SIAM J. Matrix Anal. Appl.
Vol. 17, No. 1, pp. 47–68, January 1996

003

PRECONDITIONING REDUCED MATRICES*

STEPHEN G. NASH† AND ARIELA SOFER‡

Abstract. We study preconditioning strategies for linear systems with positive-definite matrices of the form Z^TGZ, where Z is rectangular and G is symmetric but not necessarily positive definite. The preconditioning strategies are designed to be used in the context of a conjugate-gradient iteration, and are suitable within algorithms for constrained optimization problems. The techniques have other uses, however, and are applied here to a class of problems in the calculus of variations. Numerical tests are also included.

Key words. preconditioning, conjugate-gradient method, reduced Hessian, nonlinear programming

AMS(MOS) subject classifications. 65F10, 90C06, 90C30

1. Introduction.

We are interested in solving linear systems of the form

$$Z^TGZp = d \tag{1.1}$$

via the preconditioned conjugate-gradient method. The matrix Z^TGZ is assumed to be positive definite, although G need not be. Our primary concern is the choice of a preconditioner for this system.

We intend to apply the techniques within algorithms for solving large nonlinear optimization problems:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g(x) = 0, \quad h(x) \leq 0, \end{array}$$

where $f(x)$, $g(x)$, and $h(x)$ are nonlinear functions ($f(x)$ is scalar valued, $g(x)$ and $h(x)$ are vector valued). Many algorithms for this problem solve a sequence of linear systems of the form (1.1). This is true for active set methods [6] as well as stabilized penalty methods [13]. It is also true for sequential quadratic programming algorithms [6]. The matrix $G = G(x)$ may represent the Hessian of f at the point x, or it may represent the Hessian of the corresponding Lagrangian function. The matrix Z is a basis for the tangent subspace defined by the active constraints at x. When the number of variables is large, it is often appropriate to apply an iterative method to (1.1), such as a truncated-Newton method [2].

As the solution to the optimization problem is approached, the optimality conditions guarantee that Z^TGZ will be positive semi-definite. In non-degenerate cases, Z^TGZ will be positive definite in a neighborhood of the solution. It is always the case that Z^TGZ will be symmetric. For these reasons, the conjugate-gradient method is normally used to solve (1.1), with safeguards in case Z^TGZ is not positive definite [2, 10].

*Received by the editors March 10, 1993; accepted for publication (in revised form) by L. Kaufman January 20, 1995.

†Operations Research and Engineering Department, George Mason University, Fairfax, VA 22030. The work of this author was supported by National Science Foundation grant DDM–9104670.

‡Operations Research and Engineering Department, George Mason University, Fairfax, VA 22030. The work of this author was supported by National Science Foundation grant DDM–9104670.

As a simple special case, consider a quadratic programming problem of the form

$$\begin{aligned} &\text{minimize}_x \quad f(x) = \tfrac{1}{2}x^TGx - c^Tx \\ &\text{subject to} \quad Ax = b, \end{aligned}$$

where A is an $m \times n$ matrix with $m < n$, and G is positive definite on the null space of A. The first-order optimality conditions for the problem are

$$\begin{pmatrix} G & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} c \\ b \end{pmatrix}. \tag{1.2}$$

(The vector λ is the vector of Lagrange multipliers for the constraints.) Then $x = x_0 + Zp$ where p is the solution to

$$Z^TGZp = Z^T(c - Gx_0),$$

x_0 is any solution to the constraints $Ax = b$, and Z is a matrix whose columns form a basis for the null space of A (i.e., $AZ = 0$). Auxiliary calculations are needed to compute λ.

In this simple case the solution is obtained by solving a single system of the form (1.1). In more complicated cases a sequence of systems of this form must be solved, with perhaps both G and Z changing from one system to the next. The null-space matrix Z can change not only in its entries, but also in its size as the dimension of the relevant null-space changes.

Some optimization algorithms work directly with the linear system (1.2). This system is symmetric but not positive definite, and so the traditional conjugate-gradient method cannot be applied. It is also a larger system than (1.1), having $n+m$ variables instead of $n - m$. Preconditioning strategies for (1.2) are discussed in [5, 14]. We concentrate on the solution of (1.1).

In exact arithmetic, the number of iterations required by the conjugate gradient method is bounded by the number of distinct eigenvalues of Z^TGZ. In addition, from iteration to iteration the method displays a linear rate of convergence with rate constant $(\sqrt{\kappa} - 1)/(\sqrt{\kappa} + 1)$, where κ is the condition number of Z^TGZ [7].

A preconditioner is a (symmetric) positive-definite matrix K such that $K \approx Z^TGZ$. System (1.1) is equivalent to the *preconditioned* system

$$K^{-1}Z^TGZp = K^{-1}d.$$

The goal is to choose K so that the preconditioned matrix $K^{-1}Z^TGZ$ will have a smaller number of distinct eigenvalues, or a smaller condition number, or both. For practical purposes, it must be possible to solve linear systems of the form $Ky = v$ for arbitrary v.

If an approximation $K \approx Z^TGZ$ is provided then the situation is straightforward, since the approximation could be used directly within the preconditioned conjugate-gradient method. In the optimization setting, however, we think it more likely that only an approximation $M \approx G$ would be available. Such a matrix M might be generated by the optimization algorithm itself, using for example the techniques in [11]. Or it might be suggested by the optimization problem. For example, if the objective function $f(x)$ were derived from a differential equation, an approximation to the corresponding differential operator G might be available, such as a fast Poisson

solver. It seems less likely that an approximation to the reduced matrix Z^TGZ would be provided, particularly in cases where Z changes frequently.

In some cases, explicitly forming Z^TGZ or Z^TMZ will be undesirable. In many large problems the matrices G and A will be sparse or have special structure, and the special structure of A will often carry over to Z. Forming the reduced matrices can destroy this structure. For example, if $Ax = b$ corresponds to the simple constraint

$$\sum x_i = 1,$$

and an orthogonal null-space matrix Z is used, then Z^TGZ will be dense even if G is sparse.

For these reasons we are interested in constructing $K \approx Z^TGZ$ given Z and a (symmetric) positive-definite matrix $M \approx G$. (Z is used within the optimization algorithm and so is available.) We describe a variety of approximations K that could be used, even though some of them are likely to be too computationally expensive for routine use. We also discuss the computation of Z and its influence on the conditioning of the reduced matrices. Finally, we apply the ideas to a class of problems arising in the calculus of variations. Numerical tests are presented to illustrate the techniques.

The simplest of the preconditioners that we derive is

$$K^{-1} = W^TM^{-1}W \approx (Z^TGZ)^{-1},$$

where W^T is a left inverse for Z (i.e., $W^TZ = I$). In many cases, W can be obtained as a by-product of the calculations used to obtain Z. The computational effort required to apply the preconditioner is that required for the "unreduced" preconditioner M, plus that for W and W^T. The numerical tests in §8 indicate that the formula can be an effective preconditioner for reduced systems. It is this preconditioner that we would recommend for most applications. Hence, within the conjugate-gradient method, applying the preconditioner to a vector v would mean calculating $K^{-1}v = W^TM^{-1}Wv$.

The more elaborate preconditioners that we derive require considerably more computational effort to use, but they can further accelerate the convergence of the conjugate-gradient method. Whether these preconditioners would be appropriate would depend on the relative computational costs of applying the preconditioner (forming $K^{-1}v$) and computing a matrix-vector product (forming Z^TGZv). In cases where the latter is expensive (see below), the more elaborate preconditioners might be worthwhile.

In truncated-Newton software, it is common to compute a Hessian-vector product via finite differencing. In this case $G = \nabla^2 f(x)$ for some nonlinear function $f(x)$, and

$$Z^TGZv = Z^T\nabla^2 f(x)Zv \approx Z^T\frac{\nabla f(x + hZv) - \nabla f(x)}{h}$$

for some "small" value of h. Thus the cost of the matrix-vector product is proportional to the cost of evaluating the gradient $\nabla f(x)$. If evaluating the gradient is expensive, then the matrix-vector product will be expensive, and could easily become the dominant part of the conjugate-gradient iteration. In such cases we envision the more elaborate preconditioners being used.

