SIAM J. SCI. COMPUT.
Vol. 23, No. 4, pp. 1376–1395

ON THE SOLUTION OF EQUALITY CONSTRAINED QUADRATIC PROGRAMMING PROBLEMS ARISING IN OPTIMIZATION*

NICHOLAS I. M. GOULD†, MARY E. HRIBAR‡, AND JORGE NOCEDAL§

Abstract. We consider the application of the conjugate gradient method to the solution of large equality constrained quadratic programs arising in nonlinear optimization. Our approach is based implicitly on a reduced linear system and generates iterates in the null space of the constraints. Instead of computing a basis for this null space, we choose to work directly with the matrix of constraint gradients, computing projections into the null space by either a normal equations or an augmented system approach. Unfortunately, in practice such projections can result in significant rounding errors. We propose iterative refinement techniques, as well as an adaptive reformulation of the quadratic problem, that can greatly reduce these errors without incurring high computational overheads. Numerical results illustrating the efficacy of the proposed approaches are presented.

Key words. nonlinear optimization, conjugate gradient method, quadratic programming, preconditioning, iterative refinement

AMS subject classifications. 65K10, 49N, 49M, 65F10, 90C06, 90C30

PII. S1064827598345667

1. Introduction.

A variety of algorithms for linearly and nonlinearly constrained optimization (e.g., [9, 14, 15, 36, 37]) use the conjugate gradient (CG) method [28] to solve subproblems of the form

$$\underset{x}{\text{minimize}} \quad q(x) = \tfrac{1}{2}x^T Hx + c^T x \tag{1.1}$$

$$\text{subject to} \quad Ax = b. \tag{1.2}$$

In nonlinear optimization, the n-vector c usually represents the gradient ∇f of the objective function or the gradient of the Lagrangian, the $n \times n$ symmetric matrix H stands for either the Hessian of the Lagrangian or an approximation to it, and the solution x represents a search direction. The equality constraints $Ax = b$ are obtained by linearizing the constraints of the optimization problem at the current iterate. We will assume here that A is an $m \times n$ matrix, with $m < n$, and that A has full row rank so that the constraints $Ax = b$ constitute m linearly independent equations. We also assume for convenience that H is positive definite in the null space of the constraints, as this guarantees that (1.1)–(1.2) has a unique solution.

As we shall see in section 2.1, the solution of (1.1)–(1.2) can be characterized in terms of a nonunique matrix Z whose columns form a basis for the null space of A. Numerous options are available for computing and representing Z, both explicitly and

*Received by the editors October 7, 1998; accepted for publication (in revised form) August 22, 2001; published electronically December 18, 2001.
http://www.siam.org/journals/sisc/23-4/34566.html

†Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire OX11 0QX, England (n.gould@rl.ac.uk).

‡Cray Inc., 411 1st Avenue S, Suite 600, Seattle, WA 98104-2860 (marybeth@cray.com). This author's research was supported by Department of Energy grant DE-FG02-87ER25047-A004.

§ECE Department, Northwestern University, Evanston, IL 60208 (nocedal@ece.nwu.edu). This author's research was supported by National Science Foundation grant CDA-9726385 and by Department of Energy grant DE-FG02-87ER25047-A004.

implicitly [11, 12, 19, 26, 41, 44]. In the context of large-scale optimization, operations with Z and Z^T can be performed using the LU factorization of a nonsingular submatrix of A; see, for example, [20].

In this paper, we consider techniques for solving (1.1)–(1.2) that use a preconditioned CG method and retain feasibility of the iterates by performing projections into the null space of A without a representation of Z. Unfortunately, a straightforward implementation of these techniques may produce computational errors that cause deteriorating feasibility of the CG iterates. We describe iterative refinement techniques that can improve accuracy, when needed. We also propose a mechanism for redefining the vector c adaptively that does not change the solution of the quadratic problem but that has more favorable numerical properties.

Notation. Throughout the paper $\|\cdot\|$ stands for the ℓ_2 matrix or vector norm. We will denote the floating-point *unit roundoff* (or machine precision) by ϵ_m. For double precision IEEE arithmetic, $\epsilon_m \approx 10^{-16}$. We let $\kappa(A)$ denote the condition number of A, i.e., $\kappa(A) = \sigma_1/\sigma_m$, where $\sigma_1 \geq \cdots \geq \sigma_m > 0$ are the nonzero singular values of A.

2. The CG method with linear constraints. We now look at applying CG to approximate the solution of the quadratic problem (1.1)–(1.2). First, we present CG method for a reduced problem; then we show how to apply CG to the full system with a scaled projection operator. Finally, we consider the problem (1.1)–(1.2) with a trust region constraint and show how the given CG methods apply.

2.1. The CG method for the reduced system. A common approach for solving linearly constrained problems is to eliminate the constraints and solve a reduced problem (cf. [21, 39]). More specifically, suppose that Z is an $n \times (n-m)$ matrix spanning the null space of A. Then $AZ = 0$, the columns of A^T together with the columns of Z span $\mathbf{R}^n$, and any solution x^* of the linear equations $Ax = b$ can be written as

$$x^* = A^T {x_A}^* + Z {x_Z}^* \tag{2.1}$$

for some vectors ${x_A}^* \in \mathbf{R}^m$ and ${x_Z}^* \in \mathbf{R}^{n-m}$. The constraints $Ax = b$ yield

$$AA^T {x_A}^* = b, \tag{2.2}$$

which determines the vector ${x_A}^*$. Substituting (2.1) into (1.1), and omitting constant terms (${x_A}^*$ is a constant now), we see that ${x_Z}^*$ solves the reduced problem

$$\underset{x_Z}{\text{minimize}} \quad \tfrac{1}{2} {x_Z}^T H_{ZZ} x_Z + {c_Z}^T x_Z, \tag{2.3}$$

where

$$H_{ZZ} = Z^T H Z, \quad c_Z = Z^T (H A^T {x_A}^* + c).$$

As we have assumed that the reduced Hessian H_{ZZ} is positive definite, the solution of (2.3) is equivalent to that of the linear system

$$H_{ZZ} x_Z = -c_Z. \tag{2.4}$$

We can now apply the conjugate gradient method to compute an approximate solution of the problem (2.3), or, equivalently, the system (2.4), and substitute this into (2.1) to obtain an approximate solution of the quadratic program (1.1)–(1.2).

This strategy of computing the normal component $A^T x_A$ exactly and the tangential component $Z x_Z$ inexactly is followed in many nonlinear optimization algorithms which ensure that, once linear constraints are satisfied, they remain so throughout the remainder of the optimization calculation (cf. [21]).

Let us now consider the practical application of the CG method to the reduced system (2.4). It is well known that *preconditioning* can improve the rate of convergence of the CG iteration (cf. [3]). We therefore assume that a preconditioner W_{ZZ} is given, where W_{ZZ} is a symmetric, positive definite matrix of dimension $n - m$, which might be chosen to reduce the span of, and to cluster, the eigenvalues of $W_{ZZ}^{-1} H_{ZZ}$. Ideally, one would like to choose W_{ZZ} so that $W_{ZZ}^{-1} H_{ZZ} = I$, and thus $W_{ZZ} = Z^T H Z$ is an ideal preconditioner. Based on this ideal, we consider in this paper preconditioners of the form $W_{ZZ} = Z^T G Z$, where G is a symmetric matrix such that $Z^T G Z$ is positive definite. Some choices of G will be discussed in the next section.

For preconditioners of the form $W_{ZZ} = Z^T G Z$, the preconditioned CG method applied to the $(n - m)$-dimensional reduced system $H_{ZZ} x_Z = -c_Z$ is as follows (see, e.g., [22, p. 532]).

