Constrained Nonlinear Least Squares: An Exact Penalty Approach with Projected Structured Quasi-Newton Updates

Nezam Mahdavi-Amiri
Dept. of C.S., York University

Richard Bartels
Dept. of Computer Science

Research Report CS-87-44
July 1987

Constrained Nonlinear Least Squares: An Exact Penalty Approach with Projected Structured Quasi-Newton Updates

Nezam Mahdavi-Amiri

York University
Computer Science Department
Downsview, Ontario
Canada M3J 1P3

Richard H. Bartels

University of Waterloo
Department of Computer Science
Waterloo, Ontario
Canada N2L 3G1

ABSTRACT

This paper is concerned with the development, numerical implementation, and testing of an algorithm for solving constrained nonlinear least squares problems. The algorithm is an adaptation to the least squares case of an exact penalty method for solving nonlinearly constrained optimization problems due to Coleman and Conn. It also draws upon the methods of Nocedal and Overton for handling quasi-Newton updates of projected Hessians, upon the methods of Dennis, Gay, and Welsch for approaching the structure of nonlinear least squares Hessians, and upon the work of Murray and Overton for performing line searches. This method has been tested on a selection of problems listed in the collection of Schittkowski and Hock.

This research was supported under grants number A4076 and A2472 by the Natural Sciences and Engineering Research Council of Canada.

1. Introduction

We survey the exact penalty method due to Coleman and Conn [6-8] and consider its adaptation to the task of solving finite dimensional, generally constrained, nonlinear, least squares problems. The adaptation also draws upon the work of Dennis, Gay, and Welsch [15] for looking at the structure of nonlinear least squares Hessians, upon the work of Nocedal and Overton [30] for projected Hessian updating, and upon the work of Murray and Overton [29] for performing line searches. The adaptation has been tested on a selection of problems listed in the collection of Schittkowski and Hock [35]. This paper presents a case study in the implementation of a special purpose, nonlinear programming algorithm drawing from several sources in the current literature. We believe that it offers a number of contributions.

One contribution lies in the fact that attention has been given to details concerned with making the Coleman-Conn method more "structured" from an algorithmic point of view. The presentation includes inequality constraints, which were absent from the original papers because they were not essential to the theory, but which must be included in a complete implementation. The structuring was done, moreover, with consideration of the fact that some problems might be encountered for which the assumptions of the theory would not hold. We have included some safeguards and heuristics to attempt to detect such situations, and we have introduced other modifications to the Coleman-Conn method following principles laid out in [22] to account for computational issues arising from the use of finite precision arithmetic, all with a view toward adding a measure of robustness to the resulting software.

A second contribution comes from the fact that, to our knowledge, no full, computer implementation of the Coleman-Conn method has yet been made and compared with implementations of other algorithms for constrained optimization. The Coleman-Conn method is distinguished in a number of ways from other current approaches to nonlinear optimization. Although it has a relation to methods based upon the use of quadratic subproblems and an ℓ_1 style of merit function; e.g. the well-known methods of Han and Powell [25,32], it does not employ a quadratic programming substep to determine a descent direction. In this paper we use our adaptation of Coleman-Conn to solve a collection of standard test problems from [35], providing a comparison with, among others, a Han-Powell based program.

A third contribution concerns adapting the Coleman-Conn method to approaches due to Dennis, Gay, and Welsch [15] for treating least squares problems and other approaches due to Nocedal and Overton [30] for projecting quasi-Newton approximations to Hessian matrices. In particular to the Nocedal-Overton approach, this paper provides, to our knowledge, the first numerical testing of the approach within a major computational implementation. It also provides an extension of these projected Hessian techniques to problems with least squares structure.

Section 2 gives a brief survey of some methods for solving unconstrained nonlinear least squares problems, Section 3 surveys the exact penalty approach due to Coleman and Conn for solving nonlinear programming problems. Section 4 concentrates on the details of a projected Hessian update designed for the least squares form of the Hessian. Section 5 covers practical issues relating to the implementation of Sections 3 and 4. Computational results in Section 6 suggest that the algorithm has the general behavior reported for the methods [6-8,15].