Here is an outline of the paper. Basic topics are in Section 2. A family of preconditioners is derived in Section 3. They are based on an infinite series, whose

convergence is the topic of Section 4. Section 5 extends the results to the case where Z is a projection matrix that is not of full rank. Section 6 focuses on a specific choice of Z commonly used in large-scale optimization. Section 7 shows how the techniques can be applied to a class of problems in the calculus of variations. Numerical tests are in Section 8 and conclusions in Section 9.

2. Basics. For simplicity we consider the simple case (1.2), treating it if appropriate as a single instance of a sequence of linear systems of the same structure. Hence A is an $m \times n$ matrix with $m < n$ corresponding to the (perhaps linearized) constraints. (If $m = n$ then the solution is determined entirely by the constraints.) For convenience we assume that $\text{rank}(A) = m$. As before, Z is a matrix whose columns form a basis for the null space of A, so that $AZ = 0$ and $\text{rank}(Z) = n - m$.

There are two traditional ways of forming Z. The first uses an orthogonal factorization of A^T:

$$A^T = QR \equiv \begin{pmatrix} Y & Z \end{pmatrix} \begin{pmatrix} R_1 \\ 0 \end{pmatrix},$$

where R_1 is an $m \times m$ upper triangular matrix. Then $Z^TZ = I$. This technique is often used on small problems where dense-matrix methods are appropriate.

The second technique (called *variable reduction*) identifies a subset of the variables of size m, called a basis. If the first m variables were to be used we would write

$$A = \begin{pmatrix} B & N \end{pmatrix}$$

where B is nonsingular. The corresponding matrix Z is given by

$$Z = \begin{pmatrix} -B^{-1}N \\ I \end{pmatrix}.$$

This only requires a factorization of the submatrix B, and is better suited to large sparse problems. It is easily checked that $AZ = 0$, but that $Z^TZ \neq I$ unless $N = 0$.

In our formulas we will require a left-inverse for Z, i.e., a matrix W^T satisfying $W^TZ = I$. If Z is available, a left-inverse for Z is usually available at little or no additional cost. For example, if Z is formed via an orthogonal factorization of A^T, we can choose $W^T = Z^T$. If Z is computed via the variable reduction method, we can choose $W^T = (0, I)$.

2.1. Some Lemmas. Not a great deal can be said in general about the relationship between G and Z^TGZ, so only limited conclusions can be drawn about the quality of the preconditioners we derive. However, the following lemmas provide some information. In the discussion that follows, $\|\cdot\| = \|\cdot\|_2$.

The first lemma discusses the case where G and M share an eigenvalue-vector pair. Note that $M^{-1}G$ then has an eigenvalue equal to one. The lemma shows that spectral information for the original matrix can be used in constructing a preconditioner for the reduced matrix.

LEMMA 1. *Suppose that* $Gv = \lambda v$ *and* $Mv = \lambda v$, *where* $v \neq 0$. *If* $v \in range(Z)$ *then*

$$(Z^TGZ)w = \lambda w \quad \textit{and} \quad (Z^TMZ)w = \lambda w,$$

where $Zw = v$.

The next result gives a bound on the norm of a reduced matrix.

LEMMA 2. *Let* H *be an* $n \times n$ *matrix. Then* $\|Z^THZ\| \leq \|H\| \cdot \|Z\|^2$.

The lemma may be used to bound the norm of the difference between the reduced matrix and its approximation:

$$\|(Z^TMZ) - (Z^TGZ)\| = \|Z^T(M-G)Z\| \leq \|M-G\| \cdot \|Z\|^2.$$

In particular, if Z is an orthogonal matrix the bound becomes

$$\|(Z^TMZ) - (Z^TGZ)\| \leq \|M-G\|.$$

When Z is an orthogonal matrix, we also have the following interlacing property (see [7]).

LEMMA 3. *Let Z be an $n \times l$ orthogonal matrix and G an $n \times n$ symmetric matrix with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$. Let $\omega_1 \geq \cdots \geq \omega_l$ be the eigenvalues of Z^TGZ. Then*

$$\lambda_i \geq \omega_i \geq \lambda_{i+n-l}, \quad i = 1, \ldots, l.$$

An immediate conclusion is that orthogonal reduction will not increase the condition number of a positive-definite matrix. This is stated in the next lemma.

LEMMA 4. *Let Z be an orthogonal matrix, and let G be a positive definite matrix. Then $cond(Z^TGZ) \leq cond(G)$.*

Unfortunately, the bounds provided in the above lemmas can be tight. For example, consider Lemma 4 in the case where G is a diagonal matrix and Z corresponds to a set of bound constraints. Then Z^TGZ is just a principal submatrix of G. If that submatrix includes the extreme eigenvalues of G then $\text{cond}(G) = \text{cond}(Z^TGZ)$. Therefore, it is not possible to make the results more precise. Any more specific result would require precise information about the particular submatrix that had been chosen. Similar pessimistic examples can be found for the other lemmas.

3. Formulas for Preconditioners. From a theoretical standpoint, it would be ideal to precondition Z^TGZ using Z^TGZ itself. If G is nonsingular, then from the identity (see [3, p. 87])

$$Z(Z^TGZ)^{-1}Z^T = G^{-1} - G^{-1}A^T(AG^{-1}A^T)^{-1}AG^{-1}$$

it follows that the inverse of Z^TGZ satisfies the identity

$$(Z^TGZ)^{-1} = W^TG^{-1}W - W^TG^{-1}A^T(AG^{-1}A^T)^{-1}AG^{-1}W$$

where W^T is a left-inverse for Z.

In cases where it is inconvenient to use G explicitly (or if G is singular), and a positive-definite preconditioner $M \approx G$ is given, it is natural to consider using Z^TMZ as a preconditioner for Z^TGZ. Linear systems of the form $Z^TMZy = v$ can be solved using the formula

$$(Z^TMZ)^{-1} = W^TM^{-1}W - W^TM^{-1}A^T(AM^{-1}A^T)^{-1}AM^{-1}W. \tag{3.1}$$

We assume that linear systems of the form $My = v$ can be easily solved. A preconditioner for the conjugate-gradient method must be positive definite. This will be true for the preconditioners that we derive because M is positive definite.

The preconditioners we consider are based on approximations to the right-hand side of (3.1). The accuracy of the approximation can be varied, but the more accurate the approximation, the more computations are required to implement it.

In certain cases it is possible to use (3.1) in its original form. For example, if m is small then $AM^{-1}A^T$ can be formed explicitly and then either factored or inverted. (In the case where $n-m$ is small and G is available, Z^TGZ can be formed and there is no need to use an iterative method to solve (1.1).) Even if the component parts of the right-hand side of (3.1) can be formed, it requires two applications of M^{-1} as well as an application of $(AM^{-1}A^T)^{-1}$ to use it:

$$(Z^TMZ)^{-1}v = W^TM^{-1}(I - A^T(AM^{-1}A^T)^{-1}AM^{-1})Wv,$$

making it more than twice as expensive to precondition Z^TGZ as to precondition G.

If neither m nor $n-m$ is small, then some approximation to (3.1) must be used. The simplest available is

$$(Z^TMZ)^{-1} \approx W^TM^{-1}W. \tag{3.2}$$

This approximation requires only one application of M^{-1} and so is economical. Since

$$(Z^TMZ)^{-1} - W^TM^{-1}W = -W^TM^{-1}A^T(AM^{-1}A^T)^{-1}AM^{-1}W,$$

there is no guarantee that $\|(Z^TMZ)^{-1} - (W^TM^{-1}W)\|$ is small. However if either m or $n-m$ is small then the difference is of low rank and (3.2) may be satisfactory. (If the difference between the matrix and the preconditioner is of rank l then the preconditioned conjugate-gradient method will converge in at most $l+1$ iterations with exact arithmetic.)

The preconditioned matrix is

$$(W^TM^{-1}W)(Z^TMZ) = W^T(M^{-1}WZ^TM)Z.$$

The eigenvalues of the inner matrix $(M^{-1}WZ^TM)$ are all either zero or one, since the corresponding eigenvectors are the columns of $M^{-1}W$ and $M^{-1}A^T$:

$$\begin{aligned}(M^{-1}WZ^TM)(M^{-1}W) &= M^{-1}W,\\ (M^{-1}WZ^TM)(M^{-1}A^T) &= 0.\end{aligned}$$

This suggests that the preconditioned matrix may be well-suited for the conjugate-gradient method, since the convergence of this method depends on the number of distinct eigenvalues of the matrix.