ALGORITHM 2.1 (preconditioned CG for reduced systems). *Choose an initial point x_Z, compute $r_Z = Z^T H Z x_Z + c_Z$, $g_Z = (Z^T G Z)^{-1} r_Z$, and $p_Z = -g_Z$. Repeat the following steps, until a termination test is satisfied:*

$$\alpha = {r_Z}^T g_Z / {p_Z}^T Z^T H Z p_Z, \tag{2.5}$$

$$x_Z \leftarrow x_Z + \alpha p_Z, \tag{2.6}$$

$${r_Z}^+ = r_Z + \alpha Z^T H Z p_Z, \tag{2.7}$$

$${g_Z}^+ = (Z^T G Z)^{-1} {r_Z}^+, \tag{2.8}$$

$$\beta = ({r_Z}^+)^T {g_Z}^+ / {r_Z}^T g_Z, \tag{2.9}$$

$$p_Z \leftarrow -{g_Z}^+ + \beta p_Z, \tag{2.10}$$

$$g_Z \leftarrow {g_Z}^+ \quad \textit{and} \quad r_Z \leftarrow {r_Z}^+. \tag{2.11}$$

This iteration may be terminated, for example, when ${r_Z}^T (Z^T G Z)^{-1} r_Z$ is sufficiently small.

Several algorithms for large-scale optimization are based on combining a suitable representation of Z with CG methods for solving the reduced system [18, 33, 46]. Coleman and Verma [13] and Nash and Sofer [38] have proposed strategies for defining reduced-system preconditioners which approximate $Z^T H Z$ in different ways.

We present Algorithm 2.1 for illustrative purposes only. In the next section, we describe modifications to this algorithm which make it possible to avoid operating with the null space basis Z.

2.2. The CG method for the full system. If we were to compute an approximate solution using Algorithm 2.1, it must be multiplied by Z and substituted in (2.1) to give the approximate solution of the quadratic program (1.1)–(1.2). Alternatively, we may rewrite Algorithm 2.1 so that the multiplication by Z and the addition of the term $A^T {x_A}^*$ is computed within the CG iteration. To do so, we introduce, in the following algorithm, the n-vectors x, r, g, p which satisfy $x = Z x_Z + A^T {x_A}^*$, $Z^T r = r_Z$, $g = Z g_Z$, and $p = Z p_Z$. We also define the scaled projection matrix

$$P = Z (Z^T G Z)^{-1} Z^T. \tag{2.12}$$

We note, for future reference, that P is independent of the choice of null space basis Z.

ALGORITHM 2.2 (preconditioned CG in expanded form). *Choose an initial point x satisfying $Ax = b$, compute $r = Hx + c$, $g = Pr$, and $p = -g$. Repeat the following steps, until a convergence test is satisfied:*

$$\alpha = r^T g / p^T H p, \tag{2.13}$$

$$x \leftarrow x + \alpha p, \tag{2.14}$$

$$r^+ = r + \alpha H p, \tag{2.15}$$

$$g^+ = P r^+, \tag{2.16}$$

$$\beta = (r^+)^T g^+ / r^T g, \tag{2.17}$$

$$p \leftarrow -g^+ + \beta p, \tag{2.18}$$

$$g \leftarrow g^+ \quad \textit{and} \quad r \leftarrow r^+. \tag{2.19}$$

This will be the main algorithm studied and further refined in this paper. It is important to notice that this algorithm, unlike its predecessor, is independent of the choice of Z. In the next section, different choices for P will be presented.

Note that the vector g^+, which we call the *preconditioned residual*, has been defined to be in the null space of A. As a result, in exact arithmetic, all the search directions p generated by Algorithm 2.2 will also lie in the null space of A, and thus the iterates x will all satisfy $Ax = b$. However, computed representations of the scaled projection P can produce rounding errors that may cause p to have a significant component outside the null space of A, leading to convergence difficulties. This will be the subject of later sections of the paper.

Several types of stopping tests can be used, but since their choice depends on the requirements of the optimization method, we shall not discuss them here. In the numerical tests reported in this paper, we terminate the CG iteration based on the quantity $r^T g \equiv r^T P r \equiv g^T G g$. An initial point satisfying $Ax = b$ can be computed, for example, by solving the normal equations (2.2).

Two simple choices of G are $G = \mathrm{diag}(H)$, and $G = I$. The first choice may be appropriate when the diagonal elements of H are of widely different magnitudes. This is the case, for example, in barrier methods for constrained optimization that handle bound constraints $l \le x \le u$ by adding terms of the form $-\mu \sum_{i=1}^{n} (\log(x_i - l_i) + \log(u_i - x_i))$ to the objective function for some positive barrier parameter μ. The second choice, $G = I$, arises in trust region methods, as we discuss next.

2.3. The CG method and the trust region problem. In trust region methods, the problem (1.1)–(1.2) also contains a trust region constraint of the form $\|x\| \le \Delta$. Steihaug [43] noted, however, that the trust region constraint can be easily imposed if the initial estimate of the solution of (1.1)–(1.2) is chosen to be the vector zero. In this case the CG iterates are monotonically increasing in norm, and the CG iteration can be terminated as soon as the norm of one of the iterates exceeds the trust region radius. No other changes to the CG iteration are needed.

For the reduced problem (2.3), the added trust region constraint has the form $\|Z x_z\| \le \Delta_z$. In order to transform it into a spherical constraint, we introduce the change of variables $x_z \leftarrow (Z^T Z)^{-1/2} x_z$ whose effect in the CG iteration is identical to that of replacing $(Z^T G Z)^{-1}$ by $(Z^T Z)^{-1}$ in (2.8). Thus, the choice $G = I$ arises in several trust region methods for constrained optimization [9, 15, 16, 27, 36, 40, 47]. Since the role of this matrix is not to produce a clustering of the eigenvalues, we will regard Algorithm 2.2 with the choice $G = I$ as an *unpreconditioned* CG iteration.

3. The CG algorithm without a null space basis. We are interested here in using Algorithm 2.2 in such a way that a representation of Z is not necessary. This will be possible because, as is well known, there are alternative ways of expressing the scaled projection operator (2.12).

3.1. Computing projections. We now discuss how to apply the projection operator $Z(Z^TGZ)^{-1}Z^T$ to a vector without a representation of the null space basis Z.

Let us begin by considering the simple case when $G = I$, so that P is the orthogonal projection operator onto the null space of A. We denote it by P_Z, i.e.,

$$P_Z = Z(Z^TZ)^{-1}Z^T. \tag{3.1}$$

Thus the preconditioned residual g^+ (2.16) is the result of projecting r^+ into the null space of A and can be written as

$$g^+ = P_Z r^+. \tag{3.2}$$

This projection can be performed in two alternative ways.

The first is to replace P_Z by the equivalent formula

$$P_A = I - A^T(AA^T)^{-1}A \tag{3.3}$$

and thus to replace (3.2) with

$$g^+ = P_A r^+. \tag{3.4}$$

We can express this as

$$g^+ = r^+ - A^Tv^+, \tag{3.5}$$

where v^+ is the solution of

$$AA^Tv^+ = Ar^+. \tag{3.6}$$

Noting that (3.6) are the normal equations, it follows that v^+ is the solution of the least squares problem

$$\underset{v}{\text{minimize}} \quad \|r^+ - A^Tv^+\| \tag{3.7}$$

and that the desired projection g^+ is the corresponding residual. The approach (3.5)–(3.6) for computing the projection $g^+ = P_Z r^+$ will be called the *normal equations approach*. In this paper, we assume that (3.6) will be solved using a Cholesky factorization of AA^T.

The second possibility is to express the projection (3.2) as the solution of the augmented system

$$\begin{pmatrix} I & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} g^+ \\ v^+ \end{pmatrix} = \begin{pmatrix} r^+ \\ 0 \end{pmatrix}. \tag{3.8}$$

In this paper, we assume that this system will be solved by means of a symmetric indefinite factorization that uses 1×1 and 2×2 pivots [22]. We refer to this as the *augmented system approach*.

