Computing a Trust Region Step for a Penalty Function†

Thomas F. Coleman
Christian Hempel

87-847

July 1987

Department of Computer Science
Cornell University
Ithaca, New York 14853-7501

†Research supported by the Applied Mathematical Sciences Research Program (KC-04-02) of the Office of Energy Research of the U.S. Department of Energy under grant DE-FG02-86ER25013.A000.

Computing a Trust Region Step for a Penalty Function

Thomas F. Coleman*
Dept. of Computer Science
& Center for Applied Mathematics
Cornell University
Ithaca, New York 14853

Christian Hempel*
Center for Applied Mathematics
Cornell University
Ithaca, New York 14853

Abstract:

We consider the problem of minimizing a quadratic function subject to an ellipsoidal constraint when the matrix involved is the Hessian of a quadratic penalty function (i.e., a function of the form $p(x) = f(x) + \frac{1}{2\mu}c(x)^T c(x)$). Most applications of penalty functions require $p(x)$ to be minimized for values of μ decreasing to zero. In general, as μ tends to zero the nature of finite precision arithmetic causes a considerable loss of information about the null space of the constraint gradients when $\nabla^2 p(x)$ is formed. This loss of information renders ordinary trust region Newton's methods unstable and degrades the accuracy of the solution to the trust region problem. The algorithm of Moré and Sorensen [1983] is modified so as to be more stable and less sensitive to the nature of finite precision arithmetic in this situation. Numerical experiments clearly demonstrate the stability of the proposed algorithm.

*

* Research supported by the Applied Mathematical Sciences Research Program (KC-04-02) of the Office of Energy Research of the U.S. Department of Energy under grant DE-FG02-86ER25013.A000.

I. Introduction.

The generic "trust region" problem is

$$
\begin{aligned}
&\text{minimize } \psi(s) = g^T s + \frac{1}{2} s^T H s \\
&\text{subject to } \|s\|_2 \leq \Delta
\end{aligned}
\tag{1.1}
$$

where Δ is a positive number, $g \in R^n$, and $H \in R^{n \times n}$ is a symmetric matrix. We will concern ourselves with the case where H is the Hessian of an L_2 penalty function. That is, H is the Hessian of a function of the form

$$p(x) = f(x) + \frac{1}{2\mu} c(x)^T c(x) \tag{1.2}$$

where $f\colon R^n \to R$, $c\colon R^n \to R^t$ for $t \leq n$, and $\mu > 0$. For such functions the quadratic $\psi(s)$ is meant to serve as a local model at a current point, x_c. Evaluating the first and second derivatives of $p(x)$ at x_c gives

$$g = \nabla p(x_c) = \nabla f(x_c) + \frac{1}{\mu} \nabla c(x_c) c(x_c) \tag{1.3}$$

and

$$H = \nabla^2 p(x_c) = \nabla^2 f(x_c) + \frac{1}{\mu} \sum_{i=1}^{t} \nabla^2 c_i(x_c) c_i(x_c) + \frac{1}{\mu} \nabla c(x_c) \nabla c(x_c)^T \tag{1.4}$$

If we define

$$A = \nabla c(x_c) \tag{1.5}$$

and

$$B = \nabla^2 f(x_c) + \frac{1}{\mu} \sum_{i=1}^{t} \nabla^2 c_i(x_c) c_i(x_c) \tag{1.6}$$

then

$$H = B + \frac{1}{\mu} A A^T. \tag{1.7}$$

Matrices of the form (1.7) become ill-conditioned as μ tends to zero. Also, when μ is small many of the digits of B will, in general, be lost when H is formed. This loss of information and ill-conditioning can have disastrous effects on methods which rely on having H. There is, as we shall see, a way to solve (1.1) which sidesteps these difficulties.

Our method is an adaptation of the method of Moré and Sorensen [1983] and retains their method's convergence properties. That is, our algorithm will compute a nearly optimal solution in finitely many iterations. More precisely, for σ_1 and σ_2 in (0,1) the approximate solution, s, satisfies

$$\psi(s) - \psi(s^*) \leq \sigma_1(2-\sigma_1)\max\{|\psi(s^*)|, \sigma_2\} \quad \text{and} \quad \|s\| \leq (1+\sigma_1)\Delta \tag{1.8}$$

where s^* is a solution to (1.1).

The paper is organized as follows. The next section briefly reviews the theoretical background needed to develop a suitable algorithm. Readers wishing a more thorough treatment should see Moré and Sorensen [1983]. The algorithm is constructed in Section 3. Lastly, in Section 4 we give a detailed report of numerical tests comparing TRSQPF with GQTPAR on trust region problems.

II. Structure of the problem.

In this section we will show that the trust region problem usually reduces to a zero finding problem of a rational function with second order poles. This perspective will enable us to construct an effective algorithm with good convergence properties. The next two lemmas provide necessary and sufficient conditions for a point $s \in R^n$ to be a solution to (1.1).

Lemma(2.1): If s is a solution to (1.1) then s is a solution to an equation of the form

$$(H + \alpha I)s = -g \tag{2.2}$$

with $H + \alpha I$ positive definite, $\alpha \geq 0$, and $\alpha(\Delta - \|s\|) = 0$.

Lemma(2.3): Let $s \in R^n$ satisfy (2.2) with $H + \alpha I$ positive semi-definite.

(i) If $\alpha = 0$ and $\|s\| < \Delta$ then s solves (1.1)

(ii) s solves $\psi(s) = \min\{\psi(w): \|w\| = \|s\|\}$

(iii) If $\alpha \geq 0$ and $\|s\| = \Delta$ then s solves (1.1)

For proofs of these lemmas see Sorensen [1982]. Moré and Sorensen [1983] also show that (1.1) has no solutions s with $\|s\| = \Delta$ if and only if H is positive definite and $\|H^{-1}g\| < \Delta$. The important consequence of lemma(2.3) is that when (1.1) has a solution on the boundary of $\{w: \|w\| \leq \Delta\}$ then that solution can be found by solving

$$\|s_\alpha\|^2 - \Delta^2 = 0, \qquad \text{where } s_\alpha = -(H + \alpha I)^{-1}g. \tag{2.4}$$

Thus, except for a special case, (1.1) has been reduced to a zero finding problem in one variable, α. The next crucial observation, due to Reinsch [1967,1971] and Hebden [1973], is that (2.4) is a rational function in α with second order poles on a subset of the negatives of the eigenvalues of the matrix H. This follows easily from the Real–Schur decomposition,

$$H = Q\Lambda Q^T \quad \text{where } \Lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_n) \text{ and } Q^TQ = I. \tag{2.5}$$

Moreover, we assume that the eigenvalues are ordered so that $\lambda_1 \leq \ldots \leq \lambda_n$.

We now have

$$\|s_\alpha\|^2 = \|Q(\Lambda + \alpha I)^{-1} Q^T g\|^2 = \sum_{i=1}^{n} \frac{\gamma_i^2}{(\lambda_i + \alpha)^2} \tag{2.6}$$

where γ_i is the ith component of $Q^T g$. Let S_1 denote the eigenspace corresponding to the smallest eigenvalue, λ_1, of H. If g has a component in S_1 then (2.4) has a solution in $(-\lambda_1, \infty)$ and this is called the "general case". If, however, g is perpendicular to S_1 then (2.4) may not have a solution in $(-\lambda_1, \infty)$ and this leads to numerical difficulties. This situation is called the "hard case". As Moré and Sorensen point out, the characteristic difficulty of the hard case is that $\|s_\alpha\| < \Delta$ whenever $H + \alpha I$ is positive definite. In the hard case, a solution to (1.1) can be obtained by solving

$$(H - \lambda_1 I)s = -g \tag{2.7}$$

for s with $\|s\| < \Delta$ and by determining an eigenvector $z \in S_1$. Then

$$(H - \lambda I)(s + \tau z) = -g \tag{2.8}$$

and $\|s + \tau z\| = \Delta$ for some τ. Lemma(2.3) now shows that $s + \tau z$ solves (1.1).

