preconditioners M_2–M_4 improve markedly as the LP solution is approached.

Iterative solution of each KKT system is easier for the scaled problem because $B^T = LU$ is better conditioned:

	Scaled	Unscaled
cond(B)	14	443
cond(L)	4	4
cond(U)	17	235

For the scaled problem the stopping tolerance for SYMMLQ was taken to be $rtol = 10^{-6}$ (a loose value since the KKT systems need not be solved accurately). However, *rtol* terminates solution of the *preconditioned* system. For the unscaled problem it was necessary to set $rtol = 10^{-10}$ to obtain sufficient accuracy in the search direction for the first few values of k. In general it seems that high precision would be needed for safety: $rtol \approx 10^{-15}$. (This appears to be a general difficulty. If the original system is $Kx = b$ and the preconditioner is CC^T, SYMMLQ terminates when $\|C^{-1}(b - Kx)\| \leq \|C^{-1}KC^{-T}\|\|C^Tx\| rtol$, since the terms involved can be estimated. There is no certainty that $\|b - Kx\| \leq \|K\|\|x\| rtol$, although $\|b - Kx\| \leq \|b\| rtol$ could be tested after the fact.)

On nondegenerate problems such as this, preconditioners M_2–M_4 can be expected to perform similarly, at least in the scaled case. We expected M_4 to show an advantage in the unscaled case, but this did not eventuate. Greater variation can be expected on degenerate problems when V does not become suitably small. Further experiments in this direction remain for the future.

7. Conclusions. For symmetric indefinite systems of linear equations, we have shown that the Bunch–Parlett factorization can be used to provide a preconditioner for the Paige–Saunders algorithm SYMMLQ (§2). This general result led us to consider iterative methods for the KKT systems arising in barrier methods for QP and

nonlinear programming. The preconditioner (12) should play an important role in future interior-point implementations for large-scale constrained optimization.

For linear programs, the sensitivity analysis associated with the partitioned KKT system (10) led us to consider the true sensitivity of K, as reflected by the preconditioner M_1 and the transformed system $\bar{K}_1$ (13), (14). In turn, the fact that $\text{cond}(\bar{K}_1)$ is typically much smaller than $\text{cond}(AD^2A^T)$ motivated development of the preconditioners M_2, M_3, M_4 (15), (17), (19).

Subject to effective methods for choosing B, we expect these KKT preconditioners to bring improved reliability to interior-point LP algorithms. Implementations based on direct or iterative solves with AD^2A^T are often remarkably effective, but the extreme ill-conditioning of the AD^2A^T systems as solutions are approached makes their use tantamount to walking the razor's edge.

A switch to the full KKT system should be beneficial as Gay [Gay89] suggests, particularly when A contains some relatively dense columns that prevent exact Cholesky factorization of AD^2A^T. Fortunately, since B becomes more sharply defined near a solution, the KKT preconditioners will become most effective when they are most needed.

Appendix: A preconditioner from the Bunch–Parlett factorization. The following Fortran 77 routine illustrates the construction of a positive-definite matrix $M = U^T\bar{D}U$ from the Bunch–Parlett factorization $A = U^TDU$ produced by the Harwell MA27 package of Duff and Reid [DR82], [DR83].

Subroutine syprec overwrites the representation of D in the MA27 data structure. A typical application would contain calls of the form

```
call ma27ad( n, nz, ... )
call ma27bd( n, nz, ... )
call syprec( n, la, ... )
```

to factorize A and compute $\bar{D}$, followed by multiple calls of the form

```
call ma27cd( n, a , ... )
```