Now let us suppose that preconditioning has the more general form

$$g^+ = P_{Z:G} r^+, \quad \text{where} \quad P_{Z:G} = Z(Z^T G Z)^{-1} Z^T. \tag{3.9}$$

This may be expressed as

$$g^+ = P_{A:G} r^+, \quad \text{where} \quad P_{A:G} = G^{-1}\left(I - A^T (AG^{-1}A^T)^{-1} AG^{-1}\right) \tag{3.10}$$

if G is nonsingular, and can be found as the solution of

$$\begin{pmatrix} G & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} g^+ \\ v^+ \end{pmatrix} = \begin{pmatrix} r^+ \\ 0 \end{pmatrix} \tag{3.11}$$

whenever $z^T G z \neq 0$ for all nonzero z for which $Az = 0$ (see, e.g., [21, section 5.4.1]). While (3.10) is far from appealing when G^{-1} does not have a simple form, (3.11) is a useful generalization of (3.8). Clearly, the system (3.8) may be obtained from (3.11) by setting $G = I$, and the perfect preconditioner results if $G = H$, but other choices for G are also possible; all that is required is that $z^T G z > 0$ for all nonzero z for which $Az = 0$. The idea of using the projection (3.3) in the CG method dates back to at least [42]; the alternative (3.11), and its special case (3.8), are proposed in [10], although [10] unnecessarily requires that G be positive definite. A more recent study on preconditioning the projected CG method is [13], while the eigenstructure of the preconditioned system is examined by [35, 37].

Interestingly, preconditioning in Coleman and Verma's null space approach [13] requires the solution of systems like (3.11), but it allows A to be replaced by a sparser matrix. (The price to pay for this relaxation is that products involving a suitable null space matrix are required.) Such an approach has considerable merit, especially in the case where using the exact A leads to significant fill-in during the factorization of the coefficient matrix of (3.11). It remains to be seen how such an approach compares with those we propose here when used in algorithms for large-scale constrained optimization.

Note that (3.4), (3.8), and (3.11) do not make use of a null space basis Z and require only factorization of matrices involving A. Significantly, all three forms allow us to compute an initial point satisfying $Ax = b$, the first because it relies on a factorization of AA^T, from which we can compute $x = A^T(AA^T)^{-1}b$, while factorizations of the system matrices in (3.8) and (3.11) allow us to find a suitable x by solving

$$\begin{pmatrix} I & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ b \end{pmatrix} \text{ or } \begin{pmatrix} G & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ b \end{pmatrix}.$$

Unfortunately all three of our proposed alternatives, (3.4), (3.8), and (3.11) for computing g^+ can give rise to significant roundoff errors that prevent the iterates from remaining in the null space of A, particularly as the CG iterates approach the solution. The difficulties are caused by the fact that, as the iterations proceed, the projected vector $g^+ = Pr^+$ becomes increasingly small while r^+ does not. Indeed, the optimality conditions of the quadratic program (1.1)–(1.2) state that the solution x^* satisfies

$$Hx^* + c = A^T \lambda \tag{3.12}$$

for some Lagrange multiplier vector λ. The vector $Hx + c$, which is denoted by r in Algorithm 2.2, will generally stay bounded away from zero, but as indicated by

(3.12), it will become increasingly closer to the range of A^T. In other words, r will tend to become orthogonal to Z, and hence, from (3.9), the preconditioned residual g will converge to zero so long as the smallest eigenvalue of Z^TGZ is bounded away from zero.

That this discrepancy in the magnitudes of $g^+ = Pr^+$ and r^+ will cause numerical difficulties is apparent from (3.5), which shows that significant cancellation of digits will usually take place. The generation of harmful roundoff errors is also apparent from (3.8) and (3.11) because g^+ will be small while the remaining components v^+ remain large. Since the magnitude of the errors generated in the solution of (3.8) and (3.11) is governed by the size of the large component v^+, the vector g^+ is likely to contain large relative errors. These arguments will be made more precise in the next section.

Now consider an example problem. Since the goal of this paper is not to evaluate the efficiency of particular choices of preconditioners, in all the examples given in this paper we will choose $G = I$, which, as we have mentioned, arises in trust region optimization methods without preconditioning. To assess the techniques to be proposed, we need to measure the closeness of g to the null space of A. For this purpose, we have chosen

$$\cos\theta = \max_i \left\{ \frac{A_i^T g}{||A_i||\ ||g||} \right\}, \tag{3.13}$$

where A_i is the ith row of A. The value of $\cos\theta$ provides a relative measure of orthogonality with the property that, for nonzero g, it vanishes if and only if g lies in the null space of A.

Example 3.1. We applied Algorithm 2.2 to solve problem CVXQP3 from the CUTE collection [6], with $n = 1000$ and $m = 750$, where the simple bounds were removed to create a problem of the form (1.1)–(1.2). We used both the normal equations (3.5)–(3.6) and augmented system (3.8) approaches to compute the projection and define $G = I$. The results are given in Figure 3.1, which plots $\sqrt{r^Tg} = ||r_z||$ (resid), the norm of the null space component of the residual, as a function of the iteration number. In both cases the CG iteration was terminated when r^Tg became negative, which indicates that severe errors have occurred since r^Tg must be positive. (Continuing the iteration past this point resulted in oscillations in the norm of the gradient without any significant improvement.) At iteration 50 of both runs, r is of order 10^5 whereas its projection g is of order 10^{-1}. Figure 3.1 also plots (3.13), the cosine of the angle between the preconditioned residual g and the rows of A. Note that this cosine, which should be zero in exact arithmetic, increases and indicates that the CG iterates leave the constraint manifold $Ax = b$.

We believe it is reasonable to attribute the failure of the CG algorithm to the deviation of the iterates from the constraint manifold $Ax = b$, since the derivation of Algorithm 2.2 from its predecessor is predicated on the assumption that the search is restricted to this manifold. An analysis by Arioli, Duff, and de Rijk [2] (which improves on [1]) indicates that, with care, it is possible to ensure that the backward error[1]

$$A_i^T g^+ / (|A||g^+|)_i$$

of the computed g^+ is of the order of the machine precision, ϵ_m for the case $G = I$. (Here $|\cdot|$ denotes the componentwise absolute value.)

[1] This definition needs to be modified if $|A||g^+|$ is (close to) zero. See [1] for details.

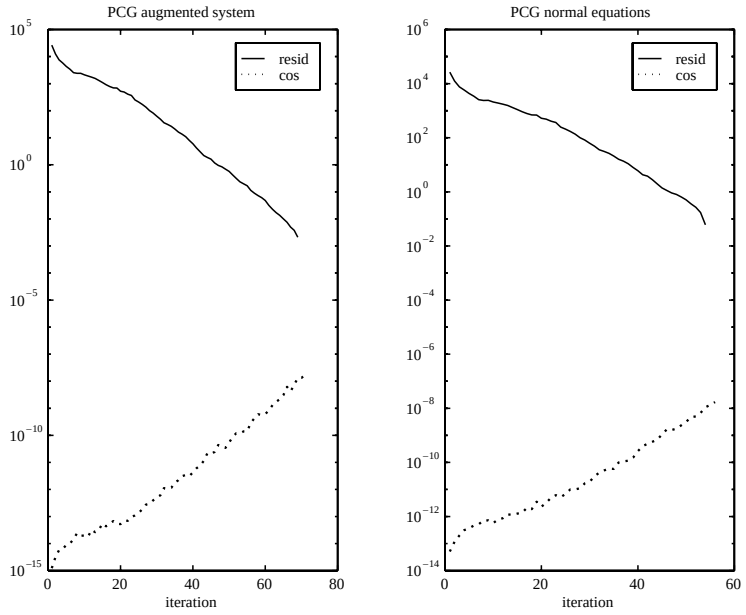

FIG. 3.1. *The CG method with two options for the projection.*

Errors such as those illustrated in Example 3.1 are not uncommon in optimization calculations based on Algorithm 2.2. This is of concern, as it may cause the outer optimization algorithms to fail to achieve feasibility or to require many iterations to do so. A particular example is given by problem ORTHREGA from the CUTE collection, which cannot be solved to a prescribed accuracy using the trust region CG approach of [31]; see [31, pp. 33–34] and section 7.

In sections 5 and 6 we propose several remedies. One of them is based on an adaptive redefinition of r that attempts to minimize the differences in magnitudes between $g^+ = Pr^+$ and r^+. We also describe several forms of iterative refinement for the projection operation. All these techniques are motivated by the roundoff error analysis given next.