More sophisticated (and more expensive) preconditioners can be obtained by approximating the inverse of the matrix (Z^TMZ). They are based on the power series expansion

$$\begin{aligned}X^{-1} &= [Y-(Y-X)]^{-1}\\ &= Y^{-1}[I+(I-XY^{-1})+(I-XY^{-1})^2+\cdots]\end{aligned} \tag{3.3}$$

where $Y \approx X$. The series converges if the eigenvalues of $(I-XY^{-1})$ are less than one in absolute value.

If we let

$$X = Z^TMZ \quad \text{and} \quad Y^{-1} = W^TM^{-1}W$$

in (3.3), then using the first k terms of the series gives the approximation

$$(Z^TMZ)^{-1} \approx (W^TM^{-1}W)\sum_{j=0}^{k}(I-T)^j\,, \tag{3.4}$$

where $T = (Z^T M Z)(W^T M^{-1} W)$. (We assume here that the series on the right converges as $k \to \infty$; see §4.) Taking for example $k = 0$, we obtain the "inverse preconditioner" (3.2). (Formula (3.4) provides the *inverse* of the preconditioner; all our formulas will be of this nature, and the preconditioners themselves will not be specified.) Taking $k = 1$ we obtain the inverse preconditioner

$$(Z^T M Z)^{-1} \approx (W^T M^{-1} W)\,(2I - (Z^T M Z)(W^T M^{-1} W))\,.$$

Since the inverse preconditioner need not be formed explicitly, it is available at no expense (once Z and W are available). The expense lies in applying it to a vector:

$$(Z^T M Z)^{-1} v \approx (W^T M^{-1} W)\,(2I - (W^T M^{-1} W)(Z^T M Z))\,v.$$

We can derive an alternative expression for the inverse preconditioner that is more efficient for computation. Using the relationship

$$\sum_{j=0}^{k} (I - T)^j = \sum_{j=0}^{k} (-1)^j \tbinom{k+1}{j+1} T^j \tag{3.5}$$

we obtain

$$(Z^T M Z)^{-1} \approx (W^T M^{-1} W) \sum_{j=0}^{k} (-1)^j \tbinom{k+1}{j+1} T^j, \tag{3.6}$$

where as before, $T = (Z^T M Z)(W^T M^{-1} W)$. This is the form of the inverse preconditioner used in our computations. However in the following sections we continue to use the representation given in (3.4), since it is more convenient for mathematical manipulation.

3.1. An Alternative Formula. It might appear that the power series expansion could be used in another way to obtain additional inverse preconditioners. If $X = A M^{-1} A^T$ and R is a matrix such that $RA^T = I$ then a "plausible" choice for an approximate inverse of X is

$$Y^{-1} = R M R^T.$$

Applying the power series to approximate $(A M^{-1} A^T)^{-1}$ on the right-hand side of (3.1) we obtain

$$(Z^T M Z)^{-1} \approx W^T M^{-1} W - W^T M^{-1} A^T Y^{-1} \Big(\sum_{j=0}^{k-1} (I - U)^j \Big) A M^{-1} W, \tag{3.7}$$

where $U = XY^{-1} = (A M A^T)(R M^{-1} R^T)$. As an illustration, if only one term in the power series is taken, we obtain the inverse preconditioner

$$(Z^T M Z)^{-1} \approx W^T M^{-1} [I - A^T Y^{-1} A M^{-1}] W.$$

A left inverse for A^T is usually available from the computation of Z at little or no additional cost. For example if Z is obtained from an orthogonal factorization of A^T then the left-inverse matrix $R = (A A^T)^{-1} A$ can be computed easily from the factors of the orthogonalization.

The following lemma shows that the inverse preconditioners (3.4) and (3.7) are mathematically identical, and so no new preconditioners are obtained—just an alternative formula. In some applications one formula may be easier to apply than the other. (See also §4.)

The lemma assumes that $A^TR + WZ^T = I$. This assumption ensures that R and W^T are chosen in a "consistent" fashion. It is satisfied when Z is computed via an orthogonal factorization of A, $R = (AA^T)^{-1}A$, and $W = Z$. It is also satisfied when Z is computed via the variable reduction method, with

$$Z = \begin{pmatrix} -B^{-1}N \\ I \end{pmatrix}, \quad R = \begin{pmatrix} B^{-T} & 0 \end{pmatrix}, \quad W = \begin{pmatrix} 0 \\ I \end{pmatrix}.$$

(In general, given a full-rank matrix Z that is a null-space matrix for A, and a matrix R that is a left-inverse for A^T, there exists a unique left-inverse matrix W^T for Z such that $A^TR + WZ^T = I$.)

LEMMA 5. *Let $RA^T = I$ $W^TZ = I$, and suppose that $A^TR + WZ^T = I$. Let $Y^{-1} = RMR^T$, $U = (AM^{-1}A^T)(RMR^T)$ and $T = (Z^TMZ)(W^TM^{-1}W)$. Then for $k \geq 1$,*

$$W^TM^{-1}W - W^TM^{-1}A^TY^{-1}\Big(\sum_{j=0}^{k-1}(I-U)^j\Big)AM^{-1}W = (W^TM^{-1}W)\sum_{j=0}^{k}(I-T)^j,$$

Proof. It is sufficient to prove that

$$-W^TM^{-1}A^TY^{-1}\left((I-U)^{k-1}\right)AM^{-1}W = (W^TM^{-1}W)(I-T)^k. \tag{3.8}$$

The proof is by induction. For $k = 1$ we have

$$\begin{aligned}
-W^TM^{-1}A^TY^{-1}&((I-U)^{k-1})AM^{-1}W \\
&= -W^TM^{-1}A^TY^{-1}AM^{-1}W \\
&= -W^TM^{-1}A^TRMR^TAM^{-1}W \\
&= -W^TM^{-1}(I-WZ^T)M(I-ZW^T)M^{-1}W \\
&= -W^TM^{-1}(W - WZ^TW - MZW^TM^{-1}W + WZ^TMZW^TM^{-1}W) \\
&= -W^TM^{-1}\left(-MZ(W^TM^{-1}W) + W(Z^TMZ)(W^TM^{-1}W)\right) \\
&= -W^TM^{-1}\left(-MZ + W(Z^TMZ)\right)(W^TM^{-1}W) \\
&= \left(I - (W^TM^{-1}W)(Z^TMZ)\right)(W^TM^{-1}W) \\
&= (W^TM^{-1}W)(I-T).
\end{aligned}$$

Assume now that (3.8) is true for $k-1$. We shall prove that it is also true for k. Note from the proof for $k = 1$ that

$$A^TY^{-1}AM^{-1}W = \left(-MZ + W(Z^TMZ)\right)(W^TM^{-1}W).$$

We conclude from this that

$$\begin{aligned}
UAM^{-1}W &= AM^{-1}A^TY^{-1}AM^{-1}W \\
&= AM^{-1}\left(-MZ + W(Z^TMZ)\right)(W^TM^{-1}W) \\
&= AM^{-1}W(Z^TMZ)(W^TM^{-1}W) \\
&= AM^{-1}WT.
\end{aligned}$$

Using this relation and the induction hypothesis we obtain

$$\begin{aligned}
&-W^TM^{-1}A^TY^{-1}\left((I-U)^{k-1}\right)AM^{-1}W \\
&\qquad = -W^TM^{-1}A^TY^{-1}\left((I-U)^{k-2}\right)\left(AM^{-1}W - UAM^{-1}W\right) \\
&\qquad = -W^TM^{-1}A^TY^{-1}\left((I-U)^{k-2}\right)\left(AM^{-1}W(I-T)\right) \\
&\qquad = (W^TM^{-1}W)(I-T)^{k-1}(I-T) \\
&\qquad = (W^TM^{-1}W)(I-T)^k.
\end{aligned}$$

■

There are several remaining questions, such as what to do if the series does not converge, how many terms to use, and which of the two equivalent formulas to use. Convergence is discussed in the next section, the choice of the number of terms to use is mentioned in §8 when we discuss computational results, and the particular formula to use would probably just be a matter of which were more convenient, perhaps depending on the relative magnitudes of m and $n-m$.