2. Unconstrained Nonlinear Least Squares

Let

$$F(x) = [f_1(x), \ldots, f_\ell(x)]^T ,$$

where x has n components, $\ell \geq n$, and the component functions f_δ are twice continuously differentiable. The unconstrained nonlinear least squares problem is

$$\underset{x}{\text{minimize}} \;\; \phi(x) = \frac{1}{2}F(x)^T F(x) = \frac{1}{2}\sum_{\delta=1}^{\ell}[f_\delta(x)]^2 = \frac{1}{2}\|F(x)\|^2 \; . \tag{2.1}$$

($\|\cdot\|$ denotes the Euclidean norm of a vector.) Let $\mathbf{G}(x)$ be the matrix whose columns are the gradients $\nabla f_\delta(x)$ for $\delta = 1, \ldots, \ell$

$$G(x) = [\nabla f_1(x), \ldots, \nabla f_\ell(x)] \ .$$

Then

$$\nabla\phi(x) = G(x)F(x) \tag{2.2}$$

and

$$\nabla^2\phi(x) = G(x)G(x)^T + S(x) \ , \tag{2.3}$$

where $S(x)$ stands for the second order term:

$$S(x) = \sum_{\delta=1}^{\ell} f_\delta(x)\nabla^2 f_\delta(x) \ .$$

A second-order Taylor expansion of ϕ around x is

$$\phi(\bar{x}) = \phi(x+p) \approx \tfrac{1}{2}F(x)^TF(x) + p^TG(x)F(x) + \tfrac{1}{2}p^T[G(x)G(x)^T + S(x)]p \ . \tag{2.4}$$

For the unconstrained nonlinear least squares problem, *Newton's method,* solves the equation

$$(G(x)G(x)^T + S(x))p_N = -G(x)F(x)$$

for the *step direction* p_N in order to estimate a point that will minimize ϕ.

If $\|F\|$ tends to zero as x approaches a minimizer x^*, the second derivative term $S(x)$ also tends to zero. Neglecting the term $S(x)$ altogether produces the *Gauss-Newton method,* in which the step direction p_N is approximated by p_{GN}, the solution of the equations

$$(G(x)G(x)^T)p_{GN} = -G(x)F(x) \ .$$

The step direction p_{GN} can be interpreted as the solution of the linear least squares problem,

$$\underset{p}{\text{minimize}} \ \|G(x)^Tp + F(x)\| \ .$$

The Gauss-Newton method works quite well if $S(x^*)$ is "sufficiently small" and $G(x)$ has full rank; e.g., see Wedin [38-40], Boggs and Dennis [1], and Dennis [12]. Dennis' condition in [12] for being able to neglect $S(x)$ is that for all x in some neighborhood of x^* it is true that

$$\|(G(x) - G(x^*))F(x^*)\| \leq \rho\, \|x - x^*\|$$

for a scalar $\rho > 0$ less than the smallest eigenvalue of $G(x^*)G(x^*)^T$. A situation in which the above would be true, for example, is the one in which the minimum eigenvalue of $G(x)G(x)^T$ is greater than $\|S(x)\|$ for x in a neighborhood of x^*. A problem for which $S(x)$ is negligible is a *small residual* problem.

When $S(x)$ cannot be neglected, an alternative to the Gauss-Newton method is the *Levenberg-Marquardt method,* [26,27], which computes $\bar{x}$ as $x+p_{LM}$, where p_{LM} is the solution of

$$(G(x)G(x)^T + \gamma D)p_{LM} = -G(x)F(x), \tag{2.5}$$

with γ a nonnegative scalar and $\mathbf{D}$ a nonnegative diagonal matrix (often taken to be the identity). This is solved for trial values of γ until $\phi(\bar{x})<\phi(x)$. Fletcher [19] gives heuristic rules for increasing or decreasing γ in an implementation of the Levenberg-Marquardt method. Moré [28] expands Fletcher's ideas, using a diagonal matrix $\mathbf{D}$ whose diagonal entries are appropriately updated at each iteration to scale the variables. In Moré's format (2.5) is solved as the least squares solution to the overdetermined linear system

$$\begin{bmatrix} G(x)^T \\ \sqrt{\gamma D} \end{bmatrix} p_{LM} = \begin{bmatrix} -F(x) \\ 0 \end{bmatrix} \ .$$

Moré's work derives insights from viewing (2.5) as a formula for choosing $p_{LM}=p(\gamma)$ to minimize the second-order expansion (2.4) under the *trust-region restriction* that

$$\| \mathbf{D}^{\frac{1}{2}} p(\gamma) \| \leq \Delta$$

for some bound Δ.