4. Sources of errors. We now present error bounds that support the arguments made in the previous section, particularly the claim that the most problematic situation occurs in the latter stages of the CG iteration when g^+ is converging to zero, but r^+ is not. That is, we shall presume that $\|r^+\|$ is much larger than its projection $\|g^+\|$. For simplicity, we shall assume henceforth that A has been scaled so that $\|A\| = \|A^T\| = 1$ and shall only consider the simplest possible choice, $G = I$. Any computed, as opposed to exact, quantity will be denoted by a subscript c.

First consider the *normal equations approach*. Here the projection $g^+ = P_A r^+$ is given by (3.5), where (3.6) is solved by means of the Cholesky factorization of AA^T. In finite precision, it is straightforward to deduce that the relative error in the projection satisfies[2]

$$\frac{\|g^+ - g_c^+\|}{\|g^+\|} \leq \gamma \epsilon_m \kappa^2(A) \frac{\|v^+\|}{\|g^+\|}, \tag{4.1}$$

[2]If $\|g^+\|$ is small, it is preferable to replace the denominators in (4.1) by $\max(\|g^+\|, \epsilon)$, where ϵ is a suitable multiple (e.g., 10) of ϵ_m.

where $\gamma = 2.5n^{3/2}$, using the analysis of [5, p. 49].[3] We can thus conclude that the error in the projection (4.1) can be significant when $\kappa(A)$ or

$$\frac{\|v^+\|}{\|g^+\|} = \frac{\|v^+\|}{\|P_A r^+\|} \approx \frac{\|r^+\|}{\|P_A r^+\|} \tag{4.2}$$

is large, the latter approximation resulting from (3.5) and the assumption that $\|A\| = 1$, since then $\|r^+\| \approx \|A^T v^+\| \leq \|v^+\|$.

When the condition number $\kappa(A)$ is moderate, the contribution of the ratio (4.2) to the relative error (4.1) is normally not large enough to cause failure of the outer optimization calculation. For example, a stopping test in a nonlinear optimization algorithm, which causes termination when projected residual g^+ is (say) 10^{-6} times smaller in norm than the initial residual, has a ratio (4.2) of roughly 10^6. In this case, using double precision arithmetic, one would have sufficient accuracy to make progress toward the solution. However, as the condition number $\kappa(A)$ grows, the loss of significant digits becomes severe, especially since $\kappa(A)$ appears squared in (4.1).

Now consider the *augmented system approach* (3.11). Again we will focus on the choice $G = I$ for which the preconditioned residual $g^+ = Pr^+$ is computed by solving the system (3.8) using a direct method. There are a number of such methods, the strategies of Bunch and Kaufman [7] and Duff and Reid [17] being the best known examples for dense and sparse matrices, respectively. Both form the LDL^T factorization of the augmented matrix (i.e., the matrix appearing on the left-hand side of (3.8)), where L is unit lower triangular and D is block diagonal with 1×1 or 2×2 blocks. This approach is usually (but not always) more stable than the normal equations approach.

In the case which concerns us most, when $\|g^+\|$ converges to zero while $\|v^+\|$ is bounded, an error analysis [4] shows that

$$\frac{\|g^+ - g_c^+\|}{\|g^+\|} \leq \eta \epsilon_m (\sigma_1 + \kappa(A)) \frac{\|v^+\|}{\|g^+\|},$$

where η is the product of a low degree polynomial in $n + m$ with the growth factor from the elimination, while σ_1 is the largest singular value of A. It is interesting to compare this bound with (4.1). We see that the ratio (4.2) again plays a crucial role in the analysis and that the augmented system approach is likely to give a more accurate solution g^+ than the method of normal equations in this case. This cannot be stated categorically, however, since the size of the factor η is difficult to predict.

The residual update strategy described in section 6 aims at minimizing the size of the ratio (4.2), and, as we will see, has a highly beneficial effect in Algorithm 2.2. Before presenting it, we discuss various iterative refinement techniques designed to improve the accuracy of the projection operation.

5. Iterative refinement. Iterative refinement is known as an effective procedure for improving the accuracy of a solution obtained by a method that is not backwards stable. We will now consider how to use it in the context of our normal equations and augmented system approaches.

[3] The bound assumes that there are no errors in the formation of AA^T and Ar^+ or in the backsolves using the Cholesky factors; this is a reasonable assumption in our context [29, section 19.4] provided that $\epsilon_m \kappa^2(A)$ is somewhat smaller than 1. It also ignores less significant errors that arise in the computation of the matrix-vector product $A^T v^+$ and in the subtraction $r^+ - A^T v^+$ given in (3.5).

5.1. Normal equations approach. Let us suppose that we choose $G = I$ and that we compute the projection $P_A r^+$ via the normal equations approach (3.5)–(3.6). An appealing idea for trying to improve the accuracy of this computation is to apply the projection repeatedly. Therefore, rather than computing $g^+ = P_A r^+$ in (2.16), we let $g^+ = P_A \cdots P_A r^+$, where the projection is applied as many times as necessary to keep the errors small. The motivation for this *multiple projections technique* stems from the fact that the computed projection $g_c^+ = (P_A r^+)_c$ is likely to have only a small component, consisting almost entirely of rounding errors, outside of the null space of A. Therefore, applying the projection P_A to the first projection g_c^+ will give an improved estimate because the ratio (4.2) will now be much smaller. By repeating this process we may hope to obtain further improvement of accuracy.

The multiple projection technique may simply be described as setting $g_0^+ = r^+$ and applying the following algorithm.

ALGORITHM 5.1 (multiple projections—normal equations). *Set $i = 0$ and repeat the following steps until a convergence test is satisfied:*

$$\textit{solve} \quad L(L^T v_i^+) = A g_i^+, \tag{5.1}$$

$$\textit{set} \quad g_{i+1}^+ = g_i^+ - A^T v_i^+, \tag{5.2}$$

$$i \leftarrow i + 1,$$

where L is the Cholesky factor of AA^T.

We note that this method is only appropriate when $G = I$, although a simple variant is possible when G is diagonal. Also note that the multiple projection technique is equivalent to performing fixed-precision iterative refinement on the normal equations. In the multiple projections approach, the projection g^+ is updated at each iteration. In fixed-precision iterative refinement of the normal equations, the solution of the normal equations v^+ is updated and the projection g^+ is recomputed from this solution.

We resolved the problem given in Example 3.1 using multiple projections and setting $G = I$. At every CG iteration, we measured the cosine (3.13) of the angle between g and the columns of A. If this cosine was greater than 10^{-12}, multiple projections were applied until the cosine was smaller than this value. Using this strategy, we were able to reduce the norm of the null space component of the residual to around 10^{-16} of its initial value.

In the optimization setting we would apply multiple corrections only when needed, e.g., when the angle between the projected residual and the columns of A is not very small; see Algorithm 6.2 in section 6.1.

It is straightforward to analyze the multiple projections strategy (5.1)–(5.2) provided that, as before, we make the simplifying assumptions that A has norm one and that the only rounding errors we make are in forming L and solving (5.1). In this case, we have that

$$\|(g_{i+1}^+)_c - g^+\| \le \|\Delta v_i^+\| \le \left(\gamma \epsilon_m \kappa^2(A)\right)^i \|v^+\|, \tag{5.3}$$

and thus that the error converges R-linearly to zero with constant $\gamma \epsilon_m \kappa^2(A)$, so long as this factor is less than 1. Of course, the reduction in error at this rate cannot be sustained indefinitely, as the other errors we have ignored in (5.1)–(5.2) become important. Nonetheless, one would expect (5.3) to reflect the true behavior until $\|(g_{i+1}^+)_c - g^+\|$ approaches a small multiple of the unit roundoff ϵ_m. It should

be stressed, however, that this approach is still limited by the fact that the condition number of A appears squared in (5.3); improvement can be guaranteed only if $\gamma\epsilon_m\kappa^2(A) < 1$.