4. Guaranteeing Convergence. In developing the polynomial preconditioners, we tacitly assumed that the corresponding infinite series converge. We now explore the conditions for convergence of the series and discuss strategies to handle the case where the conditions are not met.

Consider the power series (3.3). It is convergent if the eigenvalues of $(I - XY^{-1})$ are less than 1 in absolute value, or equivalently if the eigenvalues of XY^{-1} lie strictly between 0 and 2. In our application, where X and Y^{-1} are positive definite, the eigenvalues of XY^{-1} will be positive, since XY^{-1} is similar to a positive definite matrix:

$$Y^{-\frac{1}{2}}(XY^{-1})Y^{\frac{1}{2}} = Y^{-\frac{1}{2}}(X)Y^{-\frac{1}{2}}.$$

Let $\lambda_1(\cdot)$ and $\lambda_n(\cdot)$ denote the largest and smallest eigenvalues of a matrix, respectively. Then for a positive scalar α such that $\alpha < (2/\lambda_1(XY^{-1}))$, the eigenvalues of αXY^{-1} will lie strictly between 0 and 2. In turn, the eigenvalues of $(I - \alpha XY^{-1})$ will be smaller than 1 in absolute value. We can now use the approximation

$$\begin{aligned}
X^{-1} &= \alpha(\alpha X)^{-1} = \alpha[Y - (Y - \alpha X)]^{-1} \\
&= \alpha Y^{-1}[I + (I - \alpha XY^{-1}) + (I - \alpha XY^{-1})^2 + \cdots].
\end{aligned}$$

The expression on the right is a convergent sequence. Notice that it differs from (3.3) only in that Y^{-1} is replaced by αY^{-1}. The rate of convergence of the sequence depends on the particular choice of the scaling parameter α; ideally the eigenvalues of αXY^{-1} should be close to 1. A possible choice is the scaling parameter that minimizes the largest deviation of the eigenvalues of the scaled matrix from 1. This gives $\alpha = 2/(\lambda_1(XY^{-1}) + \lambda_n(XY^{-1}))$.

Choosing $\alpha < 2/\lambda_1(XY^{-1})$ guarantees that the resulting preconditioner will be positive definite. To see this, denote by η_i the eigenvalues of $I - \alpha XY^{-1}$; the selection of α guarantees that $-1 < \eta_i < 1$. Then the eigenvalues of $\sum_{j=0}^{k}(I - XY^{-1})^j$ will be $\bar{\eta}_i = \sum_{j=0}^{k}\eta_i^j > 0$.

We now obtain the explicit form for the inverse preconditioner that includes the scaling parameter. Consider first the family of inverse preconditioners represented by (3.4), and denote as before

$$T = (W^TM^{-1}W)(Z^TMZ).$$

Let α be a scalar such that $\alpha < 2/\lambda_1(T)$. Applying the above results we obtain

$$(Z^T M Z)^{-1} \approx \alpha(W^T M^{-1} W) \sum_{j=0}^{k} (I - \alpha T)^j .$$

Alternatively, if we choose to use an inverse preconditioner of the form (3.7), we obtain

$$(Z^T M Z)^{-1} \approx W^T M^{-1} W - \alpha W^T M^{-1} A^T Y^{-1} \Big(\sum_{j=0}^{k-1} (I - \alpha U)^j \Big) A M^{-1} W,$$

where $U = (AM^{-1}A^T)(RMR^T)$ and $\alpha < 2/(\lambda_1(U)$.

In practice, the largest eigenvalues of U and T are not available. Our tests (§8) indicate that in many cases it is sufficient to use only the first term in the power series. There is then no need to compute the scaling parameter. In other cases we have to estimate the eigenvalues. (For the purpose of estimation it may be easier to use bounds on the norms of the matrices.) The following lemma shows that if R and W satisfy $A^T R + W Z^T = I$, then the eigenvalues of U and T are closely related.

LEMMA 6. *Suppose that $A^T R + W Z^T = I$. If $m \leq n - m$ then any non-unit eigenvalue of U is also an eigenvalue of T, and all other eigenvalues of T are equal to* 1. *If $n - m \leq m$ then any non-unit eigenvalue of T is also an eigenvalue of U, and all other eigenvalues of U are equal to* 1.

Proof. Let $E = AM^{-1}W$ and $F = Z^T M R^T$. Recalling that $AR^T = I$ we obtain

$$U = AM^{-1}A^T R M R^T = AM^{-1}(I - WZ^T)MR^T = I - EF.$$

Similarly, recalling that $Z^T W = I$ we obtain

$$T = Z^T M Z W^T M^{-1} W = Z^T M (I - R^T A) M^{-1} W = I - FE.$$

Suppose that $m \leq n - m$. We first show that any nonzero eigenvalue of EF is an eigenvalue for FE. Let λ and v be an eigenvalue/vector pair for EF with $\lambda \neq 0$. Then $EFv = \lambda v$, and since $\lambda \neq 0$, $Fv \neq 0$. Now $FE(Fv) = F\lambda v = \lambda(Fv)$. Hence λ is an eigenvalue for FE with associated eigenvector Fv.

Next we show that any eigenvalue of FE that is not an eigenvalue of EF must be zero. Suppose that $FEx = \mu x$ for some nonzero vector x. Then

$$EF(Ex) = E(FEx) = \mu(Ex).$$

The above relation can occur in one of three situations: (i) $Ex \neq 0$ and $\mu \neq 0$; in this case μ is a nonzero eigenvalue of EF. (ii) $Ex \neq 0$ but $\mu = 0$ (iii) $Ex = 0$; this implies that $\mu = 0$. The proof for the case $n - m \leq n$ is similar. ■

The lemma indicates that in estimating the scaling parameter one can use either U or T, whichever is more convenient. Denote the resulting inverse preconditioners by

$$K_1^{-1}(\alpha, k) = W^T M^{-1} W - \alpha W^T M^{-1} A^T Y^{-1} \Big(\sum_{j=0}^{k-1} (I - \alpha U)^j \Big) A M^{-1} W,$$

and

$$K_2^{-1}(\alpha, k) = \alpha(W^T M^{-1} W) \sum_{j=0}^{k} (I - \alpha T)^j .$$

The following result shows the relationship between K_1 and K_2. When $\alpha = 1$ they are equal (as pointed out in §3). Otherwise, assuming they converge, their difference goes to zero as k increases.

LEMMA 7. *Let $RA^T = I$ and $W^TZ = I$, and suppose that $A^TR + WZ^T = I$. Then for $k \geq 0$*

$$K_1^{-1}(\alpha, k) = K_2^{-1}(\alpha, k) + (1-\alpha)(W^TM^{-1}W)(I - \alpha T)^k.$$

Proof. Here we just sketch the proof. First, using arguments similar to those in Lemma 5, we can show that for $j \geq 0$,

$$-W^TM^{-1}A^TY^{-1}\left((I-\alpha U)^j\right)AM^{-1}W = (W^TM^{-1}W)(I-T)(I-\alpha T)^j.$$

Next, the relationship $\alpha(I-T) = (I-\alpha T) - (1-\alpha)I$ gives

$$\begin{aligned}
W^TM^{-1}W &- \alpha W^TM^{-1}A^TY^{-1}\Big(\sum_{j=0}^{k-1}(I-\alpha U)^j\Big)AM^{-1}W \\
&= W^TM^{-1}W + \alpha(W^TM^{-1}W)\sum_{j=0}^{k-1}(I-T)(I-\alpha T)^j \\
&= W^TM^{-1}W + (W^TM^{-1}W)\sum_{j=0}^{k-1}(I-\alpha T)^{j+1} \\
&\qquad - (1-\alpha)(W^TM^{-1}W)\sum_{j=0}^{k-1}(I-\alpha T)^j \\
&= (W^TM^{-1}W)\sum_{j=0}^{k}(I-\alpha T)^{j+1} - (1-\alpha)(W^TM^{-1}W)\sum_{j=0}^{k-1}(I-\alpha T)^j \\
&= \alpha\sum_{j=0}^{k}(W^TM^{-1}W)(I-\alpha T)^j + (1-\alpha)(W^TM^{-1}W)(I-\alpha T)^k.
\end{aligned}$$

■

5. Using an Orthogonal Projection Matrix. In the previous sections we assumed that the matrix Z that generates the null space of A has full column rank. In this section we shall focus on the case where the null-space matrix is an orthogonal projection $P = I - A^T(AA^T)^{-1}A$, where A is an $m \times n$ matrix of full row rank. (To avoid confusion we use the notation P rather than Z.) P is an $n \times n$ matrix of rank $n - m$. We shall be concerned with the solution of the system

$$PGPp = d. \tag{5.1}$$

Why use an orthogonal projection? First, if A is sparse it is possible to apply the projection in a way that utilizes the sparsity. Second, applying an orthogonal projection does not increase the norm of a matrix; that is, $\|PGP\| \leq \|G\|$.

