On the Solution of Equality Constrained Quadratic Programming Problems Arising in Optimization

Nicholas I. M. Gould* Mary E. Hribar † Jorge Nocedal‡

September 6, 2000

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: conjugate gradient method, quadratic programming, preconditioning, large-scale optimization, iterative refinement.

*Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, England, EU. Email: n.gould@rl.ac.uk. Current reports available from "http://www.numerical.rl.ac.uk/reports/reports.html".

†CAAM Department, Rice University, Houston TX 77005. This author was supported by Department of Energy grant DE-FG02-87ER25047-A004.

‡ECE Department, Northwestern University, Evanston Il 60208. Email: nocedal@ece.nwu.edu. Current reports available from www.ece.nwu.edu/~nocedal. This author was supported by National Science Foundation grant CDA-9726385 and by Department of Energy grant DE-FG02-87ER25047-A004.

1. Introduction

A variety of algorithms for linearly and nonlinearly constrained optimization (e.g., [8, 13, 14, 35, 36]) use the conjugate gradient (CG) method [28] to solve subproblems of the form

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

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

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. This positive definiteness assumption is not needed in trust region methods, but our discussion will also be valid in that context because trust region methods normally terminate the CG iteration as soon as negative curvature is encountered (see [42, 44], and, by contrast, [24]).

The quadratic program (1.1)–(1.2) can be solved by computing a basis Z for the null space of A, using this basis to eliminate the constraints, and then applying the CG method to the reduced problem. This approach has been successfully implemented in various algorithms for large scale optimization (cf. [17, 32, 45]).

In this paper we study how to apply the preconditioned CG method to (1.1)–(1.2) without computing a null-space basis Z. There are two reasons for this. Several optimization algorithms require the solution of two distinct forms of linear systems of equations at every iteration; one to compute least squares Lagrange multipliers and the normal (or feasibility) step, and one to compute a null-space basis Z, which is subsequently used to find the solution of (1.1)–(1.2). The use of Z, and the scaling this implies for the trust-region in trust region methods, leads us to the difficult issue of preconditioning the usually dense reduced Hessian matrix $Z^T HZ$ (see the comments concerning Algorithm 1 in section 2). By bypassing the computation of Z in the way that will be described later on, it is possible to solve only one linear system of equations and significantly reduce the cost of the optimization iteration. The second reason for not wanting to compute Z is that it sometimes gives rise to unnecessary ill-conditioning [10, 11, 18, 26, 40, 43]. Although the carefully constructed null-space basis provided by `LUSOL` [19]) is largely successful in avoiding this potential defect [21], it requires two LU factorizations to compute Z.

We thus contend that it can be very useful for general-purpose optimization codes to provide the option of not computing with a null-space basis, and the development of suitable methods is our goal in this paper. The price to pay for such an alternative is that it can give rise to excessive roundoff errors that can cause the constraints $Ax = b$ not to be satisfied to the desired accuracy, and, ultimately, even to failure of the CG iteration. In this paper we describe iterative refinement techniques that can improve the accuracy of the solution, 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, while the G-norm of the vector x is defined to be $\|x\|_G = \sqrt{x^T G x}$, where G is a given symmetric, positive-definite matrix. We will denote the floating-point *unit roundoff* (or machine precision) by ϵ_m. 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 and linear constraints

A common approach for solving linearly constrained problems is to eliminate the constraints and solve a reduced problem (cf. [20, 38]). 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

$$x^* = A^T x_{\mathrm{A}}{}^* + Z x_{\mathrm{Z}}{}^*, \tag{2.1}$$

for some vectors $x_{\mathrm{A}}{}^* \in \mathbf{R}^m$ and $x_{\mathrm{Z}}{}^* \in \mathbf{R}^{n-m}$. The constraints $Ax = b$ yield

$$AA^T x_{\mathrm{A}}{}^* = b, \tag{2.2}$$

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

$$\underset{x_{\mathrm{Z}}}{\text{minimize}} \quad \tfrac{1}{2} x_{\mathrm{Z}}{}^T H_{\mathrm{ZZ}} x_{\mathrm{Z}} + c_{\mathrm{Z}}{}^T x_{\mathrm{Z}}, \tag{2.3}$$

where

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

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

$$H_{\mathrm{ZZ}} x_{\mathrm{Z}} = -c_{\mathrm{Z}}. \tag{2.4}$$

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 $A^T x_{\mathrm{A}}$ exactly and the tangential component $Z x_{\mathrm{Z}}$ 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. [20]).

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. [2]), and we therefore assume that a preconditioner W_{ZZ} is given. 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 the perfect preconditioner. Based on this formula, we consider in this paper preconditioners of the form

$$W_{ZZ} = Z^T G Z, \tag{2.5}$$

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.

Regardless of how W_{ZZ} is defined, the preconditioned conjugate gradient method applied to (2.4) is as follows (see, e.g. [22, p. 532]).

Algorithm I. Preconditioned CG for Reduced Systems.

Choose an initial point x_Z, compute $r_Z = H_{ZZ}x_Z + c_Z$, $g_Z = (Z^T G Z)^{-1} r_Z$ and $p_Z = -g_Z$. Repeat the following steps, until a termination test is satisfied:

$$\begin{aligned}
\alpha &= r_Z{}^T g_Z / p_Z{}^T H_{ZZ} p_Z && (2.6)\\
x_Z &\leftarrow x_Z + \alpha p_Z && (2.7)\\
r_Z{}^+ &= r_Z + \alpha H_{ZZ} p_Z && (2.8)\\
g_Z{}^+ &= (Z^T G Z)^{-1} r_Z{}^+ && (2.9)\\
\beta &= (r_Z{}^+)^T g_Z{}^+ / r_Z{}^T g_Z && (2.10)\\
p_Z &\leftarrow -g_Z{}^+ + \beta p_Z && (2.11)\\
g_Z &\leftarrow g_Z{}^+ \quad \text{and} \quad r_Z \leftarrow r_Z{}^+ && (2.12)
\end{aligned}$$

This iteration may be terminated, for example, when $r_Z{}^T (Z^T G Z)^{-1} r_Z$ is sufficiently small. Coleman and Verma [12] and Nash and Sofer [37] have proposed strategies for defining the preconditioner $Z^T G Z$ which make use of products involving the null-space basis Z and its transpose.