We should also note that multiple projections are almost identical in their form and numerical properties to *fixed precision iterative refinement to the least squares problem* [5, p. 125]. Since a perturbation analysis of the least squares problem [5, Theorem 1.4.6] gives

$$\|g^+ - g_c^+\| = O\left(\epsilon_m(\|v\| + \kappa(A)\|g^+\|)\right), \tag{5.4}$$

and as the dependence here on the condition number is linear—not quadratic as we have seen for (4.1)—we may deduce that the normal equations approach is not backward stable [5, section 2.2]. Indeed, since $\kappa(A)$ is multiplied by $\|g^+\|$, when g^+ is small the effect of the condition number of A is much smaller in (5.4) than in (4.1). It is precisely under such circumstances that fixed-precision iterative refinement is most appropriate [5, section 2.9.3].

We should mention two other iterative refinement techniques that one might consider which are either not effective or not practical in our context.

The first is to use fixed-precision iterative refinement [5, section 2.9] to attempt to improve the solution v^+ of the normal equations (3.6). This, however, will generally be unsuccessful because fixed-precision iterative refinement improves only a measure of backward stability [22, p. 126], and the Cholesky factorization is already a backward stable method. We have performed numerical tests and found no improvement from this strategy.

However, as is well known, iterative refinement will often succeed if extended precision is used to evaluate the residuals. We could therefore consider using extended precision iterative refinement to improve the solution v^+ of the normal equations (3.6). So long as $\epsilon_m\kappa(A)^2 < 1$, and the residuals of (3.6) are smaller than one in norm, we can expect that the error in the solution of (3.6) will decrease by a factor $\epsilon_m\kappa(A)^2$ until it reaches $O(\epsilon_m)$. However, since optimization algorithms normally use double precision arithmetic for all their computations, extending the precision may not be simple or efficient, and this strategy is not suitable for general purpose software.

For the same reason we will not consider the use of extended precision in (5.1)–(5.2) or in the iterative refinement of the least squares problem.

5.2. Augmented system approach. We can apply fixed precision iterative refinement to the solution obtained from the augmented system (3.11). This gives the following iteration.

ALGORITHM 5.2 (iterative refinement—augmented system). *Repeat the following steps until a convergence test is satisfied.*

$$\begin{aligned} &\textit{Compute } \rho_g = r^+ - Gg^+ - A^Tv^+ \textit{ and } \rho_v = -Ag^+, \\ &\textit{solve } \begin{pmatrix} G & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta g^+ \\ \Delta v^+ \end{pmatrix} = \begin{pmatrix} \rho_g \\ \rho_v \end{pmatrix}, \\ &\textit{and update } g^+ \leftarrow g^+ + \Delta g^+ \textit{ and } v^+ \leftarrow v^+ + \Delta v^+. \end{aligned}$$

Note that this method is applicable for general preconditioners G. The general analysis of Higham [30, Theorem 3.2] indicates that, if the condition number of A is not too large, we can expect high relative accuracy in v^+ and good absolute accuracy in g^+ in most cases.

We solved the problem given in Example 3.1 using this iterative refinement technique. As before, we measured the angle between g and the columns of A at every CG iteration. Iterative refinement was applied so long as the cosine of this angle was greater than 10^{-12}. We observed that $\sqrt{r^T g}$ decreased almost as much as with the multiple projections approach.

In our experience, one iterative refinement step is normally enough to provide good accuracy, but we have encountered cases in which two or three steps are beneficial. As in the case of the multiple projections using the normal equations, we would apply this refinement technique selectively in optimization algorithms.

6. Residual update strategy. We have seen that significant roundoff errors may occur in the computation of the projected residual g^+ if this vector is much smaller than the residual r^+. As discussed in the paragraph preceding Example 3.1, the reason for this error is cancellation. We now describe a procedure for redefining r^+ so that its norm is closer to that of g^+. This will dramatically reduce the roundoff errors in the projection operation and thus nearly eliminate the need to use iterative refinement.

We begin by noting that the iterates x of Algorithm 2.2 are theoretically unaffected if, immediately after computing r^+ in (2.15), we redefine it as

$$r^+ \leftarrow r^+ - A^T y \tag{6.1}$$

for some $y \in \mathbf{R}^m$. This equivalence is due to the fact that r^+ appears only in (2.16) and (2.17) and that we have both $PA^T y = 0$, and $(g^+)^T A^T y = 0$. It follows that we can redefine r^+ by means of (6.1) in either the normal equations approach (3.4) and (3.9) or in the augmented system approach (3.8) and (3.11), and the results would, in theory, be unaffected.

Having this freedom to redefine r^+, we seek the value of y that minimizes

$$\|r^+ - A^T y\|_{G^{-1}}, \tag{6.2}$$

where $\|\cdot\|_{G^{-1}}$ is the dual (semi-) norm to the norm $s^T G s$ defined on the manifold $As = 0$, and where we require that G is positive definite over this manifold (see [14]). This dual norm is convenient, since the vector y that solves (6.2) is precisely $y = v^+$ from (3.11). This gives rise to the following modification of the CG iteration.

ALGORITHM 6.1 (preconditioned CG with residual update). *Choose an initial point x satisfying $Ax = b$ and compute $r = Hx + c$. Find the vector y that minimizes $\|r - A^T y\|_{G^{-1}}$; this can be done by solving* (6.2) *and setting $y \leftarrow v^+$. Set $r \leftarrow r - A^T y$, compute $g = Pr$, and set $p = -g$. Repeat the following steps until a convergence test is satisfied:*

$$\alpha = r^T g / p^T H p, \tag{6.3}$$

$$x \leftarrow x + \alpha p, \tag{6.4}$$

$$r^+ = r + \alpha H p, \tag{6.5}$$

compute y that minimizes (6.2), (6.6)

$$r^+ \leftarrow r^+ - A^T y, \tag{6.7}$$

$$g^+ = P r^+, \tag{6.8}$$

$$\beta = (r^+)^T g^+ / r^T g, \tag{6.9}$$

$$p \leftarrow -g^+ + \beta p, \tag{6.10}$$

$$g \leftarrow g^+ \quad \textit{and} \quad r \leftarrow r^+. \tag{6.11}$$

This procedure can be improved by adding iterative refinement of the projection operation in (6.8). In this case, at most one or two iterative refinement steps should be used. The added cost of this algorithm is the storage and computation of y each iteration.

Notice that there is a simple interpretation of steps (6.6)–(6.8). We first obtain y by solving (6.2), and as we have indicated the required value is $y = v^+$ from (3.11). However, (3.11) may be rewritten as

$$\begin{pmatrix} G & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} g^+ \\ 0 \end{pmatrix} = \begin{pmatrix} r^+ - A^T v^+ \\ 0 \end{pmatrix}, \tag{6.12}$$

and thus when we obtain g^+ in step (6.8), it is as if we had instead found it by solving

$$\begin{pmatrix} G & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} g^+ \\ u^+ \end{pmatrix} = \begin{pmatrix} r^+ - A^T v^+ \\ 0 \end{pmatrix}. \tag{6.13}$$

Comparing (6.12) and (6.13), it follows that $u^+ = 0$ in exact arithmetic, although all we can expect in floating point arithmetic is that the computed u^+ will be very small, provided of course that (6.13) is solved in a stable fashion. The advantage of using (6.13) compared to (3.11) is that the solution in the latter may be dominated by the large components v^+, while in the former g^+ are the (relatively) large components, and thus we can expect to find them with high relative accuracy if (6.13) is solved in a stable fashion. Viewed in this way, we see that steps (6.6)–(6.8) are actually a limited form of iterative refinement in which the computed v^+, but not the computed g^+ which is discarded, is used to refine the solution. This "iterative semirefinement" has been used in other contexts [8, 23].

There is another interesting interpretation of the reset $r \leftarrow r - A^T y$ performed at the start of Algorithm 6.1. In the parlance of optimization, $r = Hx + c$ is the gradient of the objective function (1.1) and $r - A^T y$ is the gradient of the Lagrangian for the problem (1.1)–(1.2). The vector y computed from (6.2) is called the least squares Lagrange multiplier estimate. (It is common, but not always the case, for optimization algorithms to set $G = I$ in (6.2) to compute these multipliers.) Thus in Algorithm 6.1 we propose that the initial residual be set to the current value of the gradient of the Lagrangian, as opposed to the gradient of the objective function.

