RAL-TR-1998-069

On the solution of equality constrained quadratic programming problems arising in optimization

Nicholas I. M. Gould[1,2], Mary E. Hribar[3] and Jorge Nocedal[4,5]

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 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. This can yield substantial economies in both line search and trust region methods for large-scale optimization, but can also 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 in high computational overhead. Numerical results illustrating the efficiency of the proposed approaches are presented.

---

[1] Department for Computation and Information, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, England, EU. Email : n.gould@rl.ac.uk

[2] Current reports available from from "http://www.numerical.rl.ac.uk/reports/reports.html".

[3] CAAM Department, Rice University, Houston TX 77005, USA. Email : marybeth@tera.com.
This author was supported by Department of Energy grant DE-FG02-87ER25047-A004.

[4] Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL 60208-3118, USA. Email : nocedal@eecs.nwu.edu.
This author was supported by Science Foundation grant CCR-9625613 and by Department of Energy grant DE-FG02-87ER25047-A004.

[5] Current reports available from "http://www.eecs.nwu.edu/~nocedal/PSfiles".

Department for Computation and Information
Atlas Centre
Rutherford Appleton Laboratory
Oxfordshire OX11 0QX

September 30, 1998.

1 Introduction

A variety of algorithms for nonlinearly constrained optimization [7, 8, 12, 29, 31] use the conjugate gradient (CG) method [25] to solve subproblems of the form

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

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

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 (1.2) 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 (1.2) 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 [36, 38], and, by contrast, [23]).

The use of an iterative method such as CG is attractive in large scale optimization because, when the number of variables is large, it can be cost effective to solve (1.1)–(1.2) approximately, and only increase the accuracy of the solution as the iterates of the optimization algorithm approach the minimizer. In addition, the properties of the CG method merge very well with the requirements of globally convergent optimization methods (see e.g. [36]). In this paper we study how to apply the preconditioned CG method to (1.1)–(1.2) so as to keep the computational cost at a reasonable level while ensuring that rounding errors do not degrade the performance of the optimization algorithm.

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. We will argue, however, that due to the form of the preconditioners used in practice, the explicit use of Z will cause the iteration to be very expensive, and that significant savings can be achieved by means of approaches that bypass the computation of Z altogether. The price to pay for these alternatives is that they can give rise to excessive roundoff errors that can slow the optimization iteration and may even prevent it from converging.

As we shall see, these errors cause the constraints (1.2) not to be satisfied to the desired accuracy. We describe iterative refinement techniques that can improve the accuracy of the solution in highly ill-conditioned problems. 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 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. [17, 20]). 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 (1.2) 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 (1.2) 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, (2.3) is equivalent to 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 compatible with the requirements of many nonlinear optimization algorithms which need to 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. [1]), 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 $W_{ZZ}^{-1} H_{ZZ}$, or could be the result of an automatic scaling of the variables [7, 29]. Regardless of how W_{ZZ} is defined, the preconditioned conjugate gradient method applied to (2.4) is as follows (see, e.g. [20]).

Algorithm I. Preconditioned CG for Reduced Systems.

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

$$\alpha = {r_Z}^T g_Z / {p_Z}^T H_{ZZ} p_Z \tag{2.5}$$

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

$${r_Z}^+ = r_Z + \alpha H_{ZZ} p_Z \tag{2.7}$$

$${g_Z}^+ = W_{ZZ}^{-1} {r_Z}^+ \tag{2.8}$$

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

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

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

This iteration may be terminated, for example, when ${r_Z}^T W_{ZZ}^{-1} r_Z$ is sufficiently small. Once an approximate solution is obtained, 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 $p = Zp_Z$.

Algorithm II Preconditioned CG (in Expanded Form) for Reduced Systems.

Choose an initial point x satisfying (1.2), compute $r = Hx + c$, $g = ZW_{ZZ}^{-1}Z^T r$ and $p = -g$. Repeat the following steps, until a convergence test is satisfied:

$$\alpha = r^T g / p^T H p \tag{2.12}$$

$$x \leftarrow x + \alpha p \tag{2.13}$$

$$r^+ = r + \alpha H p \tag{2.14}$$

$$g^+ = Z W_{ZZ}^{-1} Z^T r^+ \tag{2.15}$$

$$\beta = (r^+)^T g^+ / r^T g \tag{2.16}$$

$$p \leftarrow -g^+ + \beta p. \tag{2.17}$$

$$g \leftarrow g^+ \quad \text{and} \quad r \leftarrow r^+ \tag{2.18}$$

This will be the main algorithm studied in this 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 will use the quantity $r^T g = {r_Z}^T W_{ZZ}^{-1} r_Z$ to terminate the CG iteration.