It is now easier to see how the ill-conditioning in matrices of the form (1.7) causes problems. Specifically, the solution to equation (2.2) becomes unstable as μ tends to zero. Gould [1986] offers a numerically stable way to solve such systems. Essentially the idea is to avoid using the true Hessian and instead compute the trust region steps with an extended matrix whose condition number is normally much better. If $H + \alpha I = B + \frac{1}{\mu} AA^T + \alpha I$, as in (1.7), we then set

$$X = \begin{pmatrix} B + \alpha I & A \\ A^T & -\mu I \end{pmatrix} \tag{2.9}$$

and if

$$X \begin{pmatrix} s_\alpha \\ r \end{pmatrix} = \begin{pmatrix} -\nabla f \\ -c \end{pmatrix} \tag{2.10}$$

for some $s_\alpha \in R^n$ and $r \in R^t$, then $(H + \alpha I)s = -\nabla f(x) - \frac{1}{\mu} Ac(x) = -g$. This way of solving (2.2) will play an instrumental role in the algorithm for solving (1.1). It is important to note that X does become ill-conditioned as $\mu \to 0$ if A is nearly singular. While there is a way to cope with that situation we shall reserve the discussion for another paper. For now we assume that A has full column rank in a neighborhood of the minimum of $p(x)$. Every iteration of our algorithm will need to know if $H + \alpha I$ is positive definite. Therefore, the following definition and lemma are in order.

Definition(2.11): The inertia of an $n \times n$ symmetric matrix H is the triple $In(H) = (a, b, c)$ where a, b, c are respectively the number of positive, negative, and zero eigenvalues of H respectively.

Lemma(2.12): $In(H + \alpha I) = (a, b, c)$ if and only if $In(X) = (a, b + t, c)$.

Proof: See Gould [1986].

III. The algorithm.

The development here will, as in Section 2, be much the same as that of Moré and Sorensen [1983] but will include those modifications necessary to make the algorithm perform well on penalty functions. Specifically, we present a condition estimator based on the Bunch-Kaufman [1977] factorization which, in the limit as $\alpha \to -\lambda_1^+$, returns a vector $\hat{z}$ of unit norm such that $\hat{z}^T(H + \alpha I)\hat{z} \to 0$. We also derive a starting value for α that will usually be close to α^* (the value of α that solves 2.4).

The algorithmic development begins with a search for a good way to solve

$$\Phi(\alpha) = \|s_\alpha\|^2 - \Delta^2 = 0, \qquad \text{where } s_\alpha = -(H + \alpha I)^{-1} g \text{ and } \alpha > -\lambda_1(H). \tag{3.1}$$

It is well known that $\Phi(\alpha)$ is convex and decreasing on $(-\lambda_1, \infty)$. When Newton's method is applied to such functions α_+ (the updated value of α) is always less than α^*, and if $\alpha_c > \alpha^*$ then α_+ may be much less than α^*. Dennis and Schnabel [1983] attempted to overcome the problem of excessively short or long steps by modeling (3.1) with a single term rational function and derived the following update.

$$\alpha_+ = \alpha_c + \left(\frac{\|s_{\alpha_c}\|^2}{\Delta}\right)\left[\frac{\|s_\alpha\| - \Delta}{s_\alpha^T (H + \alpha I)^{-1} s_\alpha}\right]. \tag{3.2}$$

Moré and Sorensen [1983] obtain the same update by applying Newton's method to the function

$$\phi(\alpha) = \frac{1}{\Delta} - \frac{1}{\|s_\alpha\|} = 0 \tag{3.3}$$

In practice we have observed that (3.1) works well when $\alpha_c < \alpha^*$ but it still generally overcorrects when $\alpha_c > \alpha^*$. This isn't surprising since a one term approximation to (3.1) is valid only near $-\lambda_1$. Moreover, if we examine

$$\phi''(\alpha) = 3\|s_\alpha\|^{-3}\{s_\alpha^T(H + \alpha I)^{-2} s_\alpha \; - \; \|s_\alpha\|^{-2}[s_\alpha^T(H + \alpha I)^{-1} s_\alpha]^2\} \tag{3.4}$$

and apply the Cauchy-Schwartz inequality to the term in square brackets we see that $\phi''(\alpha) \geq 0$ with equality when s_α is an eigenvector or when $H = \beta I$ for some β. Moreover, if we let $v = (H + \alpha I)^{-1} s_\alpha$ then

$$\phi''(\alpha) = \frac{3\|v\|^2}{\|s_\alpha\|} \sin^2\theta \tag{3.5}$$

where θ is the angle between v and s_α. Since s_α is not, in general, an eigenvector and H can be massively ill-conditioned we have no reason to expect θ to be small. It follows that for typical values of α^* (i.e., $O(-\lambda_1(H))$) $\phi''(\alpha)$ will usually be much greater than zero. Thus the ill-conditioning of H builds a certain amount of nonlinearity into (3.1) and (3.3). Moré and Sorensen[1983] point out that $\phi''(\alpha) \to 0$ as $\alpha \to \infty$. For small values of μ, however, $\phi''(\alpha)$ tends to zero rather slowly so it is unreasonable to expect Newton's method to work well outside a small neighborhood of α^*.

If a method is to produce a good update for α when $\alpha_c > \alpha^*$ it must in some way pay attention to the curvature of ϕ. With this goal in mind we will try to construct a meaningfull quadratic model of ϕ and then let α_+ be the left root of the model. For the same reason Newton's method failed when $\alpha_c > \alpha^*$, the usual quadratic model (i.e., the first three terms of a Taylor series) also often produces a poor update in this context. That is, $\phi''(\alpha)$ tends to zero as α goes to infinity so that any quadratic model built on local information at α_c will not have enough curvature and will therefore greatly underestimate α^*. A more successful approach is to construct a quadratic model with information acquired at points left and right of α^*. As we shall see, when $\alpha_c > \alpha^*$ the algorithm usually has on hand a close upper bound, λ_S, for $\lambda_1(H)$ and a good approximate eigenvector, $\hat{z}$, corresponding to $\lambda_1(H)$. Since $\phi(-\lambda_1) = 1/\Delta$ and $\phi'(-\lambda_1) = -1/|v_1^T g|$, where v_1 is the exact eigenvector corresponding to λ_1, we can compute resonably accurate information about ϕ at a point to the left of α^*. We are now poised to state the basic algorithm.

Algorithm(3.6): Let $\alpha \geq 0$ with $H + \alpha I$, as in (1.7), positive definite and $\Delta > 0$ be given.

1) Form X and compute its Bunch-Kaufman [1977] factorization, $X = US^{-1}SDSS^{-1}U^T$, where U is a product of elementary unit upper triangular and permutation matrices, D is a block diagonal matrix with blocks of order 1 or 2, and S is a scaling matrix chosen so that the entries of US^{-1} are bounded.

2) Solve $X \begin{pmatrix} s_\alpha \\ r \end{pmatrix} = \begin{pmatrix} -\nabla f \\ -c \end{pmatrix}$, as in (2.9).

3) Solve $X \begin{pmatrix} q \\ w \end{pmatrix} = \begin{pmatrix} s_\alpha \\ 0 \end{pmatrix}$

4) **If** $\|s_\alpha\| > \Delta$ **then**
   Update α via (3.2)
   Elseif $\phi(\alpha_c) > -1/\Delta$ **then**
   Determine the quadratic, $\chi(\alpha)$, which interpolates the points $(\lambda_S, 1/\Delta)$, $(\alpha_c, \phi(\alpha_c)$ and satisfies $\chi'(\alpha_c) = \phi'(\alpha_c)$.
   Let α_{1+} be the left root of $\chi(\alpha)$.
   Let α_{2+} be the left root of the best local quadratic approximation to ϕ at α_c (i.e., the first three terms of a Taylor series).

If $\alpha_{2+} > \lambda_S$ **then**

Let $\alpha_+ = \min\{\alpha_{1+}, \alpha_{2+}\}$.

Else

$\alpha_+ = \alpha_{1+}$

Else

Determine the quadratic, $\chi(\alpha)$, which interpolates the points $(\lambda_S, 1/\Delta), (\alpha_c, \phi(\alpha_c))$ and satisfies $\chi'(\lambda_S) = -1/|\hat{z}^T g|$.

Let α_+ be the left root of $\chi(\alpha)$.

Notice that we have, where possible, used the best local model of ϕ. This was done to ensure quadratic convergence near α^*. In practice we have seen that Algorithm(3.6) is usually robust enough to converge from almost any reasonable starting value. However, to ensure convergence Algorithm(3.6) must, especially in the hard case, be safeguarded. Also, trust region methods usually require only an approximate solution. Thus, it remains to develop good convergence criteria. We shall postpone these matters until we have dealt effectively with the hard case.

In Section 2 we indicated that an eigenvector corresponding to $-\lambda_1$ is required to solve the hard case. Moré and Sorensen [1983] show that a good approximate eigenvector can provide an acceptable inexact solution.