In [34] Salane explores a regularized version of (2.1),

$$\text{minimize } \phi(x,\lambda) = \frac{1}{2}(\| F(x) - \lambda F(b) \|^2 + \lambda a \| \mathbf{D}^{\frac{1}{2}}(x-b) \|^2) \ ,$$

where λ is a continuation parameter lying between 0 and 1, a is a positive weight, and $\mathbf{D}$ is a diagonal scaling matrix.

Gill and Murray [21] account for the difference between p_N, the Newton step direction, and p_{GN}, the Gauss-Newton step direction, by using the singular values decomposition of $\mathbf{G}(x)^T$

$$\mathbf{G}(x)^T = \mathbf{U} \begin{bmatrix} \Sigma \\ \mathbf{0} \end{bmatrix} \mathbf{V}^T \ ,$$

where $\mathbf{U}$,$\mathbf{V}$ are orthogonal and Σ is the diagonal matrix of the ordered singular values of $\mathbf{G}(x)^T$. This decomposition is partitioned according to a separation between the singular values larger and smaller than a selected tolerance

$$\begin{bmatrix} \mathbf{U}_1 \mathbf{U}_2 \mathbf{U}_3 \end{bmatrix} \begin{bmatrix} \Sigma_1 & \mathbf{0} \\ \mathbf{0} & \Sigma_2 \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1^T \\ \mathbf{V}_2^T \end{bmatrix} \ .$$

The *Gill-Murray method* produces a step direction p_{GM} that satisfies

$$p_{GM} = -\mathbf{V}_1 \Sigma_1^{-1} \mathbf{U}_1^T F(x)$$

and constitutes the component of the Gauss-Newton step direction, p_{GN}, lying in the space spanned by the columns of $\mathbf{V}_1$. (p_{GM} will, in fact, be equal to p_{GN} if all singular values of $\mathbf{G}(x)^T$ are large, which makes $\mathbf{V}_2$ and Σ_2 vacuous.) If sufficient decrease of ϕ is not achieved using this step direction, p_{GM} is augmented to solve the system

$$(\mathbf{G}(x)\mathbf{G}(x)^T + \mathbf{B}) p_{GM} = -\mathbf{G}(x) F(x) \ ,$$

where $\mathbf{B}$ is an approximation to $\mathbf{S}(x)$. Gill and Murray tested three possibilities for $\mathbf{B}$. The first used $\mathbf{S}(x)$ itself. The other two used (1) finite differences taken along the step directions defined by the columns of $\mathbf{V}_2$, and (2) a quasi-Newton approximation.

Dennis, Gay, and Welsch [15] made further investigations of quasi-Newton updates to $\mathbf{B} \approx \mathbf{S}(x)$. If (2.4) is used to produce a step direction p, and if the actual step taken is

$$\bar{x} = x + s \ ,$$

where

$$s = \bar{x} - x = \alpha p$$

for a *step length* $\alpha > 0$, then $\mathbf{S}$ will satisfy

$$\begin{aligned} \mathbf{S}(\bar{x}) s &= \sum_{\delta=1}^{\ell} f_\delta(\bar{x}) \nabla^2 f_\delta(\bar{x})(\bar{x} - x) \\ &\approx \sum_{\delta=1}^{\ell} f_\delta(\bar{x})(g_\delta(\bar{x}) - g_\delta(x)) \\ &= \bar{\mathbf{G}}\bar{F} - \mathbf{G}\bar{F} \ . \end{aligned} \tag{2.6}$$

Let

$$y = \overline{\mathbf{G}}\overline{F} - \mathbf{G}\overline{F} = (\overline{\mathbf{G}} - \mathbf{G})\overline{F} \ .$$

In order to approximate $\mathrm{S}(x)$, the standard quasi-Newton approach would start with a matrix $\mathbf{B} \approx \mathrm{S}(x)$ and impose the relation

$$\overline{\mathbf{B}}s = y \tag{2.7}$$

to define an approximation $\overline{\mathbf{B}} \approx \mathrm{S}(\overline{x})$.