to solve systems involving M.

```
*    ------------------------------------------------------------------
     subroutine syprec( n, la, liw, a, iw, neg1, neg2 )

     implicit           double precision ( a-h, o-z )
     double precision   a(la)
     integer*2          iw(liw)
     integer            neg1, neg2

*    ------------------------------------------------------------------
*    syprec (SYmmetric PREConditioner) takes the factors
*          A = U' D U
*    from Duff and Reid's Harwell subroutine MA27BD and changes the
*    block-diagonal matrix D to be a positive-definite matrix Dbar with
*    the same 1x1 and 2x2 block-diagonal structure.
*
*    The eigensystem D = Q E Q' is used to define Dbar = Q |E| Q',
*    where |E| contains the absolute values of the eigenvalues of D.
*    The matrix
*          Abar = U' Dbar U
*    is then an exact preconditioner for A, in the sense that SYMMLQ
*    would take only 2 iterations to solve Ax = b (or 1 iteration if
```

```
*     D = Dbar is already positive definite).
*
*     If the original matrix A is close to some other matrix K,
*     Abar should be a good preconditioner for solving K x = b.
*
*     Note that MA27 stores the elements of D(inverse) and ( - U )
*     within A and IW.  However, modifying a 2x2 block of D(inverse)
*     looks the same as modifying the 2x2 block itself.
*
*     10 Mar 1989: First version.
*     Systems Optimization Laboratory, Stanford University.
*    ------------------------------------------------------------------

      intrinsic          abs , sqrt
      integer            alen, apos
      logical            single
      parameter        ( zero = 0.0d+0,  one = 1.0d+0,  two = 2.0d+0 )

      neg1   = 0
      neg2   = 0
      nblk   = abs( iw(1) )
      ipos   = 2
      apos   = 1

      do 100, iblk = 1, nblk
         ncols     =   iw(ipos)

         if (ncols .lt. 0) then
            nrows  =   1
            ncols  = - ncols
         else
            ipos   =   ipos + 1
            nrows  =   iw(ipos)
         end if

*        Process the diagonals in this block.

         alen   =   ncols
         single =  .true.

         do 50, k = ipos + 1, ipos + nrows
            if ( single ) then
               alpha  =   a(apos)
               j      =   iw(k)
               single =   j .gt. 0

               if ( single ) then
                  if ( alpha .lt. zero ) then
*                    ---------------------------------------------
*                    The 1x1 diagonal is negative.
*                    ---------------------------------------------
                     neg1    =   neg1 + 1
                     a(apos) = - alpha
                  end if
```

```
               else
                  beta   =   a(apos+1   )
                  gamma  =   a(apos+alen)

                  if ( alpha * gamma  .lt.  beta**2 ) then
*                    ------------------------------------------------
*                    The 2x2 diagonal is indefinite.
*                    Find its eigensystem in the form
*
*                    ( alpha  beta  )  =  ( c   s ) ( e1    ) ( c  s )
*                    ( beta   gamma )     ( s  -c ) (    e2 ) ( s -c )
*                    ------------------------------------------------
                     tau    =   ( gamma - alpha ) / ( two * beta )
                     t      =   abs( tau )  +  sqrt( tau**2 + one )
                     t      = - one / t
                     if ( tau .lt. zero ) t = - t
                     c      =   one / sqrt( t**2 + one )
                     s      =   t * c
                     e1     =   alpha  +  beta * t
                     e2     =   gamma  -  beta * t

*                    Change e1 and e2 to their absolute values
*                    and then multiply the three 2x2 matrices
*                    to get the modified  alpha, beta and gamma.


                     if ( e1 .lt. zero ) then
                        neg2   =   neg2 + 1
                        e1     = - e1
                     end if
                     if ( e2 .lt. zero ) then
                        neg2   =   neg2 + 1
                        e2     = - e2
                     end if

                     alpha        =   c**2 * e1  +  s**2 * e2
                     beta         =   c*s  *(e1  -  e2)
                     gamma        =   s**2 * e1  +  c**2 * e2
                     a(apos     ) =   alpha
                     a(apos+1   ) =   beta
                     a(apos+alen) =   gamma
                  end if
               end if
            else
               single =  .true.
            end if

            apos   =   apos + alen
            alen   =   alen - 1
   50    continue

         ipos   =   ipos + ncols + 1
  100 continue
```

```
*     end of syprec
      end
```

Acknowledgments. We thank Dr. Ed Klotz for making the test problem *exp1* available to us, and Gerry Miranda for his help in translating SYMMLQ into MATLAB. We also thank the referees for their helpful suggestions.