Once an approximate solution is obtained using Algorithm I, 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 I so that the multiplication by Z and the addition of the term $A^T x_A{}^*$ is performed explicitly in the CG iteration. To do so, we introduce, in the following algorithm, the n-vectors x, r, g, p which satisfy $x = Zx_Z + A^T x_A{}^*$, $Z^T r = r_Z$, $g = Zg_Z$ and $p = Zp_Z$. We also define the scaled projection matrix

$$P = Z(Z^T G Z)^{-1} Z^T. \tag{2.13}$$

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

Algorithm II 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:

$$\alpha = r^T g / p^T H p \tag{2.14}$$

$$x \leftarrow x + \alpha p \tag{2.15}$$

$$r^+ = r + \alpha H p \tag{2.16}$$

$$g^+ = P r^+ \tag{2.17}$$

$$\beta = (r^+)^T g^+ / r^T g \tag{2.18}$$

$$p \leftarrow -g^+ + \beta p. \tag{2.19}$$

$$g \leftarrow g^+ \quad \text{and} \quad r \leftarrow r^+ \tag{2.20}$$

This will be the main algorithm studied in this paper. It is important to notice that this algorithm, unlike its predecessor, is independent of the choice of Z. 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 will use the quantity $r^T g \equiv r^T P r \equiv g^T G g$ to terminate the CG iteration. An initial point satisfying $Ax = b$ can be computed, for example, by solving the normal equations (2.2).

Two simple choices of G are

$$G = \text{diag}(H), \qquad \text{and} \qquad G = I.$$

The first choice is appropriate when H contains some large elements on the diagonal. This is the case, for example, in barrier methods for constrained optimization that handle bound constraints $l \leq x \leq u$ 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 choice $G = I$ arises in several trust region methods for constrained optimization [8, 14, 15, 27, 35, 39, 46]. These methods include a trust region constraint of the form $\|Z x_{\rm Z}\| \leq \Delta$ in the subproblem (2.3). In order to transform it into a spherical constraint, we introduce the change of variables $x_{\rm Z} \leftarrow (Z^T Z)^{-1/2} x_{\rm Z}$ whose effect in the CG iteration is identical to that of defining $Z^T G Z = (Z^T Z)^{-1}$. Since the role of this matrix is not to produce a clustering of the eigenvalues, we will regard Algorithm II with the choice $G = I$ as an *unpreconditioned* CG iteration.

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 II will also lie in 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 the next sections.

3. CG Algorithm Without a Null-Space Basis

We are interested here in using Algorithm II 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 (2.13).

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.,

$$P_{\mathrm{Z}} = Z(Z^TZ)^{-1}Z^T, \tag{3.1}$$

that is, g^+ is the result of projecting r^+ into the null space of A. Thus the preconditioned residual (2.17) can be written as

$$g^+ = P_{\mathrm{Z}} r^+. \tag{3.2}$$

This projection can be performed in two alternative ways.

The first is to replace P_{Z} by the equivalent formula

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

and thus to replace (3.2) with

$$g^+ = P_{\mathrm{A}} r^+. \tag{3.4}$$

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 $g^+ = P_{\mathrm{Z}} r^+$ will be called the *normal equations approach*, and will be implemented in this paper using a Cholesky factorization of AA^T to solve (3.6).

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

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

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.*

Let us suppose now that preconditioning has the more general form

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

This may be expressed as

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

if G is non-singular, 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., [20, 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 [41]; the alternative (3.11), and its special case (3.8), are proposed in [9], although [9] unnecessarily requires that G be positive definite. A more recent study on preconditioning the projected CG method is [12], while the eigenstructure of the preconditioned system is examined by [34, 36].

Interestingly, preconditioning in Coleman and Verma's null-space approach [12] requires the solution of systems like (3.11), but allowing 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 matrix Z and only require 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

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

Unfortunately all three of our proposed alternatives, (3.4), (3.8) and (3.11), for computing g^+ can give rise to significant round-off 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 II, 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)/(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)/(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.

Example 1. Consider the case

$$A = \begin{pmatrix} 1000 & 0 & 1 & 1 \\ 1 & 0.0001 & 0.1 & 0.1 \end{pmatrix}, \qquad r = \begin{pmatrix} 1000001.00001 \\ 0.0001 \\ 1000.1 \\ 1000.1 \end{pmatrix}.$$

The condition number of A is $\kappa(A)$ =7.142E+03, and r has been chosen to lie almost in the range of A^T. The required projection, to 16 significant figures, is

$$g = \begin{pmatrix} 1.020298963475658E\text{-}17 \\ 1.010095973840901E\text{-}11 \\ -5.101494817378290E\text{-}15 \\ -5.101494817378290E\text{-}15 \end{pmatrix}. \tag{3.13}$$

Using the normal equations approach we obtain

$$g = P_{\rm A} r = \begin{pmatrix} -3.0267E\text{-}09 \\ 1.0641E\text{-}11 \\ 7.2054E\text{-}10 \\ 7.2054E\text{-}10 \end{pmatrix}, \tag{3.14}$$

which contains significant errors. To measure the angle between g and the rows of A, we define

$$\cos\theta = \max_i \left\{ \frac{A_i^T g}{||A_i||\ ||g||} \right\} \tag{3.15}$$

where A_i is the i-th row of A. For the value of g given by (3.14), we have $\cos\theta = -0.15$, which is unacceptably large—we note that $\cos\theta$ for (3.13) is 9.6E−21.