Lemma(3.7): Let $0 < \sigma < 1$ be given and suppose that for $\alpha \geq 0$, $H + \alpha I$ is positive definite and $(H + \alpha I)s_\alpha = -g$. Let $z \in R^n$ satisfy

$$\|s_\alpha + z\| = \Delta, \quad \text{and } z^T(H + \alpha I)z \leq \sigma[s_\alpha^T(H + \alpha I)s_\alpha + \alpha\Delta^2)] \tag{3.8}$$

then

$$-\psi(s_\alpha + z) \geq \frac{1}{2}(1 - \sigma)[s_\alpha^T(H + \alpha I)s_\alpha + \alpha\Delta^2] \geq (1 - \sigma)|\psi(s^*)| \tag{3.9}$$

where $\psi(s^*)$ is the optimal value of (1.1).

Proof: See Moré and Sorensen [1983].

It follows from lemma(3.7) that $|\psi(s_\alpha + z) - \psi(s^*)| \leq \sigma|\psi(s^*)|$ and so $s_\alpha + z$ is a nearly optimal solution to (1.1). Another useful result of lemma(3.7) is that (3.8) provides termination criteria which are applicable to both the general and the hard cases. In our tests we observed that when a very inexact solution (i.e., $\sigma = .02$) is sought termination usually occurs on (3.8) even in the general case. As Moré and Sorensen suggest, anytime we have $\|s_\alpha\| < \Delta$ we try to satisfy (3.8) with $z = \tau\hat{z}$ where $\|s_\alpha + \tau\hat{z}\| = \Delta$ and $\hat{z}$ is an approximate eigenvetor of unit norm corresponding to $-\lambda_1$. A vector $\hat{z}$ with $\|\hat{z}\| = 1$ such that $\hat{z}^T(H + \alpha I)\hat{z}$ is as small as possible can be obtained with the following algorithm.

Algorithm(3.10): Given $H+\alpha I$ positive definite, X and its $US^{-1}SDSS^{-1}U^T$ factorization the following algorithm computes a vector $\hat{z}$ such that $\|\hat{z}\|=1$ and $\hat{z}^T(H+\alpha I)\hat{z}$ is as small as possible.

1) If $U = P_n U_n^{-1} \ldots P_1 U_1^{-1}$, let $P = P_1 P_2 \ldots P_n$ and let $\tilde{U} = PU$. Note that $\tilde{U}$ is upper triangular.
2) Compute the QR factorization of $\tilde{U}DS$. We hasten to point out that since D is tridiagonal, the product $\tilde{U}DS$ is upper Hessenberg and so its QR factorization can be computed in $O(n+t)$ flops.
3) Solve $Rw = e$ where e is a vector of the form $(\pm 1, \ldots, \pm 1)^T$ using the following procedure

   $\theta_i = 0$ **for** $i = 1, \ldots, n+t$

   For $k = n+t, \ldots, 1$

   $w_k^+ = (1+\theta_k)/r_{kk}$

   $w_k^- = (-1+\theta_k)/r_{kk}$

   $$\nu^+ = |w_k^+| + \sum_{i=1}^{k-1} |\theta_i + r_{ik} w_k^+|$$

   $$\nu^- = |w_k^-| + \sum_{i=1}^{k-1} |\theta_i + r_{ik} w_k^-|$$

   If $\nu^+ \geq \nu^-$

   then $w_k = w_k^+$

   else $w_k = w_k^-$

   $\theta_i = \theta_i + r_{ik} w_k$ **for** $i = 1, \ldots, k-1$
4) Solve $S^{-1}\tilde{U}^T x = w$.
5) Solve $Py = x$.
6) If $y = (z_1, z_2)^T$ with $z_1 \in R^n$ and $z_2 \in R^t$ we then set $\hat{z} = z_1/\|z_1\|$.

The only property of $\hat{z}$ required by the algorithm is that $\hat{z}^T(H+\alpha I)\hat{z}$ approach zero as α approaches $-\lambda_1$. The following lemma confirms this.

Lemma(3.11): If $\alpha \to -\lambda_1^+$, then for the $\hat{z}$ from algorithm(3.11), $\hat{z}^T(H+\alpha I)\hat{z} \to 0$.
Proof: See appendix A.

Lastly, given $\hat{z}$ from Algorithm(3.10) the proof of lemma(3.7) shows that the correct choice of τ is

$$\tau = \frac{\Delta^2 - \|s_\alpha\|^2}{s_\alpha^T\hat{z} + sgn(s_\alpha^T\hat{z})[(s_\alpha^T\hat{z})^2 + (\Delta^2 - \|s_\alpha\|^2)]^{1/2}}. \tag{3.12}$$

Algorithm(3.10) will not, in general, be used at every iteration where $\|s_\alpha\| < \Delta$, but rather to generate starting values for inverse iteration. If at some iteration we have $\|s_\alpha\| < \Delta$ and we have an approximate eigenvector, $\hat{z}$, of unit norm from a previous iteration (where $\alpha_p \geq -\lambda_1$) we then solve

$$(H+\alpha I)y = \hat{z} \tag{3.13}$$

and if

$$\hat{y}^T(H+\alpha I)\hat{y} < \hat{z}^T(H+\alpha_p I)\hat{z} \quad \text{where } \hat{y} = y/\|y\| \tag{3.14}$$

we then take $\hat{y}$ to be our new approximate eigenvector. Otherwise we resort to Algorithm(3.10). In practice Algorithm(3.10) is seldom used more than twice within each instance of (1.1).

As it stands it is possible, though unlikely, for algorithm(3.6) to return a value of α_+ which is less than $-\lambda_1$. To prevent this and to guarantee that a solution will be found in finitely many iterations the algorithm must be safeguarded. Moré's and Sorensen's safeguarding scheme is perfectly adequate for this setting and so we use it. An interval containing α^*, $[\lambda_L, \lambda_U]$, and a lower bound, λ_S, for $-\lambda_1$ are maintained at all times. At the beginning of each iteration the value of α_+ inherited from the previous iteration is checked against these parameters.

Safeguard α:

1) $\alpha = \max(\alpha, \lambda_L)$
2) $\alpha = \min(\alpha, \lambda_U)$
3) If $\alpha \leq \lambda_S$ then $\alpha = \max(.001\lambda_U, (\lambda_L, \lambda_U)^{1/2})$

As Moré and Sorensen point out, the third step is probably the most important. Without it, we could not prove that the algorithm will yield an approximate soluton in finitely many iterations. The procedure for updating λ_L, λ_U, and λ_S is straightforward, for details see Moré and Sorensen[1983].

Update λ_L, λ_U, and λ_S:

1) If $H + \alpha I$ is non-singular then
   Compute $\hat{z}$ as described above
   If $H + \alpha I$ is positive definite then
   $\lambda_S = \max\{\lambda_S, \alpha - \hat{z}^T(H + \alpha I)\hat{z}\}$
   Else
   $\lambda_S = \max\{\lambda_S, \alpha, \alpha - \hat{z}^T(H + \alpha I)\hat{z}\}$
2) If $\alpha \in (-\lambda_1, \infty)$ and $\phi(\alpha) < 0$ then
   $\lambda_U = \min(\lambda_U, \alpha)$
   Else
   $\lambda_L = \max(\lambda_L, \alpha)$
3) Let $\lambda_L = \max(\lambda_L, \lambda_S)$.

Initial values for the safeguarding parameters are

$$\begin{aligned}\lambda_S &= \max\{-h_{ii}, i = 1, \ldots, n,\} \\ \lambda_L &= \max\{0, \lambda_S, \frac{\|g\|}{\Delta} - \|H\|_1\} \\ \lambda_U &= \frac{\|g\|}{\Delta} + \|H\|_1\end{aligned} \tag{3.15}$$

where h_{ii} is the i'th diagonal element of $H = B + \frac{1}{\mu}AA^T$.