Note that the vector g, which we call the *preconditioned residual*, has been explicitly defined to be in the range of Z. As a result, in exact arithmetic, all the search directions p generated by Algorithm II will also lie in the range of Z, and thus the iterates x will all satisfy (1.2). Rounding errors when computing (2.17) may cause p to have a component outside the range of Z, but this component will normally be too small to cause difficulties.

3 Implementation of the Projected CG Method

Algorithm II constitutes an effective method for computing the solution to (1.1)–(1.2) and has been successfully used in various algorithms for large scale optimization (cf. [16, 28, 39]). The main drawback is the need for a null-space basis matrix Z, whose computation and manipulation can be costly, and which can sometimes give rise to unnecessary ill-conditioning [9, 10, 18, 24, 33, 37]. These difficulties will become apparent when we describe practical procedures for computing Z and when we consider the types of preconditioners W_{ZZ} used in practice. Let us begin with the first issue.

3.1 Computing a basis for the null space

There are many possible choices for the null-space matrix Z. Possibly the best strategy is to choose Z so as to have orthonormal columns, for this provides a well conditioned representation of the null space of A. However computing such a null-space matrix can be very expensive when the number of variables is large; it essentially requires the computation of a sparse LQ factorization of A and the implicit or explicit generation of Q, which has always been believed to be rather expensive when compared with the alternatives described in [24]. Recent research [30, 35] has suggested that it is in fact possible to generate Q as a product of sparse Householder matrices, and that the cost of this may, after all, be reasonable. We have not experimented with this approach, however, because, to our knowledge, general purpose software implementing it is not yet available.

Another possibility is to try to compute a basis of the null-space which involves as few nonzeros as possible. Although this problem is computationally hard [9], sub-optimal heuristics are possible but still rather expensive [10, 18, 33, 37].

A more economical alternative is based on simple elimination of variables [17, 20]. To define Z we first group the components of x into m basic or dependent variables (which for simplicity are assumed to be the first m variables) and $n-m$ nonbasic or control variables, and partition A as

$$A = (B \quad N),$$

where the $m \times m$ basis matrix B is assumed to be nonsingular. Then we define

$$Z = \begin{pmatrix} -B^{-1}N \\ I \end{pmatrix}, \tag{3.1}$$

which clearly satisfies $AZ = 0$ and has linearly independent columns. In practice Z is not formed explicitly; instead we compute and store sparse LU factors [13] of B, and compute products of the form Zv and $Z^T v$ by means of solves using these LU factors. Ideally we would like to choose a basis B that is as sparse as possible and whose condition number is not significantly worse than that of A, but these requirements can be difficult to achieve. In fact, simply ensuring that B is well conditioned can be difficult when the task of choosing a basis is delegated to a sparse LU factorization algorithm such as MA48 [15]. Some recent codes (see, e.g., [19]) have been designed

to compute a well-conditioned basis, but it is not known to us to what extent they reach their objective.

3.2 Preconditioning

These potential drawbacks of the null-space basis (3.1) are not sufficiently serious to prevent its effective use in Algorithm II. However, when considering practical choices for the preconditioning matrix W_{ZZ}, one exposes the weaknesses of this approach. Ideally, one would like to choose W_{ZZ} so that $W_{ZZ}^{-1} H_{ZZ} = I$, and thus

$$W_{ZZ}^{-1} = (Z^T H Z)^{-1} \tag{3.2}$$

is the perfect preconditioner. However, it is unlikely that $Z^T H Z$ or its inverse are sparse matrices, and even if $Z^T H Z$ is of small dimension, forming it can be quite costly. Therefore operating with this ideal preconditioner is normally out of the question.

In this paper we consider preconditioners of the form

$$W_{ZZ}^{-1} = (Z^T G Z)^{-1}, \tag{3.3}$$

where G is a symmetric matrix such that $Z^T G Z$ is positive definite. Some suggestions on how to choose G have been made in [32]. Two particularly simple choices are