One could ask whether it is sufficient to do this resetting of r at the beginning of Algorithm 6.1 and omit steps (6.6)–(6.7) in subsequent iterations. Our computational experience shows that, even though this initial resetting of r causes the first few CG iterations to take place without significant errors, deviations from the null space due to rounding errors arise in subsequent iterations. The strategy proposed in Algorithm 6.1 is safe in that it ensures that r is small at every iteration.

As it stands, Algorithm 6.1 would appear to require two products with P, or, at the very least, one with P to perform (6.8) and some other means, such as (3.6), to determine y. As we shall now see, this need not be the case.

6.1. The case $G = I$. There is a particularly efficient implementation of the residual update strategy when $G = I$. We can redefine r without the extra cost of storing and computing y as required by Algorithm 6.1. In Algorithm 6.2 below, we present the residual update for $G = I$, combined with iterative refinement. It is this algorithm that is used in the numerical tests in section 7.

ALGORITHM 6.2 (residual update and iterative refinement for $G = I$). *Choose an initial point x satisfying $Ax = b$, compute $r = Hx + c$, $r \leftarrow Pr$, $g \leftarrow Pr$, where*

the projection is computed by the normal equations (3.4) *or augmented system* (3.8) *approaches, and set* $p = -g$. *Choose a tolerance* $\theta_{\max}$. *Repeat the following steps until a convergence test is satisfied:*

$$\alpha = r^T g / p^T H p, \tag{6.14}$$

$$x \leftarrow x + \alpha p, \tag{6.15}$$

$$r^+ = r + \alpha H p, \tag{6.16}$$

$$g^+ = P r^+, \tag{6.17}$$

apply iterative refinement to Pr^+, *if necessary,*

until (3.13) *is less than* $\theta_{\max}$,

$$\beta = (r^+)^T g^+ / r^T g, \tag{6.18}$$

$$p \leftarrow -g^+ + \beta p, \tag{6.19}$$

$$g \leftarrow g^+ \quad \textit{and} \quad r \leftarrow g^+. \tag{6.20}$$

This algorithm was derived from noting that (6.2) is precisely the objective of the least squares problem (3.7) that occurs when computing Pr^+ via the normal equations approach, and therefore the desired value of y is nothing other than the vector v^+ in (3.6) or (3.8). Furthermore, the first block of equations in (3.8) shows that $r^+ - A^T v^+ = g^+$. Therefore, when $G = I$ the computation (6.7) can be replaced by $r^+ \leftarrow Pr^+$ and (6.8) is $g^+ = Pr^+$. In other words, we have applied the projection operation twice, and this is a special case of the multiple projections approach described in the previous section.

Further, (6.7) can be written as $r^+ \leftarrow Pr^+$, or $r^+ = Pr + PH\alpha p$, and therefore (6.8) is

$$g^+ = P(Pr + PH\alpha p). \tag{6.21}$$

As the CG iteration progresses we can expect αp, but not r, to become small. Therefore, we will apply the projection twice to r but only once to $H\alpha p$. Thus (6.21) is replaced by

$$g^+ = P(Pr + H\alpha p), \tag{6.22}$$

which is mathematically equivalent to (6.21), since $PP = P$. This expression is convenient because the term Pr was computed at the previous CG iteration, and therefore we can obtain (6.22) by simply setting $r \leftarrow g^+$ in (6.11) instead of $r \leftarrow r^+$.

Also note that the numerator in the definition (6.3) of α now becomes $g^T g$, which equals $r^T P g = r^T g$. Thus the formula for α is theoretically the same as in Algorithm 6.1, but the symmetric form $\alpha = g^T g / p^T H p$ has the advantage that its numerator can never be negative, as is the case with (6.3) when rounding errors dominate the projection operation.

We solved the problem given in Example 3.1 using this residual update strategy with $G = I$. Both the normal equations and augmented system approaches were equally effective in this case. The cosine (3.13) of the angle between the preconditioned residual and the columns of A remained very small as the computation proceeded. For the normal equations approach this cosine was of order 10^{-14} throughout the CG iteration; for the augmented system approach it was of order 10^{-15}. We also noted that we were able to obtain higher accuracy than with the iterative refinement strategies described in the previous section.

6.2. General G. We can also improve upon the efficiency of Algorithm 6.1 for general G using slightly outdated information. The idea is simply to use the v^+ obtained when computing g^+ in (6.8) as a suitable y rather than waiting until after the following step (6.5) to obtain a slightly more up-to-date version. The resulting iteration is as follows.

ALGORITHM 6.3 (residual update strategy for general G). *Apply Algorithm* 6.1 *with the following two changes:*

omit (6.6)–(6.7),

replace (6.11) *by* $g \leftarrow g^+$ *and* $r \leftarrow r^+ - A^T v^+$, *where* v^+ *is obtained as a by-product when using* (3.11) *to compute* (6.8).

Thus a single projection in step (6.8) is needed for each iteration. Notice, however, that for general G, the extra matrix-vector product $A^T v^+$ will be required, since we no longer have the relationship $g^+ = r^+ - A^T v^+$ that we exploited when $G = I$. Although we have not experimented on this idea for this paper, it has proved to be beneficial in other similar circumstances [23] and provides the backbone for the developing HSL [32] nonconvex quadratic programming packages HSL_VE12 [14] (interior-point) and HSL_VE19 [25] (active set). See also [34] for a thorough discussion of existing and new preconditioners along these lines and the results of some comparative testing.

7. Numerical results. We now test the efficacy of the techniques proposed in this paper on a collection of quadratic programs of the form (1.1)–(1.2). The problems were generated during the last iteration of the interior point method for nonlinear programming described in [9] when this method was applied to a set of test problems from the CUTE [6] collection. We apply the CG method without preconditioning, i.e., with $G = I$, to solve these quadratic programs.

We use the augmented system and normal equations approaches to compute projections, and for each we compare the standard CG iteration (stand), given by Algorithm 2.2, with the iterative refinement (ir) techniques described in section 5 and the residual update strategy combined with iterative refinement (update) as given in Algorithm 6.2. The results are given in Table 7.1. The first column gives the problem name and the second gives the dimension of the quadratic program. To test the reliability of the techniques proposed in this paper we used a very demanding stopping test: the CG iteration was terminated when $\sqrt{r^T g} \leq 10^{-12}$. This stopping test would not be used in practice; rather, we wanted to observe the level of accuracy that could be achieved with each approach.

In these experiments we included several other stopping tests in the CG iteration that are typically used by trust region methods for optimization. We terminate if the number of iterations exceeds $2(n - m)$, where $n - m$ denotes the dimension of the reduced system (2.4); a superscript ^{1} in Table 7.1 indicates that this limit was reached. The CG iteration was also stopped if the length of the solution vector is greater than a "trust region radius" that is set by the optimization method (see [9]). We use a superscript ^{2} to indicate that this safeguard was activated, and note that in these problems only excessive rounding errors can trigger it. Finally we terminate if $p^T H p < 0$, indicated by ^{3}, or if significant rounding error resulted in $r^T g < 0$, indicated by ^{4}. The presence of any superscript indicates that the residual test $\sqrt{r^T g} \leq 10^{-12}$ was not met. Note that the standard CG iteration was not able to meet the residual stopping test for any of the problems in Table 7.1 but that iterative refinement and update residual were successful in most cases.

Table 7.2 reports the CPU time for the problems in Table 7.1. Note that the times for the standard CG approach (stand) should be interpreted with caution, since

TABLE 7.1

Number of CG iterations for the different approaches. [1] *indicates that the iteration limit was reached,* [2] *indicates termination from trust region bound,* [3] *indicates negative curvature was detected, and* [4] *indicates that* $r^T g < 0$.