Since each iteration of the algorithm will require $O(n+t)^3$ it is imperative that we try to minimize the number of iterations needed to find a solution. A poor initial value for α will certainly hinder progress toward a solution. It has been our experience that it is generally worthwhile expending some effort to compute a better starting value for α. The initial value described below proved itself computationally cost effective when compared against less expensive inital values. The structure of the problem precludes the computation of a starting value for α which is both inexpensive and guaranteed to be close to the solution. Thus, we are reduced to heuristic methods. To derive an initial value for α we observe that (2.5) implies

$$\|s_\alpha\|^2 \leq \sum_{i=1}^{n-t} \frac{\gamma_i^2}{(\lambda_1 + \alpha)^2} + \sum_{i=n-t+1}^{n} \frac{\gamma_i^2}{\lambda_{n-t+1}^2} \tag{3.16}$$

and if reasonable estimates for $\sum_{i=1}^{n-t} \gamma_i^2$, $\sum_{i=n-t+1}^{n} \gamma_i^2$, λ_1, and λ_{n-t+1} can be found then we may be able to use (3.16) to compute a suitable initial value for α. The estimates should, if possible, improve as μ tends to zero since the accuracy of the initial values becomes increasingly critical as μ becomes small. To secure such estimates we let

$$A = QR \tag{3.17}$$

where Q is orthogonal and R is upper-triangular. Notice that since Q is orthogonal the last $n - t$ columns, $[q_{t+1}, \ldots, q_n]$, form an orthonormal basis for the null space of A^T (computing a QR factorization of A is even more useful when A is nearly rank deficient in a neighborhood of the solution, but this is the topic of another paper). We now make the following approximations;

$$\sum_{i=1}^{n-t} \gamma_i^2 \approx \sum_{j=t+1}^{n} (q_j^T g)^2 \equiv \delta \tag{3.18}$$

and

$$\sum_{i=n-t+1}^{n} \gamma_i^2 \approx \|g\|^2 - \delta \equiv \zeta. \tag{3.19}$$

Unfortunately, inexpensive "self-improving" estimates for λ_1 and λ_{n-t+1} are not possible. It is easy to see that if $\lambda_1(H) < 0$ then $\lambda_1(B) < \lambda_1(H)$. To compute $\lambda_1(B)$ we reduce B to tridiagonal form with Householder transformations. The characteristic polynomial is then easily formed and its smallest root can be readily found via bisection. For details see Golub and Van Loan [1983] If A has full rank it can be shown that as μ tends to zero $\lambda_{n-t+1}(H)$ converges to $\frac{1}{\mu}\lambda_{n-t+1}(AA^T)$. Furthermore

$$\lambda_{n-t+1}(AA^T) \leq \frac{1}{\mu}\min\{\bar{r}_{ii}^2, \bar{a}_{ii}\} \equiv \rho \tag{3.20}$$

where $\bar{r}_{ii}$ is the element of smallest absolute value along the diagonal of R and $\bar{a}_{ii}$ is the smallest diagonal entry of AA^T. And so we use (3.20) as our approximation for λ_{n-t+1}. Substituting δ, ζ, $\lambda_1(B)$, and ρ for their respective quantities in (3.16) and solving for α gives

$$\alpha \approx \left[\frac{\delta^2}{\Delta^2 - \zeta^2/\rho^2}\right]^{1/2} + |\lambda_1(B)| \tag{3.21}$$

It is certainly possible for the denominator in (3.21) to be negative and in that case we reccommend using $\|B\|_1$ or $\|g\|/\Delta$ for a starting value. (3.21) is most likely to fail when Δ is small or when ρ is small but in these situations α^* (the value of α corresponding to s^*) is generally larger. In other words, α^* is usually inversely proportional to Δ and ρ. The final ingredient of the iteration is the convergence criteria. Here again we invoke, without modification, Moré and Sorensen's work. For convergence parameters σ_1 and σ_2 in (0,1), and an $\alpha \geq 0$ such that $H + \alpha I$ is positive definite, the candidate step s_α ,computed by algorithm(3.6), is checked for near optimality with the following scheme.

Convergence Criteria:

1) If $\big|\Delta - \|s_\alpha\|\big| \leq \sigma_1 \Delta$ or $\|s_\alpha\| \leq \Delta$, $\alpha = 0$ then

   Terminate with $s = s_\alpha$ as the approximate solution.

2) If $\|s_\alpha\| < \Delta$ then

   Use algorith(3.10) to compute $\hat{z}$ and τ

   If s_α, $\hat{z}$, and τ satisfy (3.8) then

   Terminate with $s = s_\alpha + \tau\hat{z}$ as the approximate solution.

Moré and Sorensen[1983] show that these criteria guarantee that if the algorithm terminates then the approximate solution s satisfies

$$\psi(s) - \psi(s^*) \leq \sigma_1(2-\sigma_1)\max\{|\psi(s^*)|, \sigma_2\}, \text{ and } \|s\| \leq (1+\sigma_1)\Delta \tag{3.22}$$

and thus s is a nearly optimal solution of (1.1). We have now discussed all the ingredients of the iterative scheme for solving problem (1.1). The following algorithm summarizes these ingredients and defines a typical iteration.

Algorithm(3.23):

1) Safeguard α.
2) Form X and compute its $US^{-1}SDSS^{-1}U^T$ factorization.
   If $H + \alpha I$ is not positive definite then go to 5).
3) Solve $(H + \alpha I)s_\alpha = -g$ as in Algorithm(3.6).
4) If $\|s_\alpha\| < \Delta$ use Algorithm(3.10) to compute τ and $\hat{z}$.
5) Update λ_L, λ_U, and λ_S.
6) Check the convergence criteria.
7) If $H + \alpha I$ is positive definite and $g \neq 0$ then update
   α via Algorithm(3.6), otherwise update α via $\alpha = \lambda_S$.

Lastly we must now show that the above algorithm will produce a nearly optimal solution in a finite number of steps. Suppose that the length of the interval of uncertainty, $[\lambda_L, \lambda_U]$, remains bounded away from zero. Then the bisection step of the safeguarding scheme can only be invoked finitely many times. If during some iteration a value of α is found such that $\phi(\alpha) > 0$ and $\alpha > -\lambda_1$ then, since Newton's method will procceed monotonically to the solution, an nearly optimal solution can then be found in finitely many more iterations. Now suppose that the length of the interval of uncertainty does not go to zero and that the algorithm never finds a value of α for which $\phi(\alpha) > 0$ and $\alpha > -\lambda_1$. This implies that after a finite number of iterations we have a sequence $\{\alpha_k\}_N^\infty$ which satisfies $\alpha_k > -\lambda_1$, $\phi(\alpha_k) < 0$, and $\alpha_{k+1} < \alpha_k$. We must now show that this sequence converges to $\alpha^* > 0$ in the general case, or $-\lambda_1 > 0$ in the hard case, or 0 in the positive definite case. The key observation to make is that

$$\alpha_{k+1} \leq \frac{\alpha_k - \lambda_S}{1 - \Delta\phi(\alpha_k)} + \lambda_S \tag{3.24}$$

where the right side is the update obtained from applying the secant method to the points $(\lambda_S, 1/\Delta)$ and $(\alpha_k, \phi(\alpha_k))$. (3.24) is true because any quadratic which interpolates those points must be either convex or concave. In either case, the leftmost root satisfies (3.24). Since the secant method is guranteed to converge on convex functions we now see that the method converges in the general case. In the hard case we have $\phi(-\lambda_1) < 0$ and since $\lambda_S \leq -\lambda_1$ it is easy to see that the secant method applied to points $(\lambda_S, 1/\Delta)$ and $(x, \phi(x))$, where $x > -\lambda_1$, will converge to λ_S. It follows that our method will also converge and a similar argument works for the positive definite case.

IV. Computational Results.

In this section we report the results of some tests in which we compare the performance of TRSQPF (our algorithm) with that of GQTPAR (Moré and Sorensen's algorithm) on individual trust region problems where g and H are of the form (1.3) and (1.7) respectively. The tests covered a wide range of values of n, t, and μ and for each triple (n, t, μ) the tables below summarize the the outcome of 10 different trust region problems. Individual problems were created by specifying an $n \times n$ diagonal matrix for D_B, an $n \times t$ diagonal matrix for D_A, an n vector for $\hat{g}$, and a t vector for c. By appropriately choosing the entries in each of these quantities we can construct any of the four main types of trust region problems; general case, hard case, positive definite case, and the saddle point case.

An instance of the general case is formed by choosing the entries of D_B and D_A as uniformly distributed random numbers in $(-1,1)\backslash 0$ and $(-1,-.2) \cup (.2,1)$, respectively. The entries of D_A were so chosen to avoid the case where A is nearly rank deficient. The entries of $\hat{g}$ were chosen as uniformly distributed random numbers in $(-1,1)$. However, the entries of the constraint vector, c, were restricted to the interval $(-\mu_{prev}, \mu_{prev})$ where μ_{prev} is the next largest value of μ used in the testing. For example, if the current value of μ is 10^{-12} then μ_{prev} is 10^{-9}. We feel this scaling of c provides a more realistic test because when TRSQPF is used in the context equality constrained minimization the magnitudes of the constraints are, at any given iteration, usually of the order of the previous value of μ.