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

The first choice is appropriate when H is dominated by its 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 [7, 12, 29], where the preconditioner (which derives from a change of variables) is thus given by

$$W_{ZZ}^{-1} = (Z^T Z)^{-1}. \tag{3.4}$$

Regardless of the choice of G, the preconditioner (3.3) requires operations with the inverse of the matrix $Z^T G Z$. In some applications [16, 39] Z, defined by (3.1), has a simple enough structure that forming and factorizing the $(n-m) \times (n-m)$ matrix $Z^T G Z$ is not expensive when G has a simple form. But if the LU factors of B are not very sparse and the number of constraints m is large, forming $Z^T G Z$ may be rather costly, even if $G = I$, as it requires the solution of $2m$ triangular systems with these LU factors. In this case it is preferable not to form $Z^T G Z$, but rather compute products of the form $(Z^T G Z)^{-1} u = v$ by solving $(Z^T G Z) v = u$ using the CG method. This inner CG iteration has been employed in [29] with $G = I$, and can be effective on some problems—particularly if the number of degrees of freedom, $n - m$, is very small. But it can fail when Z is badly conditioned and tends to be expensive. Moreover, since the matrix $Z^T G Z$ is not known explicitly, it is difficult to construct effective preconditioners for accelerating this inner CG iteration.

In summary when the preconditioner has the form (3.3), and when Z is defined by means of (3.1), the computation (2.15) of the preconditioned residual g is often so expensive as to

dominate the cost of the optimization algorithm. The goal of this paper is to consider alternative implementations of Algorithm II whose computational cost is more moderate and predictable. Our approach is to avoid the use of the null-space basis Z altogether.

3.3 Computing Projections

To see how to bypass the computation of Z, let us begin by considering the simple case when $G = I$, so that the preconditioner W_{ZZ} is given by (3.4). If P_Z denotes the orthogonal projection operator onto the null space of A,

$$P_Z = Z(Z^T Z)^{-1} Z^T, \tag{3.5}$$

then the preconditioned residual (2.15) can be written as

$$g^+ = P_Z r^+. \tag{3.6}$$

This projection can be performed in two alternative ways.

The first is to replace P_Z by the equivalent formula

$$P_A = \left(I - A^T (AA^T)^{-1} A\right) \tag{3.7}$$

and thus to replace (3.6) with

$$g^+ = P_A r^+. \tag{3.8}$$

We can express this as

$$g^+ = r^+ - A^T v^+, \tag{3.9}$$

where v^+ is the solution of

$$AA^T v^+ = Ar^+. \tag{3.10}$$

Noting that (3.10) are the normal equations, it follows that v^+ is the solution of the least squares problem

$$\underset{v}{\text{minimize}} \quad \|r^+ - A^T v^+\|, \tag{3.11}$$

and that the desired projection g^+ is the corresponding residual. This approach can be implemented using a Cholesky factorization of AA^T.

The second possibility is to express the projection (3.6) as the solution of the augmented system

$$\begin{pmatrix} I & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} g^+ \\ v^+ \end{pmatrix} = \begin{pmatrix} r^+ \\ 0 \end{pmatrix}. \tag{3.12}$$

This system can be solved by means of a symmetric indefinite factorization that uses 1×1 and 2×2 pivots [21].

Let us suppose now that the preconditioner has the more general form (3.3). The preconditioned residual (2.15) now requires the computation

$$g^+ = P_{Z:G} r^+ \qquad \text{where} \qquad P_{Z:G} = Z(Z^T G Z)^{-1} Z^T. \tag{3.13}$$

This may be expressed as

$$g^+ = P_{A:G} r^+ \qquad \text{where} \qquad P_{A:G} = G^{-1}\left(I - A^T(AG^{-1}A^T)^{-1}AG^{-1}\right) \tag{3.14}$$

if G is non-singular, and can be found as the solution of

$$\begin{pmatrix} G & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} g^+ \\ v^+ \end{pmatrix} = \begin{pmatrix} r^+ \\ 0 \end{pmatrix} \tag{3.15}$$

whenever $Z^T G Z$ is non-singular (see, e.g., [20, Section 5.4.1]). While (3.14) is far from appealing when G^{-1} does not have a simple form, (3.15) is a useful generalization of (3.12). Clearly the system (3.12) may be obtained from (3.15) 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 $Z^T G Z$ be positive definite. The idea of using the projection (3.7) in the CG method dates back to at least [34]; the alternative (3.15), and its special case (3.12), are proposed in [8], although [8] unnecessarily requires that G be positive definite. A more recent study on preconditioning the projected CG method is [11].

Hereafter we shall write (2.15) as

$$g^+ = Pr^+,$$

where P is any of the projection operators we have mentioned above.

Note that (3.8), (3.12) and (3.15) do not make use of the null space matrix Z and only require factorization of matrices involving A. Unfortunately they can give rise to significant round-off errors, 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

$$Hx^* + c = A^T\lambda, \tag{3.16}$$

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 (3.16), 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 (3.13), the preconditioned residual g will converge to zero so long as the smallest eigenvalue of $Z^T G Z$ is bounded away from zero.