In the typical systems that arise in optimization, the vector d can be written as $d = Pg$ for some vector g. If G is nonsingular, then (5.1) is consistent and a solution is

$$p = (PGP)^+Pg,$$

where $(PGP)^+$ is the Moore-Penrose generalized inverse of PGP.

It is possible to solve (5.1) using the linear conjugate-gradient method. If G is positive definite on the null space of A, then in exact arithmetic the conjugate-gradient method will terminate in at most $n-m$ iterations with the solution vector [8].

We shall extend the ideas of §3 to obtain a preconditioner for PGP. As before we assume that we have a positive-definite approximation $M \approx G$. The "inverse" preconditioner will be of the form $(PMP)^+$. An explicit expression for this matrix is provided in the following lemma.

LEMMA 8.

$$\begin{aligned}(PMP)^+ &= M^{-1} - M^{-1}A^T(AM^{-1}A^T)^{-1}AM^{-1} \\ &= P(M^{-1} - M^{-1}A^T(AM^{-1}A^T)^{-1}AM^{-1})P.\end{aligned}$$

Proof. It is easy to verify that

$$(PMP)(M^{-1} - M^{-1}A^T(AM^{-1}A^T)^{-1}AM^{-1}) = P.$$

The lemma follows immediately. ■

Our preconditioners are based on the ideas of §3. Let $R = (AA^T)^{-1}A$ be a left-inverse for A^T. We use a power series expansion for $(AM^{-1}A^T)^{-1}$ with RMR^T as the approximate inverse. Denoting $U = (AM^{-1}A^T)(RMR^T)$, we obtain

$$(PMP)^+ \approx PM^{-1}P - PM^{-1}A^T(RMR^T)\Big(\sum_{j=0}^{k-1}(I-U)^j\Big)AM^{-1}P.$$

Now let $T = (PMP)(PM^{-1}P) = PMPM^{-1}P$. The following result is analogous to Lemma 5.

LEMMA 9.

$$PM^{-1}P - PM^{-1}A^TY^{-1}\Big(\sum_{j=0}^{k-1}(I-U)^j\Big)AM^{-1}P = (PM^{-1}P)\sum_{j=0}^{k}(I-T)^j.$$

Proof. Similar to the proof of Lemma 5. ■

The expression on the right is another form for our inverse preconditioner. A formula analogous to (3.5) could also be derived. Techniques similar to those described in §4 can be used to guarantee convergence of the appropriate infinite series.

5.1. Application to Linear Programming. There is possibly another application of these results. In recent years several successful interior-point algorithms have been developed for solving linear programs. Interior point methods typically require few iterations, but each iteration requires the solution of a system of the form

$$AMA^Tp = d \tag{5.2}$$

to provide a search direction. The matrix M is diagonal and changes from one iteration to another (while A remains constant). The principal cost lies in (repeatedly) computing the Cholesky factorization of AMA^T.

Here we propose an approach for approximately solving (5.2) that requires only one Cholesky factorization. We start by using the approximation

$$M^{-1} - M^{-1}A^T(AM^{-1}A^T)^{-1}AM^{-1} \approx (PM^{-1}P)\sum_{j=0}^{k}(I-T)^j.$$

Premultiplying by RM (where $R = (AA^T)^{-1}A$), postmultiplying by (MR^T), and reordering gives

$$(AM^{-1}A^T)^{-1} \approx RMR^T - RMPM^{-1}P(\sum_{j=0}^{k}(I-T)^j)MR^T.$$

The expression on the right involves only P, R, the diagonal matrix M and its inverse. The main work needed is to factor AA^T (just once). The solution of (5.2) requires $3k+4$ applications of $(AA^T)^{-1}$, $2k+4$ applications of M or M^{-1}, and one application each of A and A^T per conjugate-gradient iteration.

6. Using Variable Reduction. If $Z^TZ = I$, Lemma 2 shows that Z^TGZ cannot be more ill-conditioned than G. However if Z is obtained from variable reduction, as would be more likely in sparse problems, this need not be true. We would like to have a better understanding of how the choice of Z affects the conditioning of the problem, and if the conditioning can be monitored or controlled as the optimization problem is being solved.

The ideas in this section are motivated by the following lemma:

LEMMA 10. *If Z_0 is an orthogonal null-space matrix, and Z is a null-space matrix obtained from some other technique such as variable reduction, then*

$$cond(Z^TGZ) \leq cond(Z_0^TGZ_0) \cdot cond(Z)^2.$$

Proof. Since Z_0 and Z are both null-space matrices, $Z = Z_0T$ for some nonsingular matrix T. The largest eigenvalue of Z^TGZ satisfies

$$\begin{aligned}
\lambda_{max}(Z^TGZ) &= \max_{v\neq 0} \frac{v^TZ^TGZv}{v^Tv} \\
&= \max_{v\neq 0} \frac{v^TT^TZ_0^TGZ_0Tv}{v^Tv} \\
&= \max_{v\neq 0} \frac{(Tv)^T(Z_0^TGZ_0)(Tv)}{(Tv)^T(Tv)} \frac{v^TT^TTv}{v^Tv} \\
&= \max_{v\neq 0} \frac{(Tv)^T(Z_0^TGZ_0)(Tv)}{(Tv)^T(Tv)} \frac{v^T(Z_0T)^T(Z_0T)v}{v^Tv} \\
&\leq \max_{w\neq 0} \frac{w^T(Z_0^TGZ_0)w}{w^Tw} \max_{v\neq 0} \frac{v^T(Z^TZ)v}{v^Tv} \\
&= \lambda_{max}(Z_0^TGZ_0)\lambda_{max}(Z^TZ)
\end{aligned}$$

An analogous result is true for the smallest eigenvalue. The lemma follows immediately. ■

The lemma shows that it is desirable to keep cond(Z) small. If an orthogonal factorization is used, then cond$(Z) = 1$. More generally, cond$(Z) = \sigma_{max}(Z)/\sigma_{min}(Z)$, the ratio of the largest and smallest singular values of Z. When variable reduction is used, the smallest singular value can be determined, as the following lemma indicates.

LEMMA 11. *Let Z be a null-space matrix obtained from variable reduction applied to an $m \times n$ matrix $A = (\,B \quad N\,)$, so that*

$$Z = \begin{pmatrix} -B^{-1}N \\ I \end{pmatrix}.$$

Then

$$\sigma_{min}(Z) \geq \sqrt{1 + \frac{\sigma_{min}(N)}{\sigma_{max}(B)}} \geq 1.$$

If the dimension of the null space of N is k then the k smallest singular values of Z are equal to one. In particular, if $n > 2m$ then $\sigma_{min}(Z) = 1$.

A simple consequence of the lemma is that

$$\text{cond}(Z) = \frac{\sigma_{max}(Z)}{\sigma_{min}(Z)} \leq \sigma_{max}(Z) = \|Z\|.$$

Hence

$$\begin{aligned}
\text{cond}(Z)^2 &\leq \max_{\|u\|=1} \|Zu\|^2 \\
&= \max_{\|u\|=1} \left\| \begin{pmatrix} -B^{-1}Nu \\ u \end{pmatrix} \right\|^2 \\
&= \max_{\|u\|=1} \| - B^{-1}Nu\|^2 + \|u\|^2 \\
&= 1 + \max_{\|u\|=1} \| - B^{-1}Nu\|^2 \\
&= 1 + \|B^{-1}N\|^2.
\end{aligned}$$

Thus, keeping $\text{cond}(Z)$ small is closely related to keeping $\|B^{-1}N\|$ small. If $n > 2m$ they are the same.