Using the augmented system we obtain

$$g = P_{\mathrm{A:I}} r = \begin{pmatrix} -1.117E\text{-}16 \\ 1.010E\text{-}11 \\ -1.705E\text{-}13 \\ -1.705E\text{-}13 \end{pmatrix},$$

which is clearly more accurate than (3.14). Nevertheless, $\cos\theta = -.0032$, indicating that the projection is not acceptable either.

Now consider a more realistic 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.

Example 2. We applied Algorithm II to solve problem CVXEQP3 from the CUTE collection [5], with $n = 1000$ and $m = 750$. 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 1, which plots the residual $\sqrt{r^T g}$ as a function of the iteration number. In both cases the CG iteration was terminated when $r^T g$ became negative, which indicates that severe errors have occurred since $r^T g = {r_{\mathrm{Z}}}^T Z^T Z r_{\mathrm{Z}}$ 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 1 also plots (3.15), 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 indicating 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 II from its predecessor is predicated on the assumption that the search is restricted to this manifold. As we have mentioned, the search direction will lie on the constraint manifold if and only if the cosine (3.15) is zero, and thus it is reasonable to ask that the cosine for a computed approximation to g should be small. The general analysis of Arioli, Demmel and Duff [1], indicates that, with care, it is possible to ensure that the backward error[1]

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

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

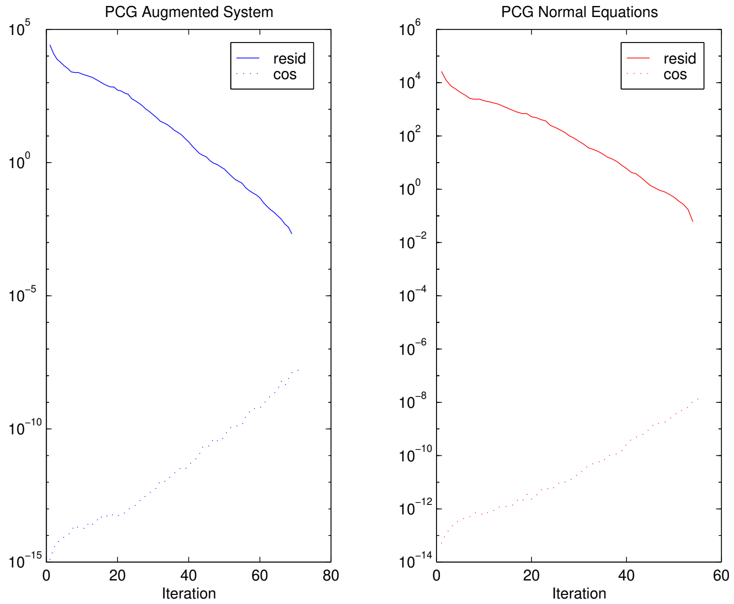

Figure 1: Conjugate gradient method with two options for the projection

of the computed g^+ is of the order of the machine precision, ϵ_m (here $|\cdot|$ denotes the componentwise absolute value). Since the absolute values of the backward error and the cosine (3.15) are quantitatively the same (the former provides an upper bound on the latter), and as we find it easier to interpret (3.15), we shall henceforth aim for approximate solutions for which the cosine is a reasonable multiple of ϵ_m. We have found that asking that (3.15) be smaller than $\epsilon_m^{.75} \approx 10^{-12}$ is sufficient.

Severe errors such as those illustrated in Example 2 are not uncommon in optimization calculations based on Algorithm II. This is of grave 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 as explained in [31, p.33–34], cannot be solved to a prescribed accuracy; see also section 7.

In §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. Analysis of the 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. 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.

Let us first consider the *normal equations approach.* Here $g^+ = P_{\rm A} r^+$ is given by (3.5) where (3.6) is solved by means of the Cholesky factorization of AA^T. In finite precision, instead of the exact solution v^+ of the normal equations we obtain $v_c^+ = v^+ + \Delta v^+$, where the error Δv^+ satisfies [4, p.49][2]

$$\|\Delta v^+\| \leq \gamma \epsilon_m \kappa^2(A) \|v^+\|, \tag{4.1}$$

with $\gamma = 2.5 n^{3/2}$. Recall that ϵ_m denotes unit roundoff and $\kappa(A)$ the condition number of A. The presence of the *square* of the condition number of A on the right hand side is a consequence of the fact that the normal equations were solved.

We can now study the total error in the projection vector g^+. To simplify the analysis, we will ignore the 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), because these errors will be dominated by the error in v^+ whose magnitude is estimated by (4.1). Under these assumptions, we have from (3.5) that the computed projection $g_c^+ = (P_{\rm A} r^+)_c$ and the exact projection $g^+ = P_{\rm A} r^+$ satisfy

$$g^+ - g_c^+ = A^T \Delta v^+, \tag{4.2}$$

and thus the error in the projection lies entirely in the range of A^T. We then have from (4.1) that the relative error in the projection satisfies[3]

$$\frac{\|g^+ - g_c^+\|}{\|g^+\|} \leq \gamma \epsilon_m \kappa^2(A) \frac{\|v^+\|}{\|g^+\|}. \tag{4.3}$$

This error can be significant when $\kappa(A)$ is large or when

$$\frac{\|v^+\|}{\|g^+\|} = \frac{\|v^+\|}{\|P_{\rm A} r^+\|} \tag{4.4}$$

is large.