That this discrepancy in the magnitudes of $g^+ = Pr^+$ and r^+ will cause numerical difficulties is apparent from (3.9), which shows that significant cancellation of digits will usually take place. The generation of harmful roundoff errors is also apparent from (3.12)/(3.15) because g^+ will be small while the remaining components v^+ remain large. Since the magnitude of the errors generated in the solution of (3.12)/(3.15) is governed by the size of the large component v^+, the vector g^+ will contain large relative errors. These arguments will be made more precise in the next section.

Example 1.

We applied Algorithm II to solve problem CVXEQP3 from the CUTE collection [4], with $n = 1000$ and $m = 750$. In this and all subsequent experiments, we use the simple preconditioner

(3.4) corresponding to the choice $G = I$. We used both the normal equations (3.8) and augmented system (3.12) approaches to compute the projection. The results are given in Figure 3.1, 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 $r^T g = {r_Z}^T W_{ZZ} r_Z$ 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 the cosine of the angle between the preconditioned residual g and the rows of A. More precisely, we define

$$\cos\theta = \max_i \left\{ \frac{A_i^T g}{||A_i||\ ||g||} \right\} \tag{3.17}$$

where A_i is the i-th row of A. Note that this cosine, which should be zero in exact arithmetic, increases indicating that the CG iterates leave the constraint manifold $Ax = b$.

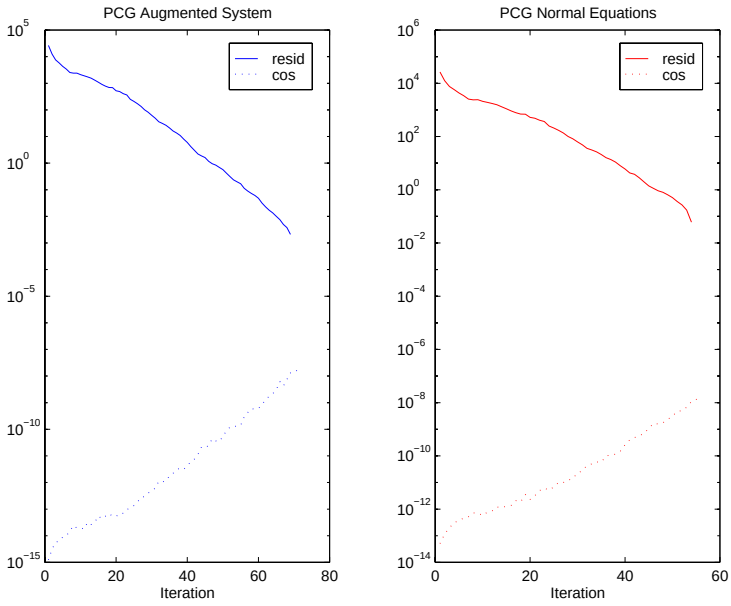

Figure 3.1: Conjugate gradient method with two options for the projection

Severe errors such as these are not uncommon in optimization calculations; see §7 and [27]. This is of grave concern as it may cause the underlying optimization algorithms to behave erratically or fail.

In this paper 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 preconditioner, $G = I$. Any computed, as opposed to exact, quantity will be denoted by a subscript c.

Let us first consider the *normal equations approach*. Here $g^+ = P_A r^+$ is given by (3.9) where (3.10) 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 [3, p.49][(1)]

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

with $\gamma = 2.5n^{3/2}$. Recall that ϵ_m denotes unit roundoff and $\kappa(A)$ the condition number of A.

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 $g^+ - A^T v^+$ given in (3.9), because these errors will be dominated by the error in v^+ whose magnitude is estimated by (4.1). Under these assumptions, we have from (3.9) that the computed projection $g_c^+ = (P_A r^+)_c$ and the exact projection $g^+ = P_A r^+$ 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

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

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

$$\frac{\|v^+\|}{\|g^+\|} = \frac{\|v^+\|}{\|P_A r^+\|} \tag{4.4}$$

is large.

Let us consider the ratio (4.4) in the case when $\|r^+\|$ is much larger than its projection $\|g^+\|$. We have from (3.9) 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

$$\frac{\|v^+\|}{\|g^+\|} \approx \frac{\|r^+\|}{\|P_A r^+\|},$$

[(1)] 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. 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.

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 $\|r^+\|/\|P_A r^+\|$ 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 optimization calculation. 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 1,

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

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. This is in agreement with our test, for at this point the CG iteration could make no further progress.