		Augmented system			Normal equations		
Problem	dim	stand	ir	update	stand	ir	update
CORKSCRW	147	16^2	9	10	4^4	9	11
COSHFUN	61	124^1	124^1	58	124^1	124^1	55
DIXCHLNV	50	91	12	12	5^4	12	12
DTOC3	999	18^4	6	6	2000^1	6	6
DTOC6	1000	6^4	16	16	2^4	16	16
HAGER4	1000	193^4	350	348	1057^4	351	349
HIMMELBK	10	22^1	3	3	7^4	3	3
NGONE	97	0^4	67	56	0^4	65	60
OPTCNTRL	9	20^4	12	4	20^1	2	5
OPTCTRL6	39	90^1	80^1	16	80^1	80^1	16
OPTMASS	402	0^4	5	6	9^3	5	5
ORTHREGA	261	13^4	16^3	16^3	14^3	16^3	16^3
ORTHREGF	805	8^4	18	18	7^4	18	18
READING1	101	3^4	5	5	3^4	5	5

TABLE 7.2

CPU time in seconds. [1] *indicates that the iteration limit was reached,* [2] *indicates termination from trust region bound,* [3] *indicates negative curvature was detected, and* [4] *indicated that* $r^T g < 0$.

		Augmented system			Normal equations		
Problem	dim	stand	ir	update	stand	ir	update
CORKSCRW	147	0.85^2	1.18	0.88	0.15^4	0.74	0.70
COSHFUN	61	0.37^1	0.66^1	0.18	0.29^1	0.54^1	0.13
DIXCHLNV	50	1.90	0.49	0.30	0.2^4	0.50	0.30
DTOC3	999	0.48^4	0.9	0.60	148.48^1	0.91	0.47
DTOC6	1000	0.32^4	1.51	0.9	0.08^4	1.16	0.66
HAGER4	1000	14.23^4	54.43	34.30	70.57^4	40.48	24.71
HIMMELBK	10	0.13^1	0.07	0.04	0.03^4	0.05	0.04
NGONE	97	0.16^4	21.19	10.69	0.98^4	125.24	77.35
OPTCNTRL	9	0.06^4	0.20	0.06	0.05^1	0.28	0.07
OPTCTRL6	39	0.36^1	0.65^1	0.08	0.29^1	0.45^1	0.06
OPTMASS	402	0.06^4	0.57	0.43	0.34^3	0.38	0.25
ORTHREGA	261	0.98^4	2.02^3	1.14^3	0.91^3	2.52^3	1.88^3
ORTHREGF	805	0.46^4	1.84	1.06	1.14^4	5.65	2.95
READING1	101	0.24^4	0.92	0.40	0.29^4	1.31	0.85

in some of these problems it terminated prematurely. We include the times for this standard CG iteration only to show that the iterative refinement and residual update strategies do not greatly increase the cost of the CG iteration.

Next we report on three problems for which the stopping test $\sqrt{r^T g} \leq 10^{-12}$ could not be met by any of the variants. For these three problems, Table 7.3 provides the least residual norm attained for each strategy.

As a final but indirect test of the techniques proposed in this paper, we report the results obtained with KNITRO (an interior point nonlinear optimization code described in [9]) on 29 nonlinear programming problems from the CUTE collection. This code applies the projected CG method to solve a quadratic program at each iteration. The CG iteration was terminated when $\sqrt{r^T g} \leq 0.1\sqrt{r_0^T g_0}$, which is much less stringent than the termination tests used above. We used the augmented system and normal equations approaches to compute projections, and for each of these strategies we tried the standard CG iteration (stand) and the residual update strategy (update)

TABLE 7.3
The least residual norm $\sqrt{r^T g}$ attained by each option.

Problem	dim	Augmented system			Normal equations		
		stand	ir	update	stand	ir	update
OBSTCLAE	900	2.3D-07	1.5D-07	5.5D-08	2.3D-07	9.9D-08	4.2D-08
SVANBERG	500	1.8D-07	9.9D-10	5.7D-12	7.7D-08	8.8D-10	2.9D-10
TORSION1	400	3.5D-09	3.5D-09	2.8D-09	5.5D-08	4.6D-08	3.2D-09

TABLE 7.4
Number of function evaluations and projections required by the optimization method for different variants of the CG iteration. n denotes the number of variables, m the number of general constraints (equalities or inequalities), excluding simple bounds, "st" is the standard CG method, and "up" includes residual updates.

Problem	n	m	Augmented system				Normal equations			
			f evals		projections		f evals		projections	
			st	up	st	up	st	up	st	up
CORKSCRW	456	350	64	61	458	422	65	61	460	411
COSHFUN	61	20	44	40	2213	1025	49	40	2998	1025
DIXCHLNV	100	50	19	19	83	83	19	19	83	83
GAUSSELM	14	11	25	26	92	93	28	41	85	97
HAGER4	2001	1000	18	18	281	281	50	18	2458	281
HIMMELBK	24	14	33	33	88	89	39	33	135	89
NGONE	100	1273	216	133	1763	864	217	187	1821	1146
OBSTCLAE	1024	0	26	26	6233	6068	26	26	6236	6080
OPTCNTRL	32	20	41	51	152	183	***	50	***	179
OPTMASS	1210	1005	36	39	129	145	218	39	427	145
ORTHREGF	1205	400	30	30	73	73	30	30	73	73
READING1	202	100	40	40	130	130	43	40	151	130
SVANBERG	500	500	35	35	7809	4265	40	35	10394	4764
TORSION1	484	0	19	19	2174	2140	19	19	2449	2120
DTOC2	2998	1996	6	6	215	215	6	6	215	215
DTOC3	2999	1998	7	7	16	16	26	7	73	16
DTOC4	2999	1998	5	5	8	8	5	5	8	8
DTOC5	1999	999	6	6	12	12	6	6	12	12
DTOC6	2001	1000	12	12	48	46	64	12	166	46
EIGENA2	110	55	4	4	4	4	4	4	4	4
EIGENC2	464	231	25	25	264	268	25	25	270	269
GENHS28	300	298	4	4	7	7	4	4	7	7
HAGER2	2001	1000	5	5	12	12	5	5	12	12
HAGER3	1001	500	4	4	9	9	4	4	9	9
OPTCTRL6	122	80	14	10	97	75	75	10	880	75
ORTHREGA	517	256	8	8	38	38	***	48	***	99
ORTHREGC	505	250	10	10	60	60	10	10	60	60
ORTHREGD	203	100	11	11	23	23	11	11	23	23

with iterative refinement described in Algorithm 6.2. Now we were concerned with reducing feasibility errors in the CG iterates, not to be able to satisfy a stringent CG termination test, but to ensure that the outer optimization algorithm would converge. The results are given in Table 7.4, where "fevals" denotes the total number of evaluations of the objective function of the nonlinear problem, and "projections" represents the total number of times that a projection operation was performed during the optimization. A *** indicates that the optimization algorithm was unable to locate the solution.

Note that the total number of function evaluations is roughly the same for all strategies, but there are a few cases where the differences in the CG iteration cause the algorithm to follow a different path to the solution. This is to be expected when

solving nonlinear problems. Note that for the augmented system approach, the residual update strategy changes the number of projections significantly only in a few problems, but when it does the improvements are very substantial. On the other hand, we observe that for the normal equations approach (which is more sensitive to the condition number $\kappa(A)$) the residual update strategy gives a substantial reduction in the number of projections in about half of the problems. It is interesting that with the residual update, the performance of the augmented system and normal equations approaches is very similar.

8. Conclusions. We have studied the properties of the projected CG method for solving quadratic programming problems of the form (1.1)–(1.2). Due to the form of the preconditioners used by some nonlinear programming algorithms we opted for not computing a basis Z for the null space of the constraints but instead projecting the CG iterates using a normal equations or augmented system approach. We have given examples showing that in either case significant roundoff errors can occur and have presented an explanation for this.

We proposed several remedies. One is to use iterative refinement of the augmented system or normal equations approaches. An alternative is to update the residual at every iteration of the CG iteration, as described in section 6. The latter can be implemented particularly efficiently in the unpreconditioned ($G = I$) case.