REFERENCES

[Aas71] J. O. AASEN, *On the reduction of a symmetric matrix to tridiagonal form*, BIT, 11 (1971), pp. 233–242.

[ADR89] M. ARIOLI, I. S. DUFF, AND P. P. M. DE RIJK, *On the augmented system approach to sparse least-squares problems*, Numer. Math., 55 (1989), pp. 667–684.

[Big75] M. C. BIGGS, *Constrained minimization using recursive quadratic programming: Some alternative subproblem formulations*, in Towards Global Optimization, L. C. W. Dixon and G. P. Szegö, eds., North–Holland, Amsterdam, 1975, pp. 341–349.

[Bjo67] Å. BJÖRCK, *Iterative refinement of linear least squares solutions*, BIT, 7 (1967), pp. 257–278.

[Bjo91] ———, *Pivoting and stability in the augmented system method*, Report LiTH-MAT-R-1991-30, Department of Mathematics, Linköping University, Linköping, Sweden, 1991.

[Bla82] C. BLAIR, *Some linear programs requiring many pivots*, Faculty Working Paper No. 867, College of Commerce and Business Administration, University of Illinois, Urbana, IL, 1982.

[BK77] J. R. BUNCH AND L. KAUFMAN, *Some stable methods for calculating inertia and solving symmetric linear systems*, Math. Comp., 31 (1977), pp. 162–179.

[BP71] J. R. BUNCH AND B. N. PARLETT, *Direct methods for solving symmetric indefinite systems of linear equations*, SIAM J. Numer. Anal., 8 (1971), pp. 639–655.

[Bur88] R. C. BURCHETT, Power Engineering Associates, private communication, 1988.

[Bur89] ———, Power Engineering Associates, private communication, 1989.

[DES82] R. S. DEMBO, S. C. EISENSTAT, AND T. STEIHAUG, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.

[DG85] J. J. DONGARRA AND E. GROSSE, *Distribution of mathematical software via electronic mail*, SIGNUM Newsletter, 20 (1985), pp. 45–47.

[DGRST89] I. S. DUFF, N. I. M. GOULD, J. K. REID, J. A. SCOTT, AND K. TURNER, *The factorization of sparse symmetric indefinite matrices*, CSS Report 236, Computer Science and Systems Division, AERE Harwell, Oxford, England, 1989.

[DR82] I. S. DUFF AND J. K. REID, MA27: *A set of Fortran subroutines for solving sparse symmetric sets of linear equations*, Report R-10533, Computer Science and Systems Division, AERE Harwell, Oxford, England, 1982.

[DR83] ———, *The multifrontal solution of indefinite sparse symmetric linear equations*, ACM Trans. Math. Software, 9 (1983), pp. 302–325.

[FM91] R. FOURER AND S. MEHROTRA, *Performance of an augmented system approach for solving least-squares problems in an interior-point method for linear programming*, Mathematical Programming Society COAL Newsletter, 19 (1991), pp. 26–31.

[Gay85] D. M. GAY, *Electronic mail distribution of linear programming test problems*, Mathematical Programming Society COAL Newsletter, December 1985.

[Gay89] ———, *Stopping tests that compute optimal solutions for interior-point linear programming algorithms*, Numerical Analysis Manuscript 89-11, AT&T Bell Laboratories, Murray Hill, NJ, 1989.

[GMS89] P. E. GILL, W. MURRAY, AND M. A. SAUNDERS, *A dual barrier method for linear programming using* LU *preconditioning*, presented at the SIAM Annual Meeting, San Diego, CA, 1989.

[GMSTW86] P. E. GILL, W. MURRAY, M. A. SAUNDERS, J. A. TOMLIN, AND M. H. WRIGHT, *On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method*, Math. Programming, 36 (1986), pp. 183–209.

[GMSW87] P. E. GILL, W. MURRAY, M. A. SAUNDERS, AND M. H. WRIGHT, *Maintaining* LU *factors of a general sparse matrix*, Linear Algebra Appl., 88/89 (1987), pp. 239–270.