Let us now consider the *augmented system approach* (3.15). 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 [5] and Duff and Reid [14] 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 [2] suggests introducing a parameter α and solving the equivalent system

$$\begin{pmatrix} \alpha I & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \alpha^{-1} g^+ \\ v^+ \end{pmatrix} = \begin{pmatrix} r^+ \\ 0 \end{pmatrix}. \tag{4.6}$$

An error analysis [3] shows that

$$\|g^+ - g_c^+\| \leq \quad \eta \epsilon_m (\sigma_1 + \alpha\kappa(A))(\alpha^{-1}\|g^+\| + \|v^+\|) \tag{4.7}$$

where η depends on n and m and in the growth factor during the factorization, and $\sigma_1 \geq \cdots \geq \sigma_m > 0$ are the nonzero singular values of A. It is important to notice that now $\kappa(A)$—and not $\kappa^2(A)$—enters in the bound. If $\alpha \approx \sigma_m(A)$, this method will give a solution that is never much worse than that obtained by a tight perturbation analysis, and therefore can be considered stable for practical purposes. But approximating $\sigma_m(A)$ can be difficult, and it is common to simply use $\alpha = 1$.

In the case which concerns us most, when $\|g^+\|$ converges to zero while $\|v^+\|$ is bounded, the term inside the last square brackets in (4.7) is approximately $\|v^+\|$, and we obtain

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

where we have assumed that $\alpha = 1$. 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 $P_A r^+$ via the normal equations approach (3.9)-(3.10). An appealing idea for trying to improve the accuracy of this computation is to apply the projection repeatedly. Therefore rather than computing g^+ by $g^+ = P_A r^+$ in (2.15), 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 $g_c^+ = (P_A r^+)_c$ 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, for $i = 0, 1, \cdots$, performing the following steps:

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

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

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.

Example 2.

We solved the problem given in Example 1 using multiple projections. At every CG iteration we measure the cosine (3.17) 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 5.1, and show that the residual $\sqrt{r^T g}$ was reduced much more than

in the plane CG iteration (Figure 3.1). Indeed the ratio between the final and initial values of $\sqrt{r^T g}$ is 10^{-16}, which is very satisfactory.

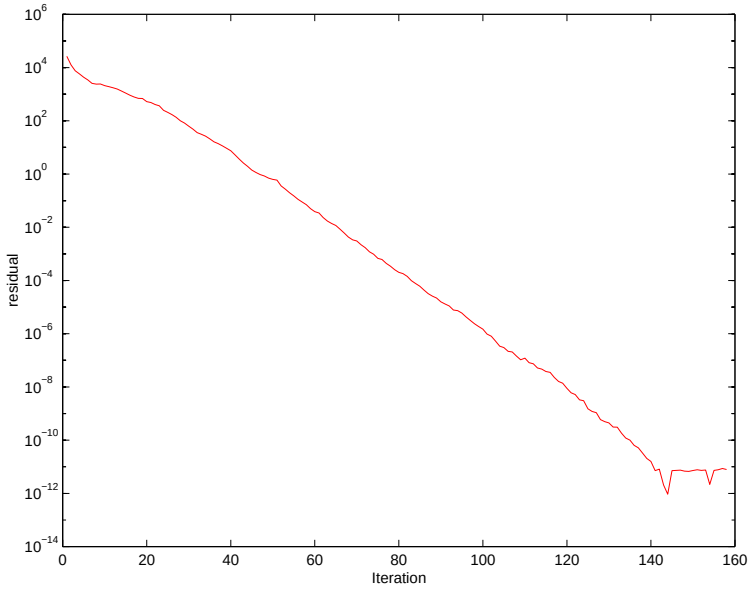

Figure 5.1: CG method using multiple projections in the normal equations approach.

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 $i = 0, 1, \cdots$,

$$(g^+_{i+1})_c = g^+ - A^T \Delta v^+_i, \tag{5.3}$$

where as in (4.1)

$$\|\Delta v^+_i\| \leq \gamma \epsilon_m \kappa^2(A) \|v^+_i\|, \qquad \text{and} \qquad v^+_i = -\Delta v^+_{i-1}. \tag{5.4}$$

A simple consequence of (5.3)–(5.4) and the assumption that A has norm one is that

$$\|(g^+_{i+1})_c - g^+\| \leq \|\Delta v^+_i\| \leq \left(\gamma \epsilon_m \kappa^2(A)\right)^i \|v^+\|, \tag{5.5}$$

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

$$\gamma \epsilon_m \kappa^2(A). \tag{5.6}$$

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 (5.5) 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 (5.5); 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* [3, p.125]. Fixed

precision iterative refinement is appropriate because the approach we have chosen to compute projections is not stable. To see this, compare (4.3) with a perturbation analysis of the least squares problem [3, Theorem 1.4.6]), which gives

$$\|g^+ - g_c^+\| = O\left(\epsilon_m(\|v\| + \kappa(A)\|g^+\|)\right). \tag{5.7}$$

Here the dependence on the condition number is linear—not quadratic. Moreover, since $\kappa(A)$ is multiplied by $\|g^+\|$, when g^+ is small the effect of the condition number of A is much smaller in (5.7) than in (4.3).