The choice of y above is that given by Dennis, et. al. in [15]. There are other reasonable choices for y summarized by Dennis in [14].

The most generally used quasi-Newton strategy employs the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update, [2,18,24,36]. It starts with some symmetric matrix $\mathbf{B} \approx \mathrm{S}(x)$ and produces $\overline{\mathbf{B}} \approx \mathrm{S}(\overline{x})$, also symmetric, satisfying (2.7) above according to

$$\overline{\mathbf{B}} = \mathbf{B} - \frac{\mathbf{B}s(\mathbf{B}s)^T}{s^T\mathbf{B}s} + \frac{yy^T}{y^Ts} \ .$$

Dennis and Moré give a theoretical discussion of this updating formula in [13].

In Dennis, et. al. [15] updating is carried out with a variant of the Davidon-Fletcher-Powell method, [10,17], but BFGS generally has better properties in practice, at least when used for approximations to the full Hessian; e.g., see Powell's comparisons in [33].

Finally, in [11] Brown and Dennis suggested maintaining a quasi-Newton approximation $\mathbf{B}_\delta$ to each individual matrix $\nabla^2 f_\delta(x)$ and representing $\mathrm{S}(x)$ by

$$\mathrm{S}(x) \approx \sum_{\delta=1}^{\ell} f_\delta(x)\mathbf{B}_\delta \ .$$

This approach has been exploited recently by Toint in [37] for solving sparse nonlinear least squares problems.

3. Constrained Nonlinear Least Squares

In this section we reformulate the exact penalty method for the nonlinear programming problem due to Coleman and Conn [6-8] from a "structured" point of view. We include equality constraints, which were omitted from these three cited works for the sake of clarity in presentation, as well as from the paper by Pietrzykowski [31], that provided background for Coleman and Conn's work.

3.1. An Exact Penalty Function

The problem to be solved is

$$\begin{aligned} &\underset{x}{\text{minimize}} \ \ \phi(x) \\ &\text{such that} \ \ c_i(x) = 0, \quad i=1,\ldots,\kappa \\ &\qquad\qquad\quad c_j(x) \geq 0, \quad j=\kappa+1,\ldots,\kappa+m \ . \end{aligned} \tag{3.1.1}$$

The objective function ϕ is $F(x)^TF(x)$ as before, and the c's are functions from $\mathbf{R}^n$ to $\mathbf{R}^1$, which are assumed to be twice continuously differentiable.

As an approach to solving (3.1.1) for general ϕ, Coleman and Conn considered the *penalty function*

$$\psi(x,\mu) = \mu\phi(x) + \sum_{i=1}^{\kappa} |c_i(x)| - \sum_{j=\kappa+1}^{m} \min(0,c_j(x)) \ , \tag{3.1.2}$$

where the *penalty parameter* μ is a positive number. Early mentions of this penalty function are to be found in Zangwill [41], Fiacco and McCormack [16], and Pietrzykowski [31]. The penalty function is not differentiable for any x at which one or more constraints are *active*. (Constraint c_r is active at x if $c_r(x)=0$.) On the other hand the penalty function is *exact* in the sense that, if x^* is an isolated minimizer for (3.1.1) and the gradients of the *active constraints* at x^* are linearly independent, then there exists a real number $\mu^*>0$ such that x^* is also an isolated local minimizer for $\psi(x,\mu)$ for each $0 < \mu \leq \mu^*$.

(Other exact penalty approaches exist; e.g. see [20] for a broader discussion.)