To form an instance of the hard case we set to zero the component of $\hat{g}$ corresponding to the smallest element of $[D_{B\,t+1,t+1}, \ldots, D_{B\,n,n}]$. If none of those entries of D_B are negative then $D_{B\,n,n}$ is chosen from the interval $(-1,0)$ and $\hat{g}_n$ is set to zero. We need only consider the last $n-t$ elements of D_B because for the values of μ that we use $(10^{-2}, 10^{-5}, 10^{-9}, 10^{-12}, 10^{-16})$ and our choice for the entries of D_A ensure that the first t eigenvalues $D_b + (1/\mu)D_A D_A^T$ are positive. An instance of the positive definite case was formed by choosing $[D_{B\,t+1,t+1}, \ldots, D_{B\,n,n}]$ from the interval $(0,1)$. The saddle point case is somewhat more involved and will be described a little later.

The next step in constructing a trust region problem is to scramble the structure of D_B and D_A. We set $B = QD_BQ^T$ and $A = QD_AZ$ where Q and Z are respectively $n \times n$ and $t \times t$ orthogonal matrices. Both Q and Z are of the form $Q_1Q_2Q_3$ or $Z_1Z_2Z_3$ where

$$Q_i = I - 2\frac{v_i v_i^T}{v_i^T v_i} \quad \text{and} \quad Z_i = I - 2\frac{w_i w_i^T}{w_i^T w_i}, \quad i = 1,2,3 \tag{4.1}$$

The entries of v_i and w_i are random numbers uniformly distributed in $(-1,1)$. We are now better able to describe the saddle point case. Quite simply, an instance of the saddle point case can be created by setting $g = -\frac{1}{\mu}Ac$. Lastly, the values of Δ were random numbers chosen from the interval $(0,10)$.

The charts also deserve some explanation. For both GQTPAR and TRSQPF we report the minimum, maximum, and average number of iterations used to solve 10 distinct trust region problems for each triple (n,t,μ). For GQTPAR we also note the number of problems (out of 10) on which it ran for 20 iterations. Moré and Sorensen interpret this as a failure to solve the problem and so we report it under the #crash column. The subsequent entries in the min, max, and mean columns were computed from the results of those problems on which GQTPAR halted in less than 20 iterations. Furthermore, to account for the work done by TRSQPF to compute the initial value for α we add an appropriate amount to the average number of iterations. For specific values of n and t number of flops required to compute the initial α is

$$\frac{2}{3}n^3 + t^2(n - t/3) \tag{4.2}$$

while the work count per iteration of TRSQPF is given by the formula

$$\frac{(n+t)^3}{6} + 7(n+t)^2. \tag{4.3}$$

The adjustment is made by evaluating the above formulas and dividing (4.3) into (4.2). For example, for $n = 40$ and $t = 20$ the ratio of (4.2) to (4.3) is .9 and this is the amount we add to the average number of iterations. Thus, the average number of floating point operations (flops) per problem can be obtained by multiplying the adjusted mean number of iterations times the number of operations per iteration (i.e., equation 4.3). In all problems $\sigma_1 = 0.01$ and $\sigma_2 = 0.0$.

			General Case							
			GQTPAR				TRSQPF			
n	t	μ	# crash	min	max	mean	min	max	mean	adjst. mean
20	5	10^{-2}	0	5	9	6.4	1	5	2.9	3.8
		10^{-5}	0	5	10	7.4	2	4	2.7	3.6
		10^{-9}	0	4	11	8.8	2	4	2.8	3.7
		10^{-12}	0	4	13	9.0	2	4	2.7	3.6
		10^{-16}	8	5	6	5.5	2	5	3.4	4.3
	10	10^{-2}	0	3	9	5.7	1	7	3.9	4.5
		10^{-5}	0	4	10	7.2	1	4	2.5	3.1
		10^{-9}	0	5	14	9.2	2	4	2.9	3.5
		10^{-12}	0	3	11	6.7	2	5	3.0	3.6
		10^{-16}	9	8	8	8.0	2	4	3.1	3.7
	15	10^{-2}	0	3	8	5.3	1	5	3.4	4.0
		10^{-5}	0	6	11	7.4	1	7	3.8	4.4
		10^{-9}	0	5	13	9.4	2	4	2.6	3.2
		10^{-12}	0	2	9	6.1	1	4	2.7	3.3
		10^{-16}	9	7	7	7.0	1	4	3.0	3.6

n	t	μ	General Case: GQTPAR # crash	GQTPAR min	GQTPAR max	GQTPAR mean	TRSQPF min	TRSQPF max	TRSQPF mean	TRSQPF adjst. mean
40	10	10^{-2}	0	3	9	6.7	1	5	3.3	4.5
		10^{-5}	0	5	10	7.3	1	5	3.2	4.4
		10^{-9}	0	5	10	7.9	2	4	3.2	4.4
		10^{-12}	0	3	12	7.6	2	4	3.3	4.5
		10^{-16}	7	2	9	6.0	2	4	3.2	4.4
	20	10^{-2}	0	3	7	5.6	1	6	3.3	4.2
		10^{-5}	0	5	9	7.2	2	6	3.3	4.2
		10^{-9}	0	5	12	8.9	2	4	2.8	3.7
		10^{-12}	0	3	12	6.9	2	4	3.6	4.5
		10^{-16}	7	7	10	8.7	2	5	2.6	3.5
	30	10^{-2}	0	4	9	5.2	2	5	3.2	3.6
		10^{-5}	0	4	10	7.5	1	6	4.2	4.6
		10^{-9}	0	6	13	9.4	2	4	2.8	3.2
		10^{-12}	0	3	12	8.5	2	4	3.2	3.6
		10^{-16}	10	-	-	-	2	5	2.8	3.2

n	t	μ	General Case GQTPAR # crash	min	max	mean	TRSQPF min	max	mean	adjst. mean
80	20	10^{-2}	0	4	9	6.4	1	6	3.9	5.5
		10^{-5}	0	5	10	7.0	1	5	3.7	5.3
		10^{-9}	0	4	12	7.6	2	4	3.3	4.9
		10^{-12}	0	3	10	7.6	2	5	3.6	5.2
		10^{-16}	5	8	13	6.3	2	5	3.8	5.4
	40	10^{-2}	0	3	9	5.5	1	5	3.5	4.7
		10^{-5}	0	4	11	7.5	4	7	5.3	6.5
		10^{-9}	0	5	13	9.0	2	4	3.3	4.5
		10^{-12}	0	2	12	7.0	2	5	3.5	4.7
		10^{-16}	6	5	8	6.8	2	5	3.2	4.4
	60	10^{-2}	0	3	6	4.6	1	6	3.6	4.5
		10^{-5}	0	3	10	8.1	3	6	5.2	6.1
		10^{-9}	0	5	11	8.7	2	6	3.2	4.1
		10^{-12}	0	3	11	7.5	2	6	3.2	4.1
		10^{-16}	7	2	7	5.3	2	4	2.5	3.4

			Hard Case							
			GQTPAR				TRSQPF			
n	t	μ	# crash	min	max	mean	min	max	mean	adjst. mean
20	5	10^{-2}	0	4	6	5.4	2	4	2.8	3.8
		10^{-5}	0	4	11	7.8	3	5	3.3	4.2
		10^{-9}	0	5	11	8.9	2	6	3.5	4.4
		10^{-12}	0	3	12	8.1	2	5	3.5	4.4
		10^{-16}	9	2	2	2.0	2	5	3.6	4.5
	10	10^{-2}	0	3	9	5.9	2	5	3.9	4.5
		10^{-5}	0	4	11	7.5	2	4	2.9	3.5
		10^{-9}	0	5	13	9.3	2	5	3.3	3.9
		10^{-12}	0	3	12	7.8	3	6	4.2	4.8
		10^{-16}	8	6	7	6.5	3	5	3.8	4.4
	15	10^{-2}	0	3	8	5.7	1	4	2.3	2.9
		10^{-5}	0	4	10	7.7	1	6	3.8	4.4
		10^{-9}	0	6	10	8.5	2	7	4.6	5.2
		10^{-12}	0	4	12	8.9	2	6	4.4	5.0
		10^{-16}	9	7	7	7.0	2	7	4.1	4.7