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* [3, Section 2.9] to attempt to improve the solution v^+ of the *normal equations* (3.10). This, however, will generally be unsuccessful because fixed-precision iterative refinement only improves a measure of backward stability [21, 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* (3.10). So long as $\epsilon_m\kappa(A)^2 < 1$, and the residuals of (3.10) are smaller than one in norm, we can expect that the error in the solution of (3.10) 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 (3.15). This gives the following iteration.

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

Note that this method is applicable for general preconditioners G. When $G = I$, and if an appropriate value of α is in hand, we should incorporate it in this iteration, as described in (4.6). The general analysis of Higham [26, Theorem 3.2] indicates that, if the condition number of A is not too large, we can expect high accuracy in v^+ and good accuracy in g^+ in most cases.

Example 3.

We solved the problem given in Example 1 using this iterative refinement technique. As in the case of multiple projections discussed in Example 2, 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}. The results are given in Figure 5.2.

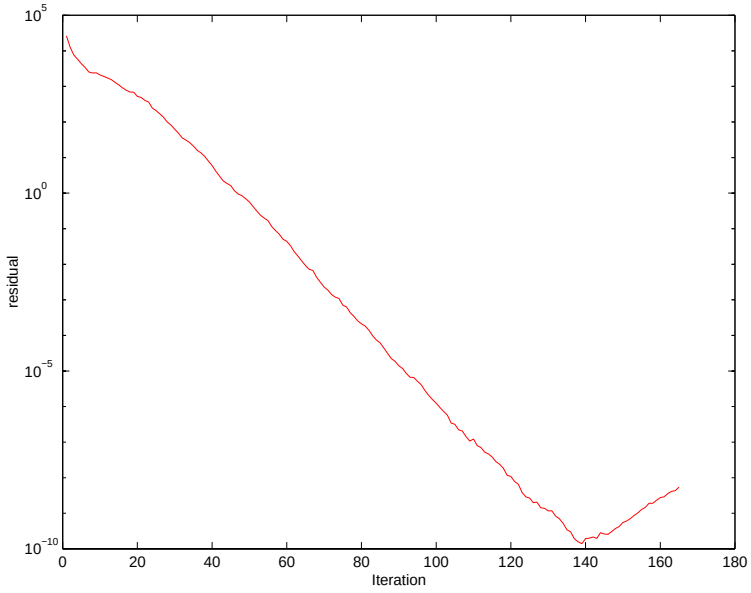

Figure 5.2: CG method using iterative refinement in the augmented system approach.

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.

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^+. 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 (2.14), 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 condition $AZ = 0$ and the fact that r^+ is only used in (2.15) and (2.16). It follows that we can redefine r^+ by means of (6.1) in either the

normal equations approach (3.8)/(3.13) or in the augmented system approach (3.12)/(3.15) 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 G is any symmetric matrix for which $Z^T G Z$ is positive definite, and G^{-1} is the generalized inverse of G. The vector y that solves (6.2) is obtained as $y = v^+$ from (3.15). This gives rise to the following modification of the CG iteration.

Algorithm III Preconditioned CG with Residual Update.

Choose an initial point x satisfying (1.2), 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 $g = Z W_{ZZ}^{-1} Z^T r$ and set $p = -g$. Repeat the following steps, until a convergence test is satisfied:

$$\begin{aligned}
\alpha &= r^T g / p^T H p && (6.3)\\
x &\leftarrow x + \alpha p && (6.4)\\
r^+ &= r + \alpha H p && (6.5)\\
r^+ &\leftarrow r^+ - A^T y \qquad \text{where } y \text{ solves (6.2)} && (6.6)\\
g^+ &= P r^+ && (6.7)\\
\beta &= (r^+)^T g^+ / r^T g && (6.8)\\
p &\leftarrow -g^+ + \beta p && (6.9)\\
g &\leftarrow g^+ \quad \text{and} \quad r \leftarrow r^+. && (6.10)
\end{aligned}$$

This procedure works well in practice, and can be improved by adding iterative refinement of the projection operation. 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 (3.15). But (3.15) 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 (6.11). The advantage of using (6.11) compared to (3.15) is that the solution in the latter may be dominated by the large components v^+, while in the former g^+ are the large components—of course, in floating point arithmetic, the zero component in the solution of (6.11) will instead be tiny rounded values provided (6.11) 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 [6, 22].