Coleman and Conn propose an algorithm for minimizing ψ for a fixed value of μ. This minimization algorithm for ψ can be extended to an algorithm for solving (3.1.1) as follows:

```
{choose $x$ and $\mu>0$};
feasible := false;
optimal := true;
while ({$\mu$ is large enough} and (not (feasible and optimal)) do
    {reduce $\mu$};
    {minimize $\psi(x,\mu)$ with respect to $x$};
    if (optimal) then
        {test feasibility}
    endif
endwhile;
if ({$\mu$ too small}) then
    if (failure) then
        {report failure}
    elseif (not feasible) then
        {report infeasible}
    else
        {report success}
    endif;
```

It is expected that the "minimize" step of the above will set the *optimal* flag to **true** if it succeeds in finding a minimizer and to **false** if it fails. The "test feasibility" step of the algorithm is expected to set the logical variable *feasible* appropriately. The flag *failure* will be set **true** (and *optimal* will be set **false**) when the "minimize" step detects one or more of a number of failure conditions.

Success consists of ending the **while** loop with μ large enough, and both *feasible* and *optimal* equal to **true**. Notice that the algorithm will not admit defeat unless the "minimize" step reports failure for a sequence of values of μ tending to zero.

The case in which problem (3.1.1) is infeasible is also assessed by detecting infeasibility for a sequence of values of μ tending to zero. On a computer, using finite precision arithmetic, an infinite sequence is not required to arrive at a conclusion of infeasibility. It is enough to reduce μ as indicated and consider it as having "converged" to zero when the term $\mu\phi(x)$ becomes negligible in the computer's arithmetic relative to the summation terms in (3.1.2). Situations do exist in which this approach would not correctly resolve infeasibility, even for exact arithmetic and for an infinite sequence of values of μ. They constitute difficult situations, in general, for nonlinear programming algorithms; see Coleman and Conn [5].

Situations of unboundedness in the general nonlinear programming problem; that is, sequences of feasible x for which $\phi(x)\rightarrow-\infty$, can often be detected in practical terms by the same approach as just described for infeasibility. However, Coleman and Conn rule out unbounded problems in their assumptions, and since we are restricting ourselves to a least squares objective function, unboundedness is not an issue.

The minimization of ψ is carried out with the aid of an ϵ-approximation:

$$\psi_\epsilon(x,\mu) = \mu\phi(x) + \sum_{i\in \mathbf{VE}(x,\epsilon)} \mathrm{sgn}(c_i(x))c_i(x) - \sum_{j\in \mathbf{VI}(x,\epsilon)} c_j(x) \ .$$

The index sets $\mathbf{AC}(x,\epsilon)$, $\mathbf{VE}(x,\epsilon)$, and $\mathbf{VI}(x,\epsilon)$ are defined in terms of an *activity tolerance* $\epsilon\geq 0$ as follows:

(1) the set of *active constraint indices*

$$\mathbf{AC}(x,\epsilon) = \{r: |c_r(x)| \le \epsilon \text{ and } 1 \le r \le \kappa+m\} \ ,$$

(2) the set of *violated equality constraint indices*

$$\mathbf{VE}(x,\epsilon) = \{i: |c_i(x)| > \epsilon \text{ and } 1 \le i \le \kappa\} \ ,$$

(3) the set of *violated inequality constraint indices*

$$\mathbf{VI}(x,\epsilon) = \{j: c_j(x) < \epsilon \text{ and } \kappa+1 \le i \le \kappa+m\} \ .$$

$\psi_\epsilon(x,\mu)$ is clearly equal in value to $\psi(x,\mu)$ when $\epsilon=0$. However, since $\psi_\epsilon(x,\mu)$ excludes active constraints for any $\epsilon \ge 0$, it can be differentiated at any x, unlike $\psi(x,\mu)$.

The minimization of ψ is carried out using several alternative step directions derived from ψ_ϵ:

(i) the *global horizontal step direction*;

(ii) the *dropping step direction*;

(iii) the *Newton step direction*;

and the Newton step direction is composed of two components:

(iii′) the *asymptotic horizontal step direction*;

(iii″) the *vertical step direction*.

For suitably chosen ϵ, and under certain conditions (most notably: two continuous derivatives exist as assumed above, the points produced by the algorithm lie in a compact set, the gradients of all active constraints are linearly independent at each point x encountered, and line search conditions, conditions of positive definiteness, and second order sufficiency conditions hold), the method of Coleman and Conn will converge globally to an isolated minimizer of ψ, and the ultimate rate of convergence will be two-step superlinear.