			Hard Case							
			GQTPAR				TRSQPF			
n	t	μ	# crash	min	max	mean	min	max	mean	adjst. mean
40	10	10^{-2}	0	3	9	5.5	1	5	3.6	4.8
		10^{-5}	0	5	10	7.3	1	4	3.0	4.2
		10^{-9}	0	4	12	8.1	2	4	3.5	4.7
		10^{-12}	0	2	12	7.3	1	6	3.3	4.5
		10^{-16}	6	7	11	8.8	2	5	3.7	4.9
	20	10^{-2}	0	1	8	5.0	1	7	3.3	4.2
		10^{-5}	0	4	9	6.5	2	6	3.9	4.8
		10^{-9}	0	5	14	8.8	2	4	3.0	3.9
		10^{-12}	0	2	13	8.0	2	5	3.5	4.4
		10^{-16}	7	9	11	9.7	3	5	3.7	4.6
	30	10^{-2}	0	3	8	5.4	1	6	3.6	4.0
		10^{-5}	0	4	10	7.5	2	7	5.0	5.4
		10^{-9}	0	6	12	9.4	3	6	3.9	4.3
		10^{-12}	0	3	11	7.9	3	7	4.0	4.4
		10^{-16}	8	5	11	8.0	2	6	4.5	4.9

			Hard Case							
			GQTPAR				TRSQPF			
n	t	μ	# crash	min	max	mean	min	max	mean	adjst. mean
80	20	10^{-2}	0	3	7	5.1	1	6	3.5	5.1
		10^{-5}	0	4	11	7.5	2	5	3.3	4.9
		10^{-9}	0	5	12	8.5	2	4	3.5	5.1
		10^{-12}	0	3	11	7.3	2	4	3.4	5.0
		10^{-16}	5	2	9	8.8	2	4	3.0	4.6
	40	10^{-2}	0	3	7	5.5	1	4	3.7	4.9
		10^{-5}	0	4	11	7.2	2	7	4.4	5.6
		10^{-9}	0	5	13	9.4	2	4	3.3	4.5
		10^{-12}	0	3	11	7.3	2	4	3.4	4.6
		10^{-16}	8	4	8	6.0	2	4	3.3	4.5
	60	10^{-2}	0	3	7	4.4	1	6	2.9	3.8
		10^{-5}	0	4	10	7.6	2	7	4.8	5.7
		10^{-9}	0	5	11	8.2	2	5	3.7	4.6
		10^{-12}	0	3	11	8.3	2	4	3.3	4.2
		10^{-16}	3	6	9	7.8	2	5	3.5	4.4

n	t	μ	GQTPAR # crash	GQTPAR min	GQTPAR max	GQTPAR mean	TRSQPF min	TRSQPF max	TRSQPF mean	TRSQPF adjst. mean
			Positive Definite Case							
20	5	10^{-2}	0	2	5	4.0	1	3	2.0	2.9
		10^{-5}	0	4	5	4.3	1	2	1.9	2.8
		10^{-9}	0	3	5	4.2	1	4	2.2	3.1
		10^{-12}	0	2	4	3.4	2	3	2.5	3.4
		10^{-16}	10	-	-	-	2	4	2.4	3.3
	10	10^{-2}	0	2	7	4.3	2	5	3.5	4.1
		10^{-5}	0	2	5	4.2	2	4	2.4	3.0
		10^{-9}	0	4	5	4.3	2	5	2.5	3.1
		10^{-12}	0	2	4	3.0	1	3	2.9	3.5
		10^{-16}	8	7	17	12.0	2	3	2.2	2.8
	15	10^{-2}	0	2	6	4.0	2	5	2.8	3.4
		10^{-5}	0	2	5	3.4	1	4	2.6	3.2
		10^{-9}	0	2	4	3.1	2	2	2.0	2.6
		10^{-12}	0	2	5	2.7	2	3	2.1	2.7
		10^{-16}	8	6	6	6.0	2	3	2.2	2.8

			Positive Definite Case							
			GQTPAR				TRSQPF			
n	t	μ	# crash	min	max	mean	min	max	mean	adjst. mean
40	10	10^{-2}	0	4	5	4.9	1	5	3.5	4.7
		10^{-5}	0	4	5	4.4	2	3	2.1	3.3
		10^{-9}	0	4	5	4.8	2	3	2.4	3.6
		10^{-12}	0	2	5	3.6	2	4	2.3	3.5
		10^{-16}	4	2	11	6.5	1	2	1.9	3.1
	20	10^{-2}	0	3	6	4.7	1	5	3.1	4.0
		10^{-5}	0	4	5	4.8	2	3	2.4	3.3
		10^{-9}	0	4	5	4.6	1	3	2.0	2.9
		10^{-12}	0	2	5	3.8	2	6	2.6	3.5
		10^{-16}	7	2	7	4.0	1	3	2.2	3.1
	30	10^{-2}	0	2	6	3.9	1	3	2.1	2.5
		10^{-5}	0	3	5	3.9	2	4	3.3	3.7
		10^{-9}	0	2	5	3.5	2	5	2.4	2.8
		10^{-12}	0	2	4	3.4	2	3	2.4	2.8
		10^{-16}	10	-	-	-	2	3	2.4	2.8

n	t	μ	GQTPAR # crash	GQTPAR min	GQTPAR max	GQTPAR mean	TRSQPF min	TRSQPF max	TRSQPF mean	TRSQPF adjst. mean
			Positive Definite Case							
80	20	10^{-2}	0	3	6	5.0	1	5	2.8	4.4
		10^{-5}	0	5	5	5.0	2	3	2.1	3.7
		10^{-9}	0	4	5	4.9	2	2	2.0	3.6
		10^{-12}	0	2	5	3.5	1	3	2.0	3.6
		10^{-16}	5	3	9	7.8	2	3	2.2	3.8
	40	10^{-2}	0	2	6	4.9	1	5	3.7	4.9
		10^{-5}	0	4	5	4.7	1	5	3.2	4.4
		10^{-9}	0	4	5	4.7	2	3	2.1	3.3
		10^{-12}	0	2	5	3.4	1	3	2.1	3.3
		10^{-16}	9	5	5	5.0	2	3	2.4	3.6
	60	10^{-2}	0	2	8	4.2	1	5	3.1	4.0
		10^{-5}	0	2	6	4.9	2	5	3.7	4.6
		10^{-9}	0	2	5	4.2	2	4	2.3	3.2
		10^{-12}	0	2	5	3.5	1	3	2.1	3.0
		10^{-16}	6	8	11	9.5	2	3	2.1	3.0

n	t	μ	Saddle Point Case: GQTPAR # crash	GQTPAR min	GQTPAR max	GQTPAR mean	TRSQPF min	TRSQPF max	TRSQPF mean	TRSQPF adjst. mean
20	5	10^{-2}	0	4	12	7.4	1	4	1.3	2.1
		10^{-5}	0	8	17	10.2	1	5	1.7	2.6
		10^{-9}	0	9	18	12.4	1	3	1.4	2.3
		10^{-12}	0	8	19	13.9	1	2	1.1	2.0
		10^{-16}	10	-	-	-	1	4	1.9	2.8
	10	10^{-2}	0	8	14	10.6	1	6	2.9	3.5
		10^{-5}	0	6	13	9.7	1	11	2.5	3.1
		10^{-9}	0	10	17	13.3	1	7	2.1	2.7
		10^{-12}	1	11	16	13.1	1	7	2.7	3.3
		10^{-16}	9	12	12	12.0	1	6	2.8	3.4
	15	10^{-2}	0	6	12	9.7	1	8	5.3	5.9
		10^{-5}	0	2	15	11.8	3	10	5.4	6.0
		10^{-9}	0	2	17	11.0	1	6	4.1	4.7
		10^{-12}	0	12	18	15.3	1	6	3.7	4.3
		10^{-16}	10	-	-	-	1	6	3.4	4.0