[GV89] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1989.

[Gou86] N. I. M. GOULD, *On the accurate determination of search directions for simple differentiable penalty functions*, IMA J. Numer. Anal., 6 (1986), pp. 357–372.

[Kar87] N. K. KARMARKAR, *Recent developments in new approaches to linear programming*, presented at the SIAM Conference on Optimization, Houston, TX, 1987.

[KR88] N. K. KARMARKAR AND K. G. RAMAKRISHNAN, *Implementation and computational results of the Karmarkar algorithm for linear programming, using an iterative method for computing projections*, Extended abstract, presented at the 13th International Symposium on Mathematical Programming, Tokyo, Japan, 1988.

[KMY88] M. KOJIMA, S. MIZUNO, AND A. YOSHISE, *A primal-dual interior-point algorithm for linear programming*, in Progress in Mathematical Programming, N. Megiddo, ed., Springer-Verlag, New York, 1988, pp. 29–48.

[LMS89] I. J. LUSTIG, R. E. MARSTEN, AND D. F. SHANNO, *Computational experience with a primal-dual interior-point method for linear programming*, Report SOR 89-17, Department of Civil Engineering and Operations Research, Princeton University, Princeton, NJ, 1989.

[MMS89] K. A. MCSHANE, C. L. MONMA, AND D. F. SHANNO, *An implementation of a primal-dual interior point method for linear programming*, ORSA J. Comput., 1 (1989), pp. 70–83.

[Meg86] N. MEGIDDO, *Pathways to the optimal set in linear programming*, in Progress in Mathematical Programming, N. Megiddo, ed., Springer-Verlag, New York, 1986, pp. 131–158.

[Meh89a] S. MEHROTRA, *Implementations of affine scaling methods: Approximate solutions of systems of linear equations using preconditioned conjugate-gradient methods*, Report 89-04, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, 1989.

[Meh89b] ———, *On finding a vertex solution using interior-point methods*, Report 89-22, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, 1989.

[Meh90] ———, *On the implementation of a primal-dual interior-point method*, Report 90-03, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, 1990.

[MLB87] C. MOLER, J. LITTLE, AND S. BANGERT, *PRO-MATLAB User's Guide*, The MathWorks, Inc., Sherborn, MA, 1987.

[Mur69] W. MURRAY, *Constrained Optimization*, Ph.D. thesis, Computer Science Department, University of London, London, England, 1969.

[PS75] C. C. PAIGE AND M. A. SAUNDERS, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12, (1975), pp. 617–629.

[Pon90] D. B. PONCELEÓN, *Barrier methods for large-scale quadratic programming*, Ph.D. thesis, Computer Science Department, Stanford University, Stanford, CA, 1990.

[Rut66] H. RUTISHAUSER, *The Jacobi method for real symmetric matrices*, Numer. Math., 9 (1966), pp. 1–10.

[Sau79] M. A. SAUNDERS, *Fortran code: A modification of SYMMLQ to allow use of a positive-definite preconditioner*, 1979; Included in *misc* chapter of *netlib*, September 1985.

[SW78] D. B. SZYLD AND O. B. WIDLUND, *Fortran code: A modification of SYMMLQ to allow use of a positive-definite preconditioner*, private communication, 1978.

[SW79] ———, *Applications of conjugate-gradient-type methods to eigenvalue calculations*, in Advances in Computer Methods for Partial Differential Equations III, R. Vichnevetsky and R. S. Stepleman, eds., IMACS, 1979, pp. 167–173.

[TZ89] R. A. TAPIA AND Y. ZHANG, *A fast optimal basis identification technique for interior point linear programming methods*, Report TR89-1, Department of Mathematical Sciences, Rice University, Houston, TX, 1989.

[Tur87] K. TURNER, *Computational experience with the projective Karmarkar algorithm*, presented at ORSA/TIMS Joint National Meeting, St. Louis, MO, 1987.

[Tur90] ———, *Computing projections for the Karmarkar algorithm*, Report 49, Department of Mathematics and Statistics, Utah State University, Logan, UT, 1990.