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 reduces its magnitude sufficiently to avoid errors in the first few CG iteration, subsequent values of r can grow, and rounding errors may reappear. The strategy proposed in Algorithm III is safe in that it ensures that r is small at every iteration, but one can think of various alternatives. One of them is to monitor the norm of r and only apply the residual update when it seems to be growing.

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 (3.11) 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 (3.10) or (3.12). Furthermore, the first block of equations in (3.12) shows that $r^+ - A^T v^+ = g^+$. Therefore, in this case (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 $r^+ \leftarrow Pr^+$. Rather than performing this projection, we will define $r^+ = g$ where g is the projected residual computed at the previous iteration. The resulting iteration is given by Algorithm III with the following two changes:

Omit (6.6)

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

This strategy has performed well in our numerical experiments and avoids the extra storage and computation required by Algorithm III. We now show that it is mathematically equivalent to Algorithm III — which in turn is mathematically equivalent to Algorithm II. The arguments that follow make use of the fact that, when $G = I$, we have that $PP = P$.

The first iteration is clearly the same as that of Algorithm III, except that the value we store in r in the last step is not r^+ but $g^+ = Pr^+$. Let us consider the effect that this has on the next iteration. The numerator in the definition (6.3) of α now becomes $g^T g$ which equals $r^T Pg = r^T g$. Thus the formula of α is theoretically unchanged, but the symmetric form $\alpha = g^T g / p^T Hp$ has the advantage that it can never be negative, as is the case with (6.3) when rounding errors dominate

the projection operation. Next, the step (6.5) becomes $r^+ = Pr + \alpha Hp$, which is different from the value calculated in Algorithm III. Step (6.6) is omitted in the new variant of Algorithm III. The projected residual calculated in (6.7) is now $P(Pr+\alpha Hp)$ which is mathematically equivalent to the value $PP(r+\alpha Hp)$ calculated in Algorithm III (recall that (6.6) can be written as Pr^+). Note that the new strategy applies the double projection only to r. Finally let us consider the numerator in (6.8). In the new variant, it is given by

$$(Pr + \alpha Hp)^T P(Pr + \alpha Hp).$$

whereas in Algorithm III it is given by

$$(P(r + \alpha Hp))^T PP(r + \alpha Hp).$$

By expanding these expressions we see that the formula for β is mathematically equivalent in both cases, but that in the new variant the projection is applied selectively.

Example 4.

We solved the problem given in Example 1 using this residual update strategy with $G = I$. The results are given in Figure 6.1 and show that the normal equations and augmented system approaches are equally effective in this case. We do not plot the cosine (3.17) 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.

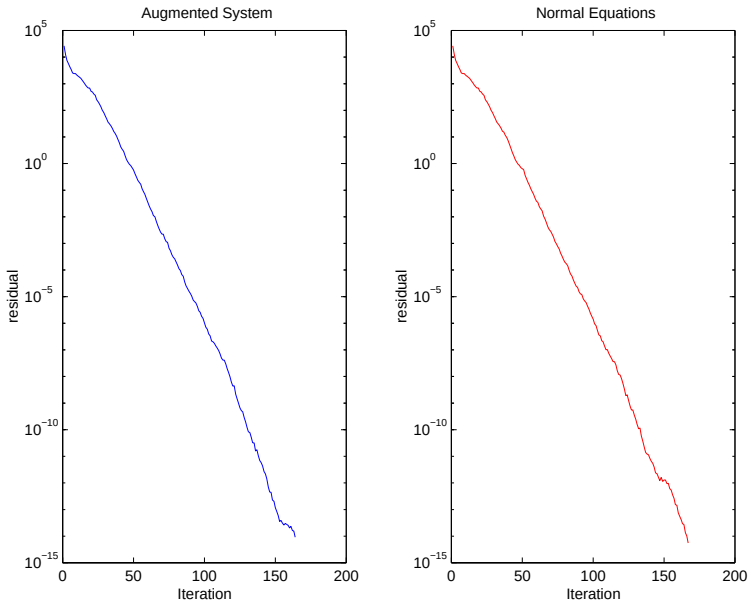

Figure 6.1: Conjugate gradient method with the residual update strategy.