There is considerable choice in the selection of B and N. Any subset of m variables that results in a nonsingular basis matrix B is acceptable. The procedure of selecting a basis, called a "crash" in the context of linear programming, can be based on various criteria. For example, columns of A can be selected to try to produce a matrix B that is sparse or nearly triangular. We would like to control $\|B^{-1}N\|$ as columns are added incrementally to the basis, but the matrix is only defined when B is invertible. However, we have the bound

$$\|B^{-1}N\| \leq \|B^{-1}\| \cdot \|N\| = \frac{\sigma_{max}(N)}{\sigma_{min}(B)}$$

and this bound is defined even if B is only partially formed.

Suppose that the columns of A are ordered according to some auxiliary criterion, such as sparsity, and that the columns will be considered for membership in B based on this ordering. We would like to estimate $\sigma_{max}(N)/\sigma_{min}(B)$ each time a column is added to the (trial) basis and then reject columns that cause the bound to be large. If N has many columns this would be expensive, so this is not likely to be useful as a general technique. In some circumstances, for example if all the constraints in the optimization problem were linear and so the basis would be used many times, it might be worth the effort.

In many cases we would expect that $\sigma_{max}(N)$ would not vary greatly as B changed. This would be true if the columns of A were all scaled to have norm 1, or if the norms of particular columns were not pathologically large or small. For this reason it should be sufficient in many circumstances merely to monitor $\sigma_{min}(B)$ as B is formed.

Traditional condition number estimators, such as the Linpack estimator, completely factor the matrix B before using the factorization to estimate the condition

number. Since the goal is to examine B as each new column is added, these techniques are not appropriate.

Incremental condition number estimators have been proposed [1], but they only apply to triangular matrices. They would be appropriate if a QR factorization of B were computed, but LU factorizations are more commonly used. To overcome this difficulty we propose exploiting the following upper bound

$$\sigma_{min}(B) = \sigma_{min}(LU) \geq \sigma_{min}(L)\sigma_{min}(U).$$

This can be a considerable overestimate, but it is certainly true that if $\sigma_{min}(L)$ and $\sigma_{min}(U)$ are reasonably sized then so is $\sigma_{min}(B)$.

The algorithm used to factor B will generally ensure that L is well conditioned. (Some software packages monitor U instead of L, in which case the comments below should be adjusted appropriately.) The diagonal entries in L will be equal to one, and the subdiagonal entries will be bounded (by one in the dense case, by a somewhat larger number in the sparse case). Thus, it would not be unreasonable to estimate simply $\sigma_{min}(U)$. The incremental condition number estimator [1] can monitor this value as U is formed one column (or row) at a time.

To summarize: Ill conditioning can be controlled, even when variable reduction is used to form Z. It requires, however, that the conditioning of the component matrices be monitored as they are formed. The trade-offs between cost and security are clear.

7. Calculus of Variations. A classical problem in the calculus of variations is

$$\text{minimize}_{\,x(t)} \int_a^b J(t, x(t), x'(t))dt, \qquad x(a) = x_a,\ x(b) = x_b, \tag{7.1}$$

where $x(t)$ is some smooth real-valued function. The problem can be converted to a finite-dimensional problem by, for example, discretizing $x(t)$ and approximating it using a cubic spline. The Hessian of the finite-dimensional problem then has the form Z^TGZ and it is possible to apply the preconditioning ideas of the previous sections, even though this is an "unconstrained" problem and the matrix Z does not correspond to an explicit set of constraints. We examine this idea here. (The use of a cubic spline was suggested in [4], although the specific approach used here is different.)

The finite-dimensional analogs of (7.1) can be difficult to solve [12], with the Hessian having many small eigenvalues as the solution is approached. However, the components Z and G of the Hessian are easy to compute and G is easy to invert. The "null-space matrix" Z corresponds to the formulas for the cubic spline, and is independent of the formulas in (7.1). The matrix G is block diagonal, with 4×4 diagonal blocks if cubic splines are used. The formulas for the partial derivatives of $J(t, x, x')$ must be specified, but since J is only a function of three variables, this is not difficult.

To derive the formulas for G and Z, we first discretize the problem using

$$a = t_0 < t_1 < \cdots < t_n < t_{n+1} = b.$$

For simplicity we use equally-spaced points: $t_i = a + ih$, where $h = (b - a)/(n + 1)$, although an adaptive mesh could also be used. The variables in the finite-dimensional problem will be $x_i \equiv x(t_i)$.

The function $x(t)$ will be approximated by a cubic spline $s(t)$ that interpolates the values $\{ x_i \}$ at the points $\{ t_i \}$. We use the representation described in [9]. On

the i-th subinterval $[x_i, x_{i+1}]$,

$$s(t) = x_i b_1(t) + d_i b_2(t) + x_{i+1} b_3(t) + d_{i+1} b_4(t) \equiv \sum_{k=1}^{4} \alpha_{i,k} b_k(t),$$

where

$$\begin{aligned} b_1(t) &= \frac{2}{h^3}(t - t_{i+1})^2(t - t_i + h/2) \\ b_2(t) &= \frac{1}{h^2}(t - t_i)(t - t_{i+1})^2 \\ b_3(t) &= -\frac{2}{h^3}(t - t_i)^2(t - t_{i+1} - h/2) \\ b_4(t) &= \frac{1}{h^2}(t - t_i)^2(t - t_{i+1}) \end{aligned}$$

and d_i and d_{i+1} are estimates of $x'(t_i)$ and $x'(t_{i+1})$, respectively. It is straightforward to verify that $s(t_j) = x_j$ and $s'(t_j) = d_j$ for $j = i, i+1$. The coefficients $\{ d_i \}$ are uniquely determined by the requirements that the function $s''(t)$ be continuous, together with two auxiliary conditions (usually involving the derivatives of $s(t)$ at the endpoints of the interval $[a, b]$) [9]. If we define $x = (x_0, \ldots, x_{n+1})^T$ and $d = (d_0, \ldots, d_{n+1})^T$ and use the auxiliary conditions

$$d_0 = \frac{x_1 - x_0}{h} \quad \text{and} \quad d_{n+1} = \frac{x_{n+1} - x_n}{h},$$

then d can be determined by solving a linear system of the form

$$Td = Sx,$$

where T and S are tridiagonal matrices. With our choice of auxiliary conditions,

$$S = \begin{pmatrix} -\frac{1}{h} & \frac{1}{h} & & & & \\ -3 & 0 & 3 & & & \\ & -3 & 0 & 3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -3 & 0 & 3 \\ & & & & -\frac{1}{h} & \frac{1}{h} \end{pmatrix}$$

and

$$T = \begin{pmatrix} 1 & 0 & & & & \\ h & 4h & h & & & \\ & h & 4h & h & & \\ & & \ddots & \ddots & \ddots & \\ & & & h & 4h & h \\ & & & & 0 & 1 \end{pmatrix}.$$

Only the non-zero entries of these matrices need be stored.

We are now in a position to define the matrices Z and G. We begin with Z. The first step is to write

$$\begin{pmatrix} x \\ d \end{pmatrix} = \begin{pmatrix} I \\ T^{-1}S \end{pmatrix} x \equiv Z_1 x.$$

(Of course, T^{-1} would not be formed explicitly since it is a dense matrix; Gaussian elimination would be used to perform the necessary calculations.)

The next step is to determine $\{ \alpha_{i,k} \}$ from x and d. Let

$$\alpha = (\alpha_{0,1}, \alpha_{0,2}, \alpha_{0,3}, \alpha_{0,4}, \alpha_{1,1}, \ldots)^T$$

be the vector of spline coefficients. Then α is just a re-ordering of the variables $\{ x_i \}$ and $\{ d_i \}$ so that

$$\alpha = Z_2 \begin{pmatrix} x \\ d \end{pmatrix},$$

where Z_2 is a matrix with 0/1 entries. For example, if $n = 1$ then

$$\begin{pmatrix} \alpha_{0,1} \\ \alpha_{0,2} \\ \alpha_{0,3} \\ \alpha_{0,4} \\ \alpha_{1,1} \\ \alpha_{1,2} \\ \alpha_{1,3} \\ \alpha_{1,4} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ d_0 \\ d_1 \\ d_2 \end{pmatrix}.$$

This re-ordering of the variables is used so that the matrix G will be block diagonal. The matrix Z_2 would not be formed explicitly, but would instead be represented by a set of rules for obtaining $\{ \alpha_{i,j} \}$ from $\{ x_i \}$ and $\{ d_i \}$. Then $Z = Z_2 Z_1$.