Let us consider the ratio (4.4) in the case when $\|r^+\|$ is much larger than its projection $\|g^+\|$. We have from (3.5) that $\|r^+\| \approx \|A^T v^+\|$, and by the assumption that $\|A\| = 1$,

$$\|r^+\| \approx \|A^T v^+\| \leq \|v^+\|.$$

Suppose that the inequality above is achieved. Then (4.4) gives

$$\frac{\|v^+\|}{\|g^+\|} \approx \frac{\|r^+\|}{\|P_{\rm A} r^+\|},$$

[2]The bound (4.1) 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. We should also note that (4.1) can be sharpened by replacing the term $\kappa^2(A)$ with $\kappa'(A)\kappa(A)$, where $\kappa'(A) = \min \kappa(AD)$ over all possible diagonal scalings D.

[3]If $\|g^+\|$ is small, it is preferable to replace the denominators in (4.3) by $\max(\|g^+\|, \epsilon)$ where ϵ is a suitable multiple (e.g. 10) of ϵ_m.

which is simpler to interpret than (4.4). We can thus conclude that the error in the projection (4.3) will be large when either $\kappa(A)$ or the ratio $\|r^+\|/\|P_{\mathrm{A}} r^+\|$ is large.

When the condition number $\kappa(A)$ is moderate, the contribution of the ratio (4.4) to the relative error (4.3) is normally not large enough to cause failure of the outer optimization calculation. This is because a typical stopping test in nonlinear optimization algorithms would cause termination when projected residual g^+ is (say) 10^{-6} times smaller in norm than the initial residual. In this case the ratio (4.4) would be roughly 10^6, and using double precision arithmetic one would have sufficient accuracy to make progress toward the solution. But as the condition number $\kappa(A)$ grows, the loss of significant digits becomes severe, especially since $\kappa(A)$ appears squared in (4.3). In Example 2,

$$\gamma = O(10^4) \quad \epsilon_m = 10^{-16} \quad \kappa(A) = O(10^3) \quad \|A\| = O(10)$$

and we have mentioned that the ratio (4.4) is of order $O(10^6)$ at iteration 50. The bound (4.3) indicates that there could be no correct digits in g^+, at this stage of the CG iteration. Even though this bound can often be overly pessimistic, it appears to be reasonably tight in this example, for at this point the CG iteration could make no further progress.

Let us 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

$$\begin{pmatrix} I & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} g^+ \\ v^+ \end{pmatrix} = \begin{pmatrix} r^+ \\ 0 \end{pmatrix} \tag{4.5}$$

using a direct method. There are a number of such methods, the strategies of Bunch and Kaufman [6] and Duff and Reid [16] 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 (4.5)), 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. To improve the stability of the method, Björck [3] suggests replacing the upper-left block of (4.5) by a multiple of the identity I, but since choosing a good value of this parameter can be difficult, we consider here only (4.5).

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

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

It is interesting to compare this bound with (4.3). We see that the ratio (4.4) 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 §6 aims at minimizing the contribution of the ratio (4.4), and as we will see, has a highly beneficial effect in Algorithm II. 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.

5.1. Normal Equations Approach

Let us suppose that we choose $G = I$ and that we compute the projection $P_{\mathrm{A}} r^+$ 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 $g^+ = P_{\mathrm{A}} r^+$ in (2.17), we let $g^+ = P_{\mathrm{A}} \cdots P_{\mathrm{A}} r^+$ 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 $g_c^+ = (P_{\mathrm{A}} r^+)_c$ will have only a small component, consisting entirely of rounding errors, outside of the null space of A, as described by (4.2). Therefore applying the projection P_{A} to the first projection g_c^+ will give an improved estimate because the ratio (4.4) 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.

> **Multiple Projections/Iterative Refinement** (Normal Equations).
>
> Set $i = 0$ and repeat the following steps, until a convergence test is satisfied:
>
> $$\text{solve} \quad L(L^T v_i^+) = A g_i^+ \tag{5.1}$$
> $$\text{set} \quad g_{i+1}^+ = g_i^+ - A^T v_i^+, \tag{5.2}$$
> $$i \leftarrow i + 1, \tag{5.3}$$

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. If we apply the method to the problem given in Example 1, we find that $\cos\theta = 6.2\text{E}{-14}$ after a single refinement, and $9.6\text{E}{-21}$ after a second.

Example 3.

We solved the problem given in Example 2 using multiple projections, and setting $G = I$. At every CG iteration we measure the cosine (3.15) of the angle between g and the columns of A. If this cosine is greater than 10^{-12}, then multiple projections are applied until the cosine is less than this value. The results are given in Figure 2, and show that the residual $\sqrt{r^T g}$ was reduced much more than in the plain CG iteration (Figure 1). Indeed the ratio between the final and initial values of $\sqrt{r^T g}$ is 10^{-16}, which is very satisfactory.

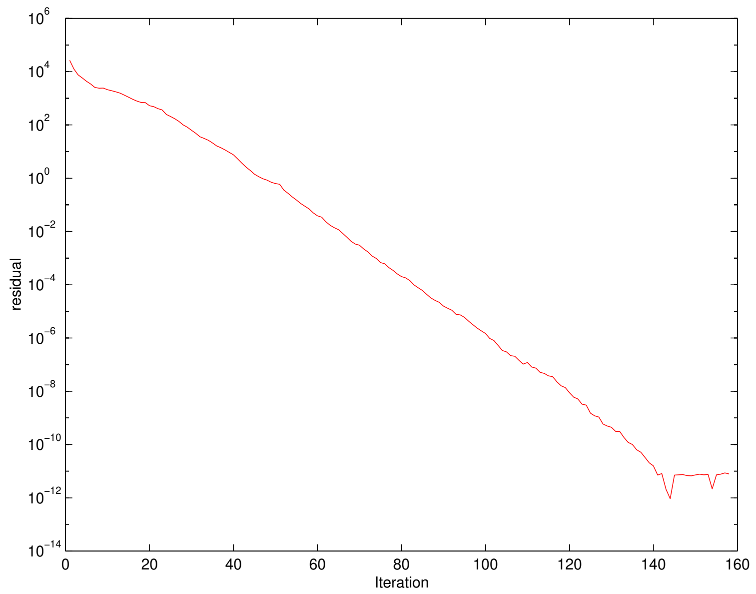

Figure 2: CG method using multiple projections in the normal equations approach.