Our numerical experience indicates that updating the residual almost always suffices to keep the errors to a tolerable level. Iterative refinement techniques are not as effective by themselves as the update of the residual but can be used in conjunction with it, and the numerical results reported in this paper indicate that this combined strategy is both economical and accurate. The techniques described here are important ingredients within the evolving large scale nonlinear programming packages KNITRO and GALAHAD, as well as the HSL [32] QP modules HSL_VE12 and HSL_VE19.

Acknowledgments. The authors would like to thank Andy Conn and Philippe Toint for their helpful input during the early stages of this research. They are also grateful to Margaret Wright and two anonymous referees for their helpful suggestions.

REFERENCES

[1] M. Arioli, J.W. Demmel, and I.S. Duff, *Solving sparse linear systems with sparse backward errors*, SIAM J. Matrix Anal. Appl., 10 (1989), pp. 165–190.
[2] 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.
[3] O. Axelsson. *Iterative Solution Methods*, Cambridge University Press, Cambridge, UK, 1996.
[4] Å. Björck, *Pivoting and stability in augmented systems*, in Numerical Analysis 1991, D.F. Griffiths and G.A. Watson, eds., Pitman Res. Notes Math. Ser. 260, Longman Scientific and Technical, Harlow, UK, 1992, pp. 1–16.
[5] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.
[6] I. Bongartz, A.R. Conn, N.I.M. Gould, and Ph. L. Toint, *CUTE: Constrained and unconstrained testing environment*, ACM Trans. Math. Software, 21 (1995), pp. 123–160.
[7] J.R. Bunch and L.C. Kaufman, *Some stable methods for calculating inertia and solving symmetric linear equations*, Math. Comput., 31 (1977), pp. 163–179.
[8] P. Businger and G.H. Golub, *Linear least squares solutions by Housholder transformations*, Numer. Math., 7 (1965), pp. 269–276.
[9] R.H. Byrd, M.E. Hribar, and J. Nocedal, *An interior point algorithm for large-scale nonlinear programming*, SIAM J. Optim., 9 (1999), pp. 877–900.
[10] T.F. Coleman, *Linearly constrained optimization and projected preconditioned conjugate gradients*, in Proceedings of the Fifth SIAM Conference on Applied Linear Algebra, J. Lewis, ed., SIAM, Philadelphia, 1994, pp. 118–122.

[11] T.F. Coleman and A. Pothen, *The null space problem* I: *Complexity*, SIAM J. Algebraic Discrete Methods, 7 (1986), pp. 527–537.
[12] T.F. Coleman and A. Pothen, *The null space problem* II: *Algorithms*, SIAM J. Algebraic Discrete Methods, 8 (1987), pp. 544–563.
[13] T.F. Coleman and A. Verma, *A Preconditioned Conjugate Gradient Approach to Linear Equality Constrained Minimization*, Technical report, Department of Computer Sciences, Cornell University, Ithaca, NY, 1998.
[14] A.R. Conn, N.I.M. Gould, D. Orban, and Ph. L. Toint, *A primal-dual trust-region algorithm for non-convex nonlinear programming*, Math. Program., 87 (2000), pp. 215–249.
[15] J.E. Dennis Jr., M. El-Alem, and M.C. Maciel, *A global convergence theory for general trust-region based algorithms for equality constrained optimization*, SIAM J. Optim., 7 (1997), pp. 177–207.
[16] J.E. Dennis, M. Heinkenschloss, and L.N. Vicente, *Trust-region interior-point SQP algorithms for a class of nonlinear programming problems*, SIAM J. Control Optim., 36 (1998), pp. 1750–1794.
[17] I.S. Duff and J.K. Reid, *The multifrontal solution of indefinite sparse symmetric linear equations*, ACM Trans. Math. Software, 9 (1983), pp. 302–325.
[18] J.C. Dunn, *Second-order multiplier update calculations for optimal control problems and related large scale nonlinear programs*, SIAM J. Optim., 3 (1993), pp. 489–502.
[19] J.R. Gilbert and M.T. Heath, *Computing a sparse basis for the null-space*, SIAM J. Algebraic Discrete Methods, 8 (1987), pp. 446–459.
[20] 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.
[21] P.E. Gill, W. Murray, and M.H. Wright, *Practical Optimization*, Academic Press, London, 1981.
[22] G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.
[23] N.I.M. Gould, *Iterative methods for ill-conditioned linear systems from optimization*, in Nonlinear Optimization and Related Topics, G. Di Pillo and F. Giannessi, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999, pp. 123–142.
[24] N.I.M. Gould, S. Lucidi, M. Roma, and Ph. L. Toint, *Solving the trust-region subproblem using the Lanczos method*, SIAM J. Optim., 9 (1999), pp. 504–525.
[25] N.I.M. Gould and Ph. L. Toint, *An Iterative Active-Set Method for Large-Scale Quadratic Programming*, Technical report RAL-TR-2001-026, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2001.
[26] M.T. Heath, R.J. Plemmons, and R.C. Ward, *Sparse orthogonal schemes for structural optimization using the force method*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 514–532.
[27] M. Heinkenschloss and L.N. Vicente, *Analysis of Inexact Trust Region Interior-Point SQP Algorithms*, Technical report CRPC-TR95546, Center for Research on Parallel Computers, Houston, TX, 1995.
[28] M.R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 409–436.
[29] N.J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996.
[30] N.J. Higham, *Iterative refinement for linear systems and LAPACK*, IMA J. Numer. Anal., 17 (1997), pp. 495–505.
[31] M.E. Hribar, *Large-Scale Constrained Optimization*, Ph.D. thesis, Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, 1996.
[32] HSL, *A collection of Fortran codes for large scale scientific computation*, 2000.
[33] D. James, *Implicit nullspace iterative methods for constrained least squares problems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 962–978.
[34] C. Keller, *Constraint Preconditioning for Indefinite Linear Systems*, D.Phil. thesis, Oxford University, Oxford, UK, 2000.
[35] C. Keller, N.I.M. Gould, and A.J. Wathen, *Constraint preconditioning for indefinite linear systems*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1300–1317.
[36] M. Lalee, J. Nocedal, and T.D. Plantenga, *On the implementation of an algorithm for large-scale equality constrained optimization*, SIAM J. Optim., 8 (1998), pp. 682–706.
[37] L. Lukšan and J. Vlček, *Indefinitely preconditioned inexact Newton method for large sparse equality constrained nonlinear programming problems*, Numer. Linear Algebra Appl., 5 (1998), pp. 219–247.
[38] S.G. Nash and A. Sofer, *Preconditioning reduced matrices*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 47–68.
[39] J. Nocedal and S.J. Wright, *Numerical Optimization*, Springer-Verlag, Heidelberg, Berlin, New York, 1999.

[40] T.D. Plantenga, *A trust region method for nonlinear programming based on primal interior-point techniques*, SIAM J. Sci. Comput., 20 (1999), pp. 282–305.
[41] R.J. Plemmons and R.E. White, *Substructuring methods for computing the null space of equilibrium matrices*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 1–22.
[42] B.T. Polyak, *The conjugate gradient method in extremal problems*, U.S.S.R. Comput. Math. Math. Phys., 9 (1969), pp. 94–112.
[43] T. Steihaug, *The conjugate gradient method and trust regions in large scale optimization*, SIAM J. Numer. Anal., 20 (1983), pp. 626–637.
[44] J.M. Stern and S.A. Vavasis, *Nested dissection for sparse nullspace bases*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 766–775.
[45] Ph. L. Toint, *Towards an efficient sparsity exploiting Newton method for minimization*, in Sparse Matrices and Their Uses, I.S. Duff, ed., Academic Press, London, 1981, pp. 57–88.
[46] Ph. L. Toint and D. Tuyttens, *On large-scale nonlinear network optimization*, Math. Program. Ser. B, 48 (1990), pp. 125–159.
[47] L.N. Vicente, *Trust-Region Interior-Point Algorithms for a Class of Nonlinear Programming Problems*, Ph.D. thesis, Report TR96-05, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 1995.