In this application the values x_0 and x_{n+1} are specified, so are not really variables. An additional matrix Z_0 of the form

$$Z_0 = \begin{pmatrix} 0 & \cdots & 0 \\ & I & \\ 0 & \cdots & 0 \end{pmatrix}$$

could be used to remove them from the problem, giving $Z = Z_2 Z_1 Z_0$. In the numerical tests in §8, Z_0 was not used.

To determine the matrix G we must first estimate the integral in (7.1). We apply a quadrature rule separately on each subinterval $[t_i, t_{i+1}]$. If we use a four-point Gaussian quadrature rule [9] then

$$\int_{t_i}^{t_{i+1}} J(t, x(t), x'(t)) dt \approx \sum_{j=1}^{4} w_j J(t_i + \theta_j, s(t_i + \theta_j), s'(t_i + \theta_j)),$$

where $\{ w_j \}$ are the weights and $\{ \theta_j \}$ are the abscissas for the quadrature rule:

$$\theta \approx h \begin{pmatrix} .0694318442030 \\ .3300094782075 \\ .6699905217925 \\ .9305681557970 \end{pmatrix} \quad \text{and} \quad w \approx h \begin{pmatrix} .1739274225685 \\ .3260725774315 \\ .3260725774315 \\ .1739274225685 \end{pmatrix}.$$

The weights and abscissas are the same for all subintervals.

The matrix G is block-diagonal, with one block for each subinterval. Within each block the (k,l) entry is the second partial derivative of the quadrature formula with respect to $\alpha_{i,k}$ and $\alpha_{i,l}$:

$$\sum_{j=1}^{4} w_j \left[\frac{\partial^2 J}{\partial s^2} b_k(t_i+\theta_j) b_l(t_i+\theta_j) + \frac{\partial^2 J}{\partial s \partial s'} b'_k(t_i+\theta_j) b_l(t_i+\theta_j) \right.$$
$$\left. + \frac{\partial^2 J}{\partial s \partial s'} b_k(t_i+\theta_j) b'_l(t_i+\theta_j) + \frac{\partial^2 J}{\partial s'^2} b'_k(t_i+\theta_j) b'_l(t_i+\theta_j) \right].$$

The formulas for $b_k(t)$ and $b'_k(t)$ depend only on those for the spline $s(t)$. The partial derivatives of J with respect to s and s' must be derived, but this is the only calculation that depends on J. Hence, once the partial derivatives of J with respect to s and s' have been specified, the derivatives of the discretized calculus-of-variations problem can be computed in a straightforward, general-purpose manner that is independent of the particular problem being solved. Finally, the Hessian of the discretized calculus of variations problem with respect to the variables $\{ x_i \}$ is given by $Z^T G Z$.

A left-inverse W^T is easy to obtain. It can be chosen as a column permutation of the matrix $(I, 0)$. For example it is possible to choose $W_{1,1} = 1$ and then $W_{i,4i-5} = 1$ for $i > 1$, and all other entries $W_{i,j} = 0$.

It is also possible to determine the left inverse $W^T = (Z^T Z)^{-1} Z^T$ in a computationally efficient manner, even though $(Z^T Z)$ is a dense matrix. Since $Z = Z_2 Z_1$, $Z^T Z = Z_1^T Z_2^T Z_2 Z_1$. It is straightforward to show that

$$Z_2^T Z_2 = \begin{pmatrix} D & 0 \\ 0 & D \end{pmatrix},$$

where D is an $(n+2)\times(n+2)$ diagonal matrix with diagonal entries $D_{1,1} = D_{n+2,n+2} = 1$ and $D_{i,i} = 2$ for $2 < i < n+1$. Then

$$Z^T Z = Z_1^T Z_2^T Z_2 Z_1 = D + S^T T^{-T} D T^{-1} S$$

in terms of the tridiagonal matrices T and S defined earlier. The last formula is the Schur complement of $-TD^{-1}T^T$ in the block 2×2 matrix

$$H \equiv \begin{pmatrix} -TD^{-1}T^T & S \\ S^T & D \end{pmatrix}.$$

If we define $\bar{Z}^T = (0\ I)$, then

$$(Z^T Z)^{-1} = (D + S^T T^{-T} D T^{-1} S)^{-1} = \bar{Z}^T H^{-1} \bar{Z}.$$

The matrix H is a permutation of a 7-diagonal matrix, which can be factored inexpensively. With this approach, the left inverse $W^T = (Z^T Z)^{-1} Z^T$ can be formed.

8. Computational Experiments. We tested the inverse preconditioners (3.6) for various values of k on three examples. The calculations were done using MATLAB on a Sun SPARCstation computer, with machine precision $\approx 2.2 \times 10^{-16}$.

In the first example, G was a diagonal matrix with $G_{i,i} = 2\gamma^{i-1}$, where γ was chosen so that $G_{n,n} = 10^7$. The $m \times n$ constraint matrix A was of the form $U\Sigma V^T$, where Σ was a diagonal $m \times n$ matrix with diagonal entries $\Sigma_{i,i} = i$, and U and V were

random square orthogonal matrices. The random number generator was initialized using the MATLAB command `randn('seed',0)` before each run, and random numbers were generated using the `randn` function. (This complicated method of generating A allowed us to control its singular values.)

The second example is based on the calculus of variations problem of §7. The integral was evaluated over the interval $[a, b] = [0, 1]$. The variables were given the values $x_i = 1 - t_i^2$. The kernel was chosen as the convex function

$$J(t, x, x') = x^4 + (x')^4,$$

for which G is positive definite.

For the first problem, two choices of Z were used: an orthogonal Z with $W = Z(Z^TZ)^{-1} = Z$, and a Z based on variable reduction (the first m variables forming the basis) together with the "special purpose" $W^T = (0, I)$. For the second problem Z is given, but we tested two choices of W: $W = Z(Z^TZ)^{-1}$ and the "simple" W mentioned in §7 (i.e., W^T is a permutation of the matrix $(I, 0)$).

The third problem uses matrices of the form

$$G = \begin{pmatrix} G_1 & I \\ I & G_2 \end{pmatrix} \quad \text{and} \quad Z = \begin{pmatrix} I \\ 0 \end{pmatrix},$$

where all the blocks are the same size. The reduced matrix is $Z^TGZ = G_1$. The matrix G_1 was chosen as in the first problem (note that it is a matrix of order $n/2$). The diagonal matrix G_2 was chosen so that the eigenvalues of the preconditioned matrix were the fifth roots of the eigenvalues of G_1. For this problem, both choices of Z are the same.

We were interested in the behavior of the preconditioners under "ideal" circumstances. We therefore used $M = G$ in all problems. To ensure convergence of the series defining the inverse preconditioners, we chose the scaling factor so that α^{-1} was the largest eigenvalue of T (see (3.4)).

The first problem was tested with $n = 100$ and $m = 20$ so that the reduced matrix was 80×80. The second problem was tested with $n = 60$ so that the reduced matrix was 62×62 and G was 244×244. The third problem was tested with $n = 200$ and $m = 100$ so that the reduced matrix was 100×100. In all cases the effect of the preconditioner was assessed in two ways: (i) by the condition number of the preconditioned matrix, (ii) by the number of conjugate-gradient iterations required to solve a linear system with right-hand side $(1, \ldots, 1)^T$. The conjugate-gradient iterations were terminated when the norm of the residual was less than 10^{-8}. The results are listed in Tables 1 and 2.

The tables indicate that the preconditioning strategies can greatly reduce the number of iterations required to solve a linear system using the conjugate-gradient method. The effect on the condition number is less pronounced. The choice of the "generic" left inverse $W^T = (Z^TZ)^{-1}Z^T$ worked well in all these cases (as in many others that we tried). The other "special-purpose" choices of W (i.e., those constructed from permutations of $(0, I)$) were less predictable. In the case of variable reduction we found them to be effective (in fact, better than other choices). However, for the calculus of variations problems we were unable to find a special-purpose choice of W that worked well.