The following vectors and matrices, defined in terms of the index sets above, play a role in the algorithm:

the *active constraint matrix*

$$\mathbf{A}(x) = \Big[\cdots \nabla c_r(x) \cdots \Big]_{r \in \mathbf{AC}(x,\epsilon)} \ ,$$

the *violated equality constraint matrix*

$$\mathbf{E}(x) = \Big[\cdots \nabla c_i(x) \cdots \Big]_{i \in \mathbf{VE}(x,\epsilon)} \ ,$$

the *violated inequality constraint matrix*

$$\mathbf{I}(x) = \Big[\cdots \nabla c_j(x) \cdots \Big]_{j \in \mathbf{VI}(x,\epsilon)} \ ,$$

the vector of signs

$$\sigma(x) = [\cdots \mathrm{sgn}(c_i(x)) \cdots]_{i \in \mathbf{VE}(x,\epsilon)} \ ,$$

and the vector of 1's

$$e = [1 \cdots 1 \cdots 1] \ .$$

The gradient of ψ_ϵ is

$$\begin{aligned}\nabla\psi_\epsilon(x,\mu) &= \mu\nabla\phi(x) + \mathbf{E}(x)\sigma(x) - \mathbf{I}(x)e \\ &= \mu\mathbf{G}(x)F(x) + \mathbf{E}(x)\sigma(x) - \mathbf{I}(x)e\end{aligned}$$

and the Hessian of ψ_ϵ is

$$\nabla^2\psi_\epsilon(x,\mu) = \mu\nabla^2\phi(x) + \sum_{i \in \mathbf{VE}(x,\epsilon)} \mathrm{sgn}(c_i(x))\nabla^2 c_i(x) - \sum_{j \in \mathbf{VI}(x,\epsilon)} \nabla^2 c_j(x)$$

$$= \mu(\mathbf{G}(x)\mathbf{G}(x)^T + \mathbf{S}(x)) + \sum_{i \in \mathrm{VE}(x,\epsilon)} \mathrm{sgn}(c_i(x)) \nabla^2 c_i(x) - \sum_{j \in \mathrm{VI}(x,\epsilon)} \nabla^2 c_j(x)$$

We will assume that the number of elements in $\mathbf{AC}(x,\epsilon)$ lies strictly between 0 and n. The extreme cases 0 and n require some separate attention in implementing the Coleman-Conn algorithm. The details are important, but they complicate the presentation. The case of *degeneracy*; that is, linear dependence among the gradients of the active constraints, is being ruled out of the presentation. For a discussion of some techniques useful in handling this case, see Busovaca [3].

A necessary condition for x to be an isolated local minimizer for ψ under the assumptions made above on ϕ and the c's is that there exist *multipliers*, λ_r for $r \in \mathbf{AC}(x,\epsilon)$,

(a) such that

$$\nabla \psi_0(x,\mu) = \sum_{r \in \mathbf{AC}(x,0)} \lambda_r \nabla c_r(x) = \mathbf{A}(x)\lambda \ , \tag{3.1.3}$$

(b) and such that

$$-1 < \lambda_r < 1, \ r \in \mathbf{AC}(x,0) \cap \{1, \ldots, \kappa\} \tag{3.1.4}$$

$$0 < \lambda_r < 1, \ r \in \mathbf{AC}(x,0) \cap \{\kappa+1, \ldots, \kappa+m\} \ . \tag{3.1.5}$$

A point, x, for which (a) above is satisfied is a *stationary point* of ψ. A minimizer, then, is a stationary point that satisfies (b). (Note that stationarity and optimality are determined using ψ_0, which is ψ_ϵ with $\epsilon = 0$.)