			Saddle Point Case							
			GQTPAR				TRSQPF			
n	t	μ	# crash	min	max	mean	min	max	mean	adjst. mean
40	10	10^{-2}	0	6	16	9.5	1	2	1.3	2.5
		10^{-5}	0	6	15	11.0	1	2	1.1	2.3
		10^{-9}	0	8	15	12.2	1	5	1.4	2.6
		10^{-12}	0	9	17	11.4	1	4	1.3	2.5
		10^{-16}	9	13	13	13.0	1	5	2.3	3.5
	20	10^{-2}	0	7	10	7.8	1	5	1.9	2.8
		10^{-5}	0	7	19	11.1	1	4	2.0	2.9
		10^{-9}	0	7	18	13.1	1	6	2.1	3.0
		10^{-12}	0	10	16	13.4	1	5	2.5	3.4
		10^{-16}	9	14	14	14.0	1	9	2.8	3.7
	30	10^{-2}	0	7	15	11.5	1	6	4.5	4.9
		10^{-5}	0	7	14	11.4	1	6	4.1	4.5
		10^{-9}	0	10	15	12.8	1	7	3.6	4.0
		10^{-12}	0	8	16	13.3	1	6	3.7	4.1
		10^{-16}	9	16	16	16.0	1	6	3.0	3.4

n	t	μ	GQTPAR # crash	GQTPAR min	GQTPAR max	GQTPAR mean	TRSQPF min	TRSQPF max	TRSQPF mean	TRSQPF adjst. mean
			Saddle Point Case							
80	20	10^{-2}	0	7	14	10.1	1	3	1.4	3.0
		10^{-5}	0	8	13	10.9	1	3	1.5	3.1
		10^{-9}	0	8	14	11.6	1	4	1.6	3.2
		10^{-12}	0	10	16	13.5	1	2	1.1	2.7
		10^{-16}	8	10	12	11.0	1	1	1.0	2.6
	40	10^{-2}	0	9	16	11.2	1	4	2.1	3.3
		10^{-5}	0	7	15	11.9	1	4	1.8	3.0
		10^{-9}	0	11	17	13.3	1	3	1.5	2.7
		10^{-12}	0	9	18	13.3	1	4	1.6	2.8
		10^{-16}	8	12	16	13.3	1	5	1.9	3.1
	60	10^{-2}	0	6	13	9.1	1	5	3.2	4.1
		10^{-5}	0	6	15	10.2	1	5	2.9	3.8
		10^{-9}	0	11	18	13.7	1	6	3.4	4.3
		10^{-12}	0	11	18	14.9	1	7	3.0	3.9
		10^{-16}	10	-	-	-	1	5	2.4	3.3

To conclude this section we would like to offer explanations for some of the results presented in the preceeding charts. When comparing average numbers of iterations one can see that TRSQPF compares quite favorably with GQTPAR in the general and hard cases. On the basis of some informal experimentation we believe that both the new starting value for α (i.e., equation 3.21) and the more complicated way of updating α (i.e., algorithm 3.6) are responsible. In the hard case TRSQPF is further aided by the condition estimator put forth in Algorithm 3.10. By contrast, the average numbers of iterations in the positive definite case are nearly the same except when $\mu = 10^{-16}$. In the positive definite case both algorithms plug along until a candidate value of α is found which is greater than $-\lambda_1$ and less than zero. When that happens, the safeguarding scheme resets α to zero and the problem is solved. Since neither algorithm is especially well tailored to the positive definite case, neither one is especially adept at it and so the results are unsurprisingly similar. The saddle point case is where TRSQPF registers the sharpest improvement over GQTPAR. The saddle point case is, essentially, a search for the direction of most negative curvature. It stands to reason then that the new initial value for α and the condition estimator in Algorithm 3.10 are the primary factors contributing to TRSQPF's good perfomance in the saddle point case.

In all of the testing presented here we set $\sigma_1 = 0.01$ and $\sigma_2 = 0.0$ which essentially requires both algorithms to return a solution in which the first two significant digits are correct. We feel this represents a realistic test because within the contex of a global algorithm to minimize penalty funtions and ultimately solve equality constrained minimization problems the accuracy requsted of a trust region solver is typically kept fixed. In spite of this, however, some informal tests were conducted in which a much more accurate solution was requested of both TRSQPF and GQTPAR. The results indicated that for a given trust region problem of the type considered in this paper the maximum number of correct significant digits GQTPAR could return was for the most part determined by the value of μ and the machine precision. For example, if μ is 10^{-9} and the machine precision is on the order of 10^{-17} then in general at most the first $17 - 9 = 8$ digits would be correct. TRSQPF, however, would in general return a solution correct to nearly full precision.

Concluding Remarks

Our objective in this work was to develop a technique for the solution of the model trust region problem (1.1) under the assumption that the objective function is the L_2 penalty function. In particular, we have tailored the Moré and Sorensen [1983] algorithm with this specific structure in mind. Our numerical experiments indicate that it can be beneficial to use this modified algorithm. This is pariticularily true when increased accuracy is demanded, the penalty parameter is quite small, or the current quadratic exhibits a saddle point. It should be noted that in this context algorithm GQTPAR requires additional preprocessing since the product AA^T must be explicitly determined. On the other hand, TRSQPF requires computation of the Bunch-Parlett factorization of the extended system which is more expensive than the Cholesky factorization of the Hessian matrix.

With respect to the method presented in this paper there are several avenues of research we intend to pursue. Among the most obvious possible things we could do is to extend the existing method to the sparse case. At first glance it might seem that the technology used to exploit sparsity in the Cholesky factorization could be adapted to the Bunch-Kaufman factorization. The latter, however, utilizes a complicated pivoting scheme which is required to guarantee stability. It is not clear that we can exploit sparsity and maintain stability in the Bunch-Kaufman factorization. Another possible endeavor would be to develop an implementation of the Bunch-Kaufman factorization on a parallel architecture. Here again the pivoting scheme would no doubt hinder ones efforts. Throughout this paper we have assumed that the matrix A is of full column rank. The case where A is rank deficient (or nearly rank deficient) will be the focus of some future work.

The most important remaining project, and the one we plan to dispatch first, is to effectively use the algorithm proposed in this paper in a global method for solving equality constrained optimization problems. Existing methods suffer from often having to decrease μ slowly and consequently can be inefficient. Our goal is to overcome such problems.

Appendix A

We shall now restate and prove lemma(3.11).

Lemma(3.11): If $\alpha \to -\lambda_1^+$, then for the $\hat{z}$ from algorithm(3.11), $\hat{z}^T(H + \alpha I)\hat{z} \to 0$.

Proof: The proof is basically a suitable generalization of the proof given by Moré and Sorensen[1983]. Unforunately, our use of the extended system and the Bunch-Kaufman factorization complicates matters. Therefore, to facilitate understanding the proof we begin by outlining the proof strategy.

I. Show that as $\alpha \to -\lambda_1^+$ and $H + \alpha I$ becomes singular
then the extended system X (equation 2.9) also becomes singular

II. For the vector $\hat{y} = y/\|y\|$ (where y is computed by Alg. 3.10), prove that $\hat{y}^T X \hat{y} \rightarrow 0$ as $\alpha \rightarrow -\lambda_1^+$.

III. If $y^T = (z_1^T, z_2^T)$ where $z_1 \in R^n$ and $z_2 \in R^t$ and $\hat{z} = z_1/\|z_1\|$, show that $\hat{z}^T(H + \alpha I)\hat{z} \rightarrow 0$ as $\alpha \rightarrow -\lambda_1^+$.

Let v_1 be an eigenvector of H corresponding to λ_1. Then $(H + \alpha I)v_1 = (\lambda_1 + \alpha)v_1$ and so $\|(H + \alpha I)v_1\| \rightarrow 0$ as $\alpha \rightarrow -\lambda_1^+$. We now observe

$$\begin{pmatrix} B + \alpha I & A \\ A^T & -\mu I \end{pmatrix} \begin{pmatrix} v_1 \\ \frac{1}{\mu} A^T v_1 \end{pmatrix} = \begin{pmatrix} (\lambda_1 + \alpha)v_1 \\ 0 \end{pmatrix} \tag{A.1}$$

This shows that some singular value of X tends to zero as $\alpha \rightarrow -\lambda_1$. Since $\|X\|_2$ is bounded in a neighborhood of $-\lambda_1$ we may conclude that the determinant of X goes to zero, and hence X becomes singular, as $\alpha \rightarrow -\lambda_1$.