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 IV 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 assumption that the only rounding errors we make are in forming L and solving (5.1). We obtain the following result which can be proved by induction. For $i = 0, 1, \cdots$,

$$(g_{i+1}^+)_c = g^+ - A^T \Delta v_i^+, \tag{5.4}$$

where as in (4.1)

$$\|\Delta v_i^+\| \leq \gamma \epsilon_m \kappa^2(A) \|v_i^+\|, \qquad \text{and} \qquad v_i^+ = -\Delta v_{i-1}^+. \tag{5.5}$$

A simple consequence of (5.4)–(5.5) and the assumption that A has norm one is that

$$\|(g_{i+1}^+)_c - g^+\| \leq \|\Delta v_i^+\| \leq \left(\gamma \epsilon_m \kappa^2(A)\right)^i \|v^+\|, \tag{5.6}$$

and thus that the error converges R-linearly to zero with rate

$$\gamma \epsilon_m \kappa^2(A), \tag{5.7}$$

as long as (5.7) is less than 1. Of course, this rate can not be sustained indefinitely as the other errors we have ignored in (5.1)–(5.2) become important. Nonetheless, one would expect (5.6) to reflect the true behaviour 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 (5.6); 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* [4, p.125]. Since a perturbation analysis of the least squares problem [4, Theorem 1.4.6]) gives

$$\|g^+ - g_c^+\| = O\left(\epsilon_m(\|v\| + \kappa(A)\|g^+\|)\right), \tag{5.8}$$

and as the dependence here on the condition number is linear—not quadratic as we have seen for (4.3)—we may deduce that the normal equations approach is not backward stable [4, 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 (5.8) than in (4.3). It is precisely under such circumstances that fixed precision iterative refinement is most appropriate [4, Section 2.9.3]).

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

The first is to use fixed-precision iterative refinement [4, 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 only improves 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 $O(\epsilon_m)$. But 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.

Iterative Refinement (Augmented system)

Repeat the following steps until a convergence
test is satisfied.

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

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. A single refinement applied to the problem given in Example 1 yields $\cos\theta = 9.6\text{E}{-21}$.

Example 4.
We solved the problem given in Example 2 using this iterative refinement technique. As in the case of multiple projections discussed in Example 3, we measure the angle between g and the columns of A at every CG iteration. Iterative refinement is applied as long as the cosine of this angle is greater than 10^{-12}. We demand, once more, that the cosine (3.15) be very small to avoid even small violations of infeasibility which can be harmful to an outer optimization algorithm. The results are given in Figure 3.

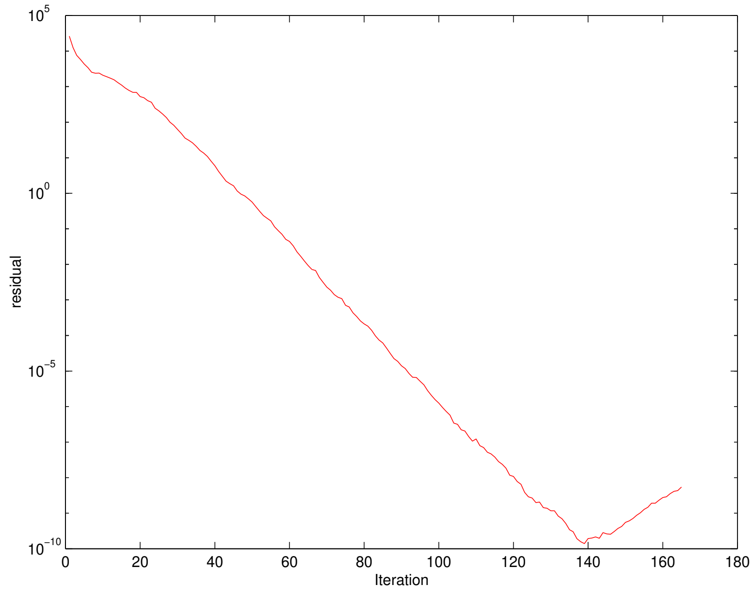

Figure 3: CG method using iterative refinement in the augmented system approach.

We observe that the residual $\sqrt{r^T g}$ is decreased almost as much as with the multiple projections approach, and attains an acceptably small value. We should point out, however, that the residual increases after it reaches the value 10^{-10}, and if the CG iteration is continued for a few hundred more iterations, the residual exhibits large oscillations. We will return to this in §6.1.

In our experience, 1 iterative refinement step is normally enough to provide good accuracy, but we have encountered cases in which 2 or 3 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 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 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.

We begin by noting that Algorithm II is theoretically unaffected if, immediately after computing r^+ in (2.16), 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 r^+ appears only in (2.17) and (2.18), and that we have both $PA^Ty = 0$, 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)/(3.9) or in the augmented system approach (3.8)/(3.11) and the results would, in theory, be unaffected.

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

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

where $\|\cdot\|$ 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 [13]). 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 III Preconditioned CG with Residual Update.

Choose an initial point x satisfying $Ax = b$, compute $r = Hx + c$, and find the vector y that minimizes $\|r - A^T y\|_{G^{-1}}$. Set $r \leftarrow r - A^T y$, compute $g = Pr$ and set $p = -g$. Repeat the following steps, until a convergence test is satisfied:

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

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

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

$$
\begin{array}{rcll}
r^+ & \leftarrow & r^+ - A^T y, & \text{where } y \text{ solves (6.2)} \qquad (6.6)\\
g^+ & = & Pr^+ & \qquad (6.7)\\
\beta & = & (r^+)^T g^+ / r^T g & \qquad (6.8)\\
p & \leftarrow & -g^+ + \beta p & \qquad (6.9)\\
g & \leftarrow & g^+ \quad \text{and} \quad r \ \leftarrow \ r^+. & \qquad (6.10)
\end{array}
$$

This procedure can be improved by adding iterative refinement of the projection operation in (6.7). In this case, at most 1 or 2 iterative refinement steps should be used.