For problems 1 and 2 (and for most examples that we tried), the most dramatic improvement comes with the simplest of the inverse preconditioners, $W^TM^{-1}W$. This

TABLE 1
Effect of Preconditioning Using $W^T = (Z^TZ)^{-1}Z^T$.

Preconditioner	Problem 1 cond	iter	Problem 2 cond	iter	Problem 3 cond	iter
None	1.0×10^6	756	1.3×10^5	164	2.2×10^3	240
$k = 0$	6.8×10^4	34	5.4×10^4	39	4.6×10^2	149
$k = 1$	3.4×10^4	31	2.7×10^4	39	2.3×10^2	125
$k = 2$	2.3×10^4	29	1.8×10^4	38	1.5×10^2	111
$k = 3$	1.7×10^4	27	1.4×10^4	40	1.2×10^2	99
$k = 4$	1.4×10^4	25	1.1×10^4	39	9.3×10^1	88

TABLE 2
Effect of Preconditioning Using Special-purpose W.

Preconditioner	Problem 1 cond	iter	Problem 2 cond	iter
None	1.5×10^5	529	1.3×10^5	164
$k = 0$	4.8×10^1	13	5.4×10^7	395
$k = 1$	2.4×10^1	13	2.7×10^7	396
$k = 2$	1.6×10^1	13	1.8×10^7	379
$k = 3$	1.2×10^1	13	1.3×10^7	372
$k = 4$	9.9×10^0	13	1.1×10^7	374

is reassuring, since it has lower costs than the others. It also does not require that α be selected to ensure convergence of the series used in the derivation. Adding more terms in the series leads to further improvements, but whether these improvements are cost-effective would depend on the specifics of the particular application. For problem 3, the later terms lead to considerable reductions in the number of iterations required, indicating the potential of these techniques.

As a final indication of the behavior of these preconditioners, Figures 1–3 show the eigenvalues of the preconditioned matrix, corresponding to the results in Table 1. In each figure, column "-1" shows the eigenvalues of the original matrix, and columns 0–4 show the eigenvalues of the preconditioned matrix for $k = 0, \ldots, 4$. Note that the range of the eigenvalues has been compressed (hence the reduced condition number) and in some cases there is increased clustering of eigenvalues, a feature that the conjugate-gradient method can exploit.

9. Conclusions. We have described a set of preconditioners for positive definite matrices of the form Z^TGZ, using information about the individual matrices Z and G to construct approximations to $(Z^TGZ)^{-1}$. Matrices of this type arise in constrained optimization problems, in particular in interior-point methods for linear programming. The techniques can also be applied to a class of problems in the calculus of variations. Although some of the preconditioning formulas may be too expensive for routine use, numerical tests suggest that the simplest of the formulas (3.2) can be an effective preconditioner for general use. The more elaborate formulas would be appropriate in

FIGURE 1
Eigenvalues of the Preconditioned Matrix for Problem 1.

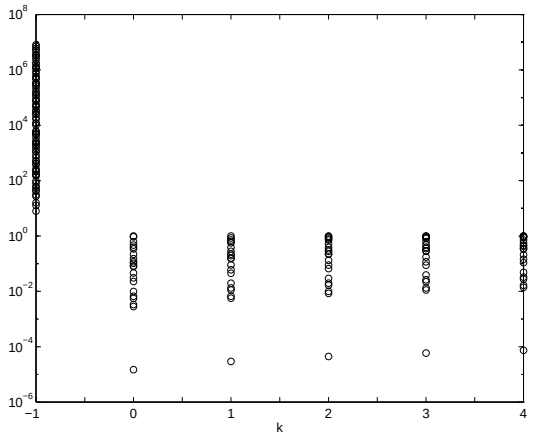
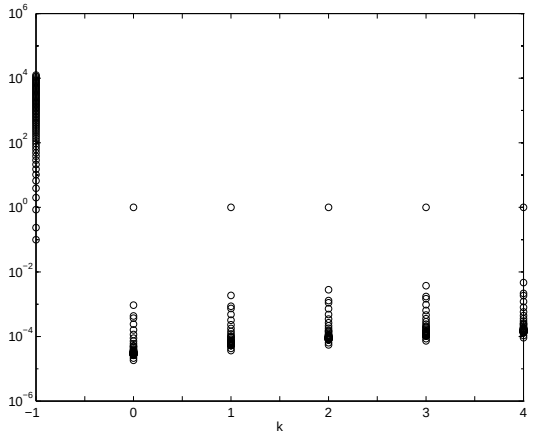


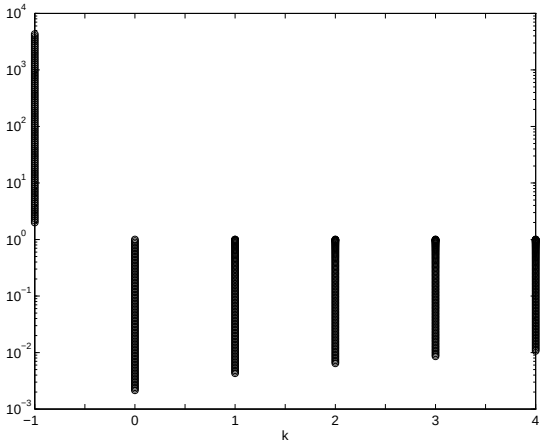
FIGURE 2
Eigenvalues of the Preconditioned Matrix for Problem 2.



cases where a product Gv is computationally expensive. Because (3.2) and the other formulas only require information about Z and G *separately* (and do not require that an approximation to Z^TGZ be provided), the preconditioners can be used within a wide variety of optimization algorithms. This is significant, since the structure of G often comes from the optimization model, whereas the form of Z is determined by the optimization software.

10. Acknowledgments. We wish to thank an anonymous referee for the careful reading of our paper, and for the insightful comments that led to many improvements in our paper.

FIGURE 3
Eigenvalues of the Preconditioned Matrix for Problem 3.



REFERENCES

[1] C.H. Bischof, *Incremental condition estimation,* SIAM J. Matrix Analysis and Applications, 11 (1990), pp. 312–322.

[2] R.S. Dembo and T. Steihaug, *Truncated-Newton algorithms for large-scale unconstrained optimization,* Math. Prog., 26 (1983), pp. 190–212.

[3] R. Fletcher, *Practical Methods of Optimization, Volume 2: Constrained Optimization*, Wiley, New York, 1981.

[4] P.E. Gill and W. Murray, *The numerical solution of a problem in the calculus of variations,* in *Recent Mathematical Developments in Control*, D.J. Bell (editor), pp. 97–122, Academic Press, London (1979).

[5] P.E. Gill, W. Murray, D.B. Ponceleón, and M.A. Saunders, *Preconditioners for indefinite systems arising in optimization,* SIAM J. Matrix Analysis and Applications, 13 (1992), pp. 292–311.

[6] P.E. Gill, W. Murray, and M.H. Wright, *Practical Optimization*, Academic Press, New York, 1981.

[7] G.H. Golub and C. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, 1989.

[8] M.R. Hestenes, *Conjugate-Direction Methods in Optimization*, Springer-Verlag, New York, 1980.

[9] D.K. Kahaner, C. Moler, and S.G. Nash, *Numerical Methods and Software*, Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

[10] S.G. Nash, *Newton-like minimization via the Lanczos method,* SIAM J. Num. Anal., 21 (1984), pp. 770–788.

[11] S.G. Nash, *Preconditioning of truncated-Newton methods,* SIAM J. Sci. Stat. Comp., 6 (1985), pp. 599–616.

[12] S.G. Nash and J. Nocedal, *A numerical study of the limited memory BFGS method and the truncated-Newton method for large scale optimization,* SIAM J. Optimization, 1 (1991), pp. 358–372.

[13] S.G. Nash and A. Sofer, *A barrier method for large-scale constrained optimization,* ORSA Journal on Computing, 5 (1993), pp. 40–53.

[14] T. Rusten and R. Winther, *A preconditioned iterative method for saddlepoint problems,* SIAM J. Matrix Analysis and Applications, 13 (1992), pp. 887–904.