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 (1.2), compute $r = Hx + c$, $r \leftarrow Pr$, $g \leftarrow Pr$, where the projection is computed by the normal equations (3.8) or augmented system (3.12) approaches, and set $p = -g$. Choose a tolerance θ_{max}. Repeat the following steps, until a convergence test is satisfied:

$$\alpha = r^T g / p^T H p \tag{6.12}$$
$$x \leftarrow x + \alpha p \tag{6.13}$$
$$r^+ = r + \alpha H p \tag{6.14}$$
$$g^+ = P r^+ \tag{6.15}$$
$$\text{Apply iterative refinement to } Pr^+, \text{ if necessary,} \tag{6.16}$$
$$\text{until (3.17) is less than } \theta_{max} \tag{6.17}$$
$$\beta = (r^+)^T g^+ / r^T g \tag{6.18}$$
$$p \leftarrow -g^+ + \beta p \tag{6.19}$$
$$g \leftarrow g^+ \quad \text{and} \quad r \leftarrow g^+. \tag{6.20}$$

We conclude this discussion by elaborating on the point made before Example 4 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 (6.12) since it 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 I using the augmented system approach; see Figure 3.1. 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 3. 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 given by 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 from (6.7).

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 here, it has proved to be beneficial in other, similar circumstances [22].

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 [7], when this method was applied to a set of test problems from the CUTE [4] collection. We apply the CG method with preconditioner (3.4) (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) 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 7.1. 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 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 [7]). We us 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 $r^T g < 0$, indicated by ^{4}. Note that the standard CG iteration was not able to meet the stopping test for any of the problems in Table 7.1, but that iterative refinement and update residual were successful in most cases.

		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 7.1: 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 $r^T g < 0$.

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 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 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 the interior point nonlinear optimization code described in [7] 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 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.

Problem	dim	Augmented System			Normal Equations		
		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

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 $r^T g < 0$.

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

			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 7.4: Number of function evaluations and projections required by the optimization method for the different implementations of the CG iteration.

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 (3.3).

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.

Acknowledgements

The authors would like to thank Andy Conn and Philippe Toint for their helpful input during the early stages of this research.

References

[1] O. Axelsson. *Iterative solution methods.* Cambridge University Press, Cambridge, 1996.

[2] 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.

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

[4] 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.

[5] 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.

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

[7] R. H. Byrd, M. E. Hribar, and J. Nocedal. Primal and primal-dual methods for nonlinear programming. Technical Report in preparation, Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, Illinois, USA, 1997.

[8] 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, 1994. SIAM.

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

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

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

[12] J. E. Dennis, M. El-Alem, and M. C. Maciel. A global convergence theory for general trust-region based algorithms for equality constrained optimization. Technical Report CRPC-TR92247, Center for Research on Parallel Computers, Rice University, Houston, Texas, USA, 1992. To appear *SIAM Journal on Optimization.*

[13] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct methods for sparse matrices.* Oxford University Press, Oxford, 1986.

[14] 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.

[15] I. S. Duff and J. K. Reid. The design of MA48, a code for direct solution of sparse unsymmetric linear systems of equations. *ACM Transactions on Mathematical Software*, 22(2):187–226, 1996.

[16] 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.

[17] R. Fletcher. *Practical Methods of Optimization.* J. Wiley and Sons, Chicester and New York, second edition, 1987.

[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, and M. A. Saunders. SNOPT: an SQP algorithm for large-scale constrained optimization. Technical Report SOL96-0, Department of Operations Research, Stanford University, Stanford, California 94305, USA, 1996.

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

[21] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, second edition, 1989.

[22] N. I. M. Gould. Iterative methods for ill-conditioned linear systems from optimization. Technical Report RAL-TR-1998-064, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1998.

[23] N. I. M. Gould, S. Lucidi, M. Roma, and Ph. L. Toint. Solving the trust-region subproblem using the Lanczos method. Technical Report RAL-TR-97-028, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1997.

[24] 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.

[25] 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.

[26] N. J. Higham. Iterative refinement and LAPACK. Numerical Analysis Report No. 277, Manchester Centre for Computational Mathematics, Manchester, England, 1995.

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

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

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

[30] S.-M. Lu and J. L. Barlow. Multifrontal computation with the orthogonal factors of sparse matrices. *SIAM Journal on Matrix Analysis and Applications*, 17(3):658–679, 1996.

[31] 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.

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

[33] 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.

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

[35] C. Puglisi. *QR Factorization of Large Sparse Overdetermined and Square Matrices with the Multifrontal Method in a Multiprocessing Environment*. PhD thesis, Institut National Polytechnique de Toulouse, Toulouse, France, 1993.

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

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

[38] Ph. L. Toint. Towards an efficient sparsity exploiting Newton method for minimization. In I. S. Duff, editor, *Sparse Matrices and Their Uses*, London and New York, 1981. Academic Press.

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