Notice that there is a simple interpretation of Steps (6.6) and (6.7). We first obtain y by solving (6.2), and as we have indicated the required value is $y = v^+$ from (3.11). But (3.11) may be rewritten as

$$
\begin{pmatrix} G & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} g^+ \\ 0 \end{pmatrix} = \begin{pmatrix} r^+ - A^T v^+ \\ 0 \end{pmatrix}, \tag{6.11}
$$

and thus when we obtain g^+ in Step (6.7), it is as if we had instead found it by solving

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

Comparing (6.11) and (6.12), it follows that $u^+ = 0$ in exact arithmetic, although all we can expect in floating point arithmetic is that the computed u^+ will be tiny rounded values, provided of course that (6.12) is solved in a stable fashion. The advantage of using (6.12) 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 (6.12) is solved in a stable fashion. Viewed in this way, we see that Steps (6.6) and (6.7) 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 semi-refinement" has been used in other contexts [7, 23]. For the problem given in Example 1, the resulting g^+ gives $\cos\theta = 9.6\text{E}{-21}$.

There is another interesting interpretation of the reset $r \leftarrow r - A^T y$ performed at the start of Algorithm III. 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 III 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 III, and omit step (6.6) 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, rounding errors arise in subsequent iterations. The strategy proposed in Algorithm III is safe in that it ensures that r is small at every iteration.

As it stands, Algorithm III would appear to require two products with P, or, at the very least, one with P to perform (6.7) and some other means, such as (3.6), to determine v^+. 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 $G = I$. Note 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 (6.6) can be replaced by $r^+ \leftarrow Pr^+$ and (6.7) 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.

Based on these observations we propose the following variation of Algorithm III that requires *only one* projection per iteration. We have noted that (6.6) can be written as $r^+ \leftarrow Pr^+$, or $r^+ = Pr + PH\alpha p$, and therefore (6.7) is

$$g^+ = P(Pr + PH\alpha p). \tag{6.13}$$

As the CG iteration progresses we can expect αp to become small, but as noted earlier, r will not. Therefore we will apply the projection twice to r but only once to $H\alpha p$. Thus (6.13) is replaced by

$$g^+ = P(Pr + H\alpha p). \tag{6.14}$$

which is mathematically equivalent to (6.13) since $PP = P$. This expression is convenient because the term Pr was computed at the previous CG iteration, and therefore we can obtain (6.14) by simply setting $r \leftarrow g^+$ in (6.10) instead of $r \leftarrow r^+$. The resulting iteration is as follows

Residual Update Strategy for $G = I$

Apply Algorithm III with the following two changes:

Omit (6.6)

Replace (6.10) by $g \leftarrow g^+$ and $r \leftarrow g^+$.

This strategy avoids the extra storage and computation required by Algorithm III. In practice, it can also achieve more accuracy than iterative refinement as shown by Example 5 and the numerical results in section 7.

We 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 of α is theoretically the same as in Algorithm, III, 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.

Example 5.

We solved the problem given in Example 2 using this residual update strategy with $G = I$. The results are given in Figure 4 and show that the normal equations and augmented system approaches are equally effective in this case. We do not plot the cosine (3.15) of the angle between the preconditioned residual and the columns of A because it was very small in both approaches, and did not tend to grow as the iteration progressed. 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}. Note that we have obtained higher accuracy than with the iterative refinement strategies described in the previous section; compare with Figures 2 and 3.

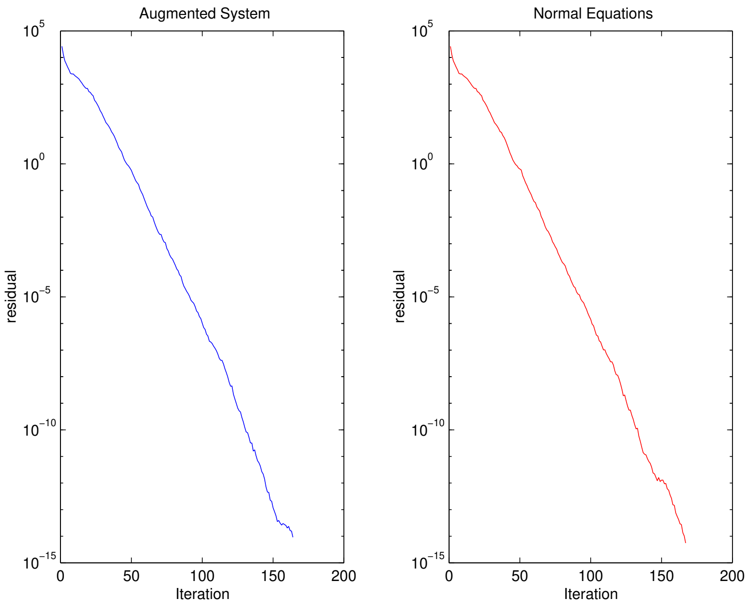

Figure 4: Conjugate gradient method with the residual update strategy.

To obtain a highly reliable algorithm for the case when $G = I$ we can combine the residual update strategy just described with iterative refinement of the projection operation. This gives rise to the following iteration which will be used in the numerical tests reported in §7.