The proof in this section relies heavily of results in Bunch and Kaufman[1977]. They show that if $\tilde{M} = \tilde{U}S^{-1}$ then $|\tilde{M}_{ij}| \leq 1/(1 - .6404)$ and if $\tilde{D} = SDS$ then $|\tilde{D}_{ij}| \leq (2.57)^{n+t-1} \max_{1 \leq i,j \leq n+t} |X_{ij}|$. Again, since the entries of X are bounded in a neighborhood of $-\lambda_1$ it follows that the entries of $\tilde{U}DS$ are also bounded. The orthogonality of Q further implies that the entries of R remain bounded as $\alpha \rightarrow -\lambda_1^+$. Let $\rho > 0$ satisfy

$$\sum_{j=1}^{n+t} \sum_{i=1}^{j} |r_{ij}| \leq \rho \tag{A.2}$$

It follows from the way we compute w_k^+ and w_k^- in Algorithm(3.10) that

$$\max(|w_k^+|, |w_k^-|) \geq \frac{1}{|r_{kk}|} \tag{A.3}$$

and for our choice of w_k we now have

$$\frac{1}{|r_{kk}|} \leq |w_k| + \sum_{i=1}^{k-1} |\theta_i + r_{ik} w_k|. \tag{A.4}$$

This together with equation A.1 implies

$$\frac{1}{|r_{kk}|} \leq (1 + \rho)\|w\| \quad \text{or} \quad \|w\| \geq \frac{1}{(1 + \rho)|r_{kk}|} \tag{A.5}$$

which holds for all k and so

$$\frac{1}{\|w\|} \leq \min(|r_{kk}| : k = 1, \ldots, n + t)(1 + \rho) \quad \textit{or} \quad \|w\| \geq \frac{1}{\min(|r_{kk}|)(1 + \rho)} \tag{A.6}$$

Now since the entries of $S^{-1}\tilde{U}^T$ are bounded there exists a $\delta > 0$ such that

$$\|S^{-1}\tilde{U}^T\|_2 \leq \delta \tag{A.7}$$

Equations A.6, A.7, and the orthogonality of P now imply

$$\|y\| = \|x\| \geq \frac{1}{\delta}\|w\| \geq \frac{1}{\delta(1+\rho)\min(|r_{kk}|)} \tag{A.8}$$

If $\hat{y} = y/\|y\|$ then

$$\hat{y}^T X \hat{y} = \frac{\hat{y}^T e}{\|y\|} \leq \hat{y}^T e[\delta(1+\rho)\min(|r_{kk}|)] \tag{A.9}$$

and by the Cauchy-Schwartz inequality

$$\hat{y}^T X \hat{y} \leq (n+t)^{1/2}\delta(1+\rho)\min(|r_{kk}|) \tag{A.10}$$

It remains to show that $\min(|r_{kk}|) \to 0$ as $\alpha \to -\lambda_1^+$. To do so we must investigate, in some detail, the pivoting strategy and the choice of entries for S used by Bunch and Kaufman[1977]. For $l = n+t, \ldots, 1$ let $A_{ij}^{(l)}$ denote the ij'th entry in the $l \times l$ unfactored matrix remaining at the l'th iteration of the Bunch-Kaufman factorization. At this point S_{ll} is chosen according to the following rule

$$S_{ll} = \begin{cases} 1 & \text{if } (|A_{ll}^{(l)}| \geq \alpha\lambda^{(l)}) \; or \; (|A_{rr}^{(l)}| \geq \alpha\sigma^{(l)}) \; or \; (\lambda^{(l)} \geq \sigma^{(k)}) \\ \min(\sqrt{\alpha\sigma^{(l)}/|A_{ll}^{(l)}|}, \sigma^{(l)}/\lambda^{(l)}) & \text{otherwise} \end{cases} \tag{A.11}$$

where $\alpha = .6404$ and $\lambda^{(l)}$ is the absolute value of the largest off-diagonal element in the last column of $A^{(l)}$. That is

$$\lambda^{(l)} = \max_{1 \leq i < l} |A_{il}^{(l)}| \tag{A.12}$$

and if r is the least integer such that $|A_{rl}^{(l)}| = \lambda^{(l)}$ then $\sigma^{(l)}$ is the largest off-diagonal element in the r'th column of $A^{(l)}$,

$$\sigma^{(l)} = \max_{\substack{1 \leq m \leq l \\ m \neq r}} |A_{mr}^{(l)}|. \tag{A.13}$$

It should be noted that $S_{ll} \neq 1$ only when a 2×2 pivot is used or when $\sigma^{(l)} > \lambda^{(l)}$ in which case a 1×1 pivot is used but no permutation is applied to $A^{(l)}$. We now observe that since $\tilde{U}$ is a product Gauss transformations it has determinant one. It follows that

$$|\det X| = |\det D| \tag{A.14}$$

and

$$|\det R| = |\det DS| \tag{A.15}$$

and hence if the determinant of X is small then so is the determinant of D. More specifically, either one of the 1×1 blocks of D must be small or one of the 2×2 blocks must have a small determinant. For the case where $S_{ll} \neq 1$ and $\sigma^{(l)} > \lambda^{(l)}$ then a 1×1 pivot is used and $D_{ll} = A_{ll}^{(l)}$. Furthermore,

$$(DS)_{ll} \leq \sqrt{\alpha\sigma^{(l)}|A_{ll}^{(l)}|} \tag{A.16}$$

and since $\sigma^{(l)}$ and α are bounded this shows that if a 1×1 block in D has small magnitude then so must the corresponding entry in DS. The case where $S_{ll} \neq 1$ and a 2×2 block is used is more complicated. When this happens

$$\begin{pmatrix} D_{l-1,l-1} & D_{l-1,l} \\ D_{l,l-1} & D_{l,l} \end{pmatrix} = \begin{pmatrix} A_{rr}^{(l)} & \lambda^{(l)} \\ \lambda^{(l)} & A_{ll}^{(l)} \end{pmatrix} \tag{A.17}$$

and the conditions of the pivoting scheme imply

$$-\lambda^{(l)^2}[\frac{\alpha|A_{rr}^{(l)}|}{\sigma^{(l)}} + 1] \leq A_{ll}^{(l)}A_{rr}^{(l)} - \lambda^{(l)^2} \leq -(1-\alpha^2)\lambda^{(l)^2} \leq 0 \tag{A.18}$$

where the center term is the determinant of the 2×2 pivot above. The important implication of equation A.18 is that if the center term is small then so is $\lambda^{(l)}$. Moreover, if the last column of the 2×2 pivot is scaled by S_{ll} then the determinant of the new 2×2 block satisfies

$$S_{ll}|A_{ll}^{(l)}A_{rr}^{(l)} - \lambda^{(l)^2}| \leq \lambda^{(l)}[\alpha|A_{rr}^{(l)}| + \sigma^{(l)}] \tag{A.19}$$

Taken together, these observations imply that $|\det DS| \to 0$ as $|\det D| \to 0$ and by virtue of equation A.15 we now know that $\min(r_{kk}) \to 0$ as $\alpha \to -\lambda_1^+$. This completes the proof of the second part.

To prove the last part we recall that $Xy = PQe$ and if we let $PQe = (u_1, u_2)^T$ then

$$\begin{pmatrix} B+\alpha I & A \\ A^T & -\mu I \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \tag{A.20}$$

from which it follows that

$$\hat{z}^T(H+\alpha I)\hat{z} = \frac{\hat{z}^T U_1 + \frac{1}{\mu}\hat{z}^T A u_2}{\|z_1\|} \tag{A.21}$$

It is easy to see that the numerator of the second term is bounded and hence all we need to show is that $\|z_1\|$ must grow as $\|y\|$ becomes large. Multiplying out equation A.20 gives

$$(B+\alpha I)z_1 + Az_2 = u_1 \tag{A.22}$$

and since A has full column rank and $\|u_1\|$ is bounded we deduce that $\|z_1\| \to \infty$ as $\|z_2\| \to \infty$. This completes the proof.

REFERENCES

J. R. Bunch and L. Kaufman. [1977], *Some stable methods for calculating inertia and solving symmetric linear systems*, Mathematics of Computation, v. 31, pp. 163-179.

J. E. Dennis, Jr. and R. B. Schnabel. [1983], *Numerical methods for unconstrained optimization and nonlinear equations*, Prentice-Hall, Inc., Englewood Cliffs, NJ.

G. H. Golub and C. F. Van Loan. [1983], *Matrix computations*, The Johns Hopkins University Press, Baltimore.

N. I. M. Gould. [1986], *On the accurate determination of search directions for simple differentiable penalty functions*, I.M.A. Journal of Numerical Analysis, v. 6, pp. 357-372.

M. D. Hebden. [1973], *An algorithm for minimization using exact second derivatives*, Atomic Energy Research Esablishment report T.P. 515, Harwell, England.

J. J. Moré and D. C. Sorensen. [1983], *Computing a trust region step*, SIAM J. Sci. Stat. Comput., v. 4, pp. 553-572.

C. H. Reinsch. [1967], *Smoothing by spline functions*, Numer. Math., v. 10, pp. 177-183.

C. H. Reinsch. [1971], *Smoothing by spline functions II*, Numer. Math., v. 16, pp. 451-454.

D. C. Sorensen. [1982], *Newton's method with a model trust-region modification*, SIAM J. Numer. Anal., v. 19, pp. 409-426.