Algorithm IV 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 θ_{max}. Repeat the following steps, until a convergence test is satisfied:

$$\alpha = r^T g / p^T H p \tag{6.15}$$

$$x \leftarrow x + \alpha p \tag{6.16}$$

$$r^+ = r + \alpha H p \tag{6.17}$$

$$g^+ = P r^+ \tag{6.18}$$

$$\text{Apply iterative refinement to } Pr^+\text{, if necessary,} \tag{6.19}$$

$$\text{until (3.15) is less than } \theta_{max} \tag{6.20}$$

$$\beta = (r^+)^T g^+ / r^T g \tag{6.21}$$

$$p \leftarrow -g^+ + \beta p \tag{6.22}$$

$$g \leftarrow g^+ \quad \text{and} \quad r \leftarrow g^+. \tag{6.23}$$

We conclude this discussion by elaborating on the point made before Example 5 concerning the computation of the steplength parameter α. We have noted that the formula $\alpha = g^T g / p^T H p$ is preferable to (6.15) since the numerator cannot give rise to cancellation. Similarly the stopping test should be based on $g^T g$ rather than on $g^T r$. The residual update implemented in Algorithm IV does this change automatically, but we believe that these expressions are to be recommended in other implementations of the CG iteration, provided the preconditioner is based on $G = I$.

To test this, we repeated the computation reported in Example 2 using the augmented system approach; see Figure 1. The only change is that Algorithm II now used the new formulae for α and for the stopping test. The CG iteration was now able to continue past iteration 70 and was able to reach the value $\sqrt{g^T g} = 10^{-8}$. We also repeated the calculation made in Example 4. Now the residual reached the level $\sqrt{g^T g} = 10^{-12}$ and the large oscillations in the residual mentioned in Example 3 no longer took place. Thus in both cases these alternative expressions for α and for the stopping test were beneficial.

6.2. General G

We can also improve upon the efficiency of Algorithm III for general G, using slightly outdated information. The idea is simply to use the v^+ obtained when computing g^+ in (6.7) 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:

Residual Update Strategy for general G

Apply Algorithm III with the following two changes:

Omit (6.6)

Replace (6.10) by $g \leftarrow g^+$ and $r \leftarrow r^+ - A^T v^+$, where v^+ is obtained as a bi-product when using (3.11) to compute (6.7).

Thus a single projection, in step (6.7), 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 Subroutine Library non-convex quadratic programming packages `VE12` [13] (interior-point) and `VE19` [25] (active set). See also [33] 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 [8], when this method was applied to a set of test problems from the CUTE [5] 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 II, with the iterative refinement (ir) techniques described in §5 and the residual update strategy combined with iterative refinement (update) as given in Algorithm IV. The results are given in Table 1. The first column gives the problem name, and the second, 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 $\sqrt{r^T g} \leq 10^{-12}$.

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 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 [8]). 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 1, but that iterative refinement and update residual were successful in most cases.

Table 2 reports the CPU time for the problems in Table 1. Note that the times for the standard CG approach (stand) should be interpreted with caution, since 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 3 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 3 provides the least residual norm attained for each strategy.

Problem	dim	Augmented System stand	ir	update	Normal Equations 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 1: Number of CG iterations for the different approaches. A ^{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 $r^T g < 0$.

Problem	dim	Augmented System stand	Augmented System ir	Augmented System update	Normal Equations stand	Normal Equations ir	Normal Equations 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

Table 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 $r^T g < 0$.

Problem	dim	Augmented System stand	Augmented System ir	Augmented System update	Normal Equations stand	Normal Equations ir	Normal Equations 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 3: The least residual norm: $\sqrt{r^T g}$ attained by each option.

As a final, but indirect test of the techniques proposed in this paper, we report the results obtained with the interior point nonlinear optimization code described in [8] on 29 nonlinear programming problems from the CUTE collection. This code applies the CG method to solve a quadratic program at each iteration. 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) with iterative refinement described in Algorithm IV. The results are given in Table 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 §6. The latter can be implemented particularly efficiently when the preconditioner is given by $G = I$ in (2.5).

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 (`NITRO` and `GALAHAD`, as well as the HSL QP modules `VE12` and `VE19`.

			Augmented System				Normal Equations			
			f evals		projections		f evals		projections	
Problem	n	m	stand	update	stand	update	stand	update	stand	update
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

Table 4: Number of function evaluations and projections required by the optimization method for the different implementations of the CG iteration. n denotes the number of variables and m the number of general constraints (equalities or inequalities), excluding simple bounds.

9. Acknowledgements

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, and to Philip Gill and Michael Saunders for advice on the suitability of their package LUSOL for computing null-space bases.

10. *

References

[1] M. Arioli, J. W. Demmel, and I. S. Duff. Solving sparse linear systems with sparse backward errors. *SIAM Journal on Matrix Analysis and Applications*, 10(1):165–190, 1989.

[2] O. Axelsson. *Iterative Solution Methods.* Cambridge University Press, Cambridge, England, 1996.

[3] Å. Björck. Pivoting and stability in augmented systems. In D. F. Griffiths and G. A. Watson, editors, *Numerical Analysis 1991*, number 260 in Pitman Research Notes in Mathematics Series, pages 1–16, Harlow, England, 1992. Longman Scientific and Technical.

[4] Å. Björck. *Numerical Methods for Least Squares Problems.* SIAM, Philadelphia, USA, 1996.

[5] I. Bongartz, A. R. Conn, N. I. M. Gould, and Ph. L. Toint. CUTE: Constrained and unconstrained testing environment. *ACM Transactions on Mathematical Software*, 21(1):123–160, 1995.

[6] J. R. Bunch and L. C. Kaufman. Some stable methods for calculating inertia and solving symmetric linear equations. *Mathematics of Computation*, 31:163–179, 1977.

[7] P. Businger and G. H. Golub. Linear least squares solutions by Housholder transformations. *Numerische Mathematik*, 7:269–276, 1965.

[8] R. H. Byrd, M. E. Hribar, and J. Nocedal. An interior point algorithm for large scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900, 1999.

[9] T. F. Coleman. Linearly constrained optimization and projected preconditioned conjugate gradients. In J. Lewis, editor, *Proceedings of the Fifth SIAM Conference on Applied Linear Algebra*, pages 118–122, Philadelphia, USA, 1994. SIAM.

[10] T. F. Coleman and A. Pothen. The null space problem I: complexity. *SIAM Journal on Algebraic and Discrete Methods*, 7(4):527–537, 1986.

[11] T. F. Coleman and A. Pothen. The null space problem II: algorithms. *SIAM Journal on Algebraic and Discrete Methods*, 8(4):544–563, 1987.

[12] T. F. Coleman and A. Verma. A preconditioned conjugate gradient approach to linear equality constrained minimization. Technical report, Department of Computer Sciences, Cornell University, Ithaca, New York, USA, July 1998.

[13] A. R. Conn, N. I. M. Gould, D. Orban, and Ph. L. Toint. A primal-dual trust-region algorithm for non-convex nonlinear programming. *Mathematical Programming*, 87(2):215–249, 2000.

[14] J. E. Dennis, M. El-Alem, and M. C. Maciel. A global convergence theory for general trust-region based algorithms for equality constrained optimization. *SIAM Journal on Optimization*, 7(1):177–207, 1997.

[15] J. E. Dennis, M. Heinkenschloss, and L. N. Vicente. Trust-region interior-point SQP algorithms for a class of nonlinear programming problems. *SIAM Journal on Control and Optimization*, 36(5):1750–1794, 1998.

[16] I. S. Duff and J. K. Reid. The multifrontal solution of indefinite sparse symmetric linear equations. *ACM Transactions on Mathematical Software*, 9(3):302–325, 1983.

[17] J. C. Dunn. Second-order multiplier update calculations for optimal control problems and related large scale nonlinear programs. *SIAM Journal on Optimization*, 3(3):489–502, 1993.

[18] J. R. Gilbert and M. T. Heath. Computing a sparse basis for the null-space. *SIAM Journal on Algebraic and Discrete Methods*, 8:446–459, 1987.

[19] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. Maintaining LU factors of a general sparse matrix. *Linear Algebra and its Applications*, 88/89:239–270, 1987.

[20] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, London, 1981.

[21] P. E. Gill and M. A. Saunders. Private communication.

[22] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, third edition, 1996.

[23] N. I. M. Gould. Iterative methods for ill-conditioned linear systems from optimization. In G. Di Pillo and F. Giannessi, editors, *Nonlinear Optimization and Related Topics*, pages 123–142, Dordrecht, The Netherlands, 1999. Kluwer Academic Publishers.

[24] N. I. M. Gould, S. Lucid, M. Roma, and Ph. L. Toint. Solving the trust-region subproblem using the Lanczos method. *SIAM Journal on Optimization*, 9(2):504–525, 1999.

[25] N. I. M. Gould and Ph. L. Toint. An iterative active-set method for large-scale quadratic programming. Technical Report in preparation, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2000.

[26] M. T. Heath, R. J. Plemmons, and R. C. Ward. Sparse orthogonal schemes for structural optimization using the force method. *SIAM Journal on Scientific and Statistical Computing*, 5(3):514–532, 1984.

[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, Texas, USA, 1995.

[28] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, 1952.

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

[30] N. J. Higham. Iterative refinement for linear systems and LAPACK. *IMA Journal of Numerical Analysis*, 17(4):495–505, 1997.

[31] M. E. Hribar. *Large-scale constrained optimization.* PhD thesis, Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, Illinois, USA, 1996.

[32] D. James. Implicit nullspace iterative methods for constrained least squares problems. *SIAM Journal on Matrix Analysis and Applications*, 13(3):962–978, 1992.

[33] C. Keller. *Constraint preconditioning for indefinite linear systems.* D. Phil. thesis, Oxford University, England, 2000.

[34] C. Keller, N. I. M. Gould, and A. J. Wathen. Constraint preconditioning for indefinite linear systems. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1300–1317, 2000.

[35] M. Lalee, J. Nocedal, and T. D. Plantenga. On the implementation of an algorithm for large-scale equality constrained optimization. *SIAM Journal on Optimization*, 8(3):682–706, 1998.

[36] L. Lukšan and J. Vlček. Indefinitely preconditioned inexact Newton method for large sparse equality constrained nonlinear programming problems. *Numerical Linear Algebra with Applications*, 5(3):219–247, 1998.

[37] S. G. Nash and A. Sofer. Preconditioning reduced matrices. *SIAM Journal on Matrix Analysis and Applications*, 17(1):47–68, 1996.

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

[39] T. D. Plantenga. A trust-region method for nonlinear programming based on primal interior point techniques. *SIAM Journal on Scientific Computing*, 20(1):282–305, 1999.

[40] R. J. Plemmons and R. E. White. Substructuring methods for computing the null space of equilibrium matrices. *SIAM Journal on Matrix Analysis and Applications*, 11(1):1–22, 1990.

[41] B. T. Polyak. The conjugate gradient method in extremal problems. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 9:94–112, 1969.

[42] T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637, 1983.

[43] J. M. Stern and S. A. Vavasis. Nested dissection for sparse nullspace bases. *SIAM Journal on Matrix Analysis and Applications*, 14:766–775, 1993.

[44] Ph. L. Toint. Towards an efficient sparsity exploiting Newton method for minimization. In I. S. Duff, editor, *Sparse Matrices and Their Uses*, pages 57–88, London, 1981. Academic Press.

[45] Ph. L. Toint and D. Tuyttens. On large-scale nonlinear network optimization. *Mathematical Programming, Series B*, 48(1):125–159, 1990.

[46] L. N. Vicente. *Trust-region interior-point algorithms for a class of nonlinear programming problems.* PhD thesis, Department of Computational and Applied Mathematics, Rice University, Houston, Texas, USA, 1995. Report TR96-05.