3.2. The Multiplier Estimates

The estimates of the numbers λ_r decide the steps that are to be used. One of the major premises of the algorithm is that the multipliers are only worth determining in the neighborhoods of stationary points. In such neighborhoods the numbers λ_r are taken to be the least squares solution to

$$\mathbf{A}(x)\lambda = \nabla \psi_\epsilon(x,\mu) \ . \tag{3.2.6}$$

In practice the QR decomposition of $\mathbf{A}(x)$ is used to solve the least squares problem:

$$\mathbf{A}(x) = \mathbf{Q}\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix} = \begin{bmatrix}\mathbf{Y}\ \mathbf{Z}\end{bmatrix}\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix} \ .$$

If t is the number of columns in $\mathbf{A}(x)$, $\mathbf{Z}$ is an $n \times (n-t)$ matrix satisfying $\mathbf{A}(x)^T\mathbf{Z} = 0$ and $\mathbf{Z}^T\mathbf{Z}$ is the identity of order $n-t$.

Nearness to a stationary point is governed by a *stationarity tolerance* $\tau > 0$. The λ's are computed only if the projected gradient

$$\mathbf{Z}\mathbf{Z}^T \nabla \psi_\epsilon(x,\mu)$$

is deemed "small enough" according to this tolerance.

3.3. The Global Horizontal Step Direction

The global horizontal step direction, h_G, is the solution to the problem

$$\underset{h}{\text{minimize}} \ \nabla \psi_\epsilon(x,\mu)^T h + \frac{1}{2} h^T \nabla^2 \psi_\epsilon(x,\mu) h \tag{3.3.7}$$

$$\text{such that } \nabla c_r(x)^T h = 0 \quad \text{for } r \in \mathbf{AC}(x,\epsilon) \ .$$

Using the QR decomposition of $\mathbf{A}(x)$, if we set $h_G = \mathbf{Z}w$ for some $w \in \mathbf{R}^n$, then w is to be found by solving

$$(\mathbf{Z}^T [\nabla^2 \psi_\epsilon(x,\mu)] \mathbf{Z}) w = -\mathbf{Z}^T \nabla \psi_\epsilon(x,\mu) \ . \tag{3.3.8}$$

The step direction h_G is a descent direction for ψ_ϵ at x provided that $(\mathbf{Z}^T [\nabla^2 \psi_\epsilon(x,\mu)] \mathbf{Z})$ is positive definite and $\mathbf{Z}^T \nabla \psi_\epsilon(x,\mu) \neq 0$.

The columns of the matrix $\mathbf{Z}$ are a basis for the null space of $\mathbf{A}(x)$; consequently, h_G is also referred to in the literature as a *null space step direction.* (The "horizontal" terminology, used by Coleman and Conn, derives from the geometric view that the null space of $\mathbf{A}(x)$ parallels the plane tangent at the point x to the active constraint manifold. Movement along the direction h_G corresponds to movement along the manifold.)

Exact second derivatives may be replaced by a quasi-Newton approximation

$$\mathbf{H} \approx \nabla^2\psi_\epsilon(x,\mu) \ ,$$

but in [6-8] it is stressed that it is better to use $\mathbf{H}_Z = \mathbf{Z}^T\mathbf{H}\mathbf{Z}$, a positive definite approximation to $\mathbf{Z}^T[\nabla^2\psi_\epsilon(x,\mu)]\mathbf{Z}$ that is updated according to a quasi-Newton formula. Thus, equation (3.3.8) is actually solved as

$$\mathbf{H}_Z w = -\mathbf{Z}^T\nabla\psi_\epsilon(x,\mu) \ . \tag{3.3.9}$$

This avoids some difficulties with the algorithm in the event that $\mathbf{Z}^T[\nabla^2\psi_\epsilon(x,\mu)]\mathbf{Z}$ is positive definite in a neighborhood of a stationary point whereas $\nabla^2\psi_\epsilon(x,\mu)$ is indefinite there. The alternative, updating $\mathbf{H}$ alone as a quasi-Newton approximation to $\nabla^2\psi_\epsilon(x,\mu)$ and then explicitly forming $\mathbf{Z}^T\mathbf{H}\mathbf{Z}$ whenever needed, is theoretically less robust. Various suggestions have been made regarding projected Hessian updates, see [4,8,30]. We will be exploring the material in [30] here.

3.4. The Asymptotic Horizontal Step Direction

The asymptotic horizontal step direction h_A is the component of the Newton step direction, $h_A + v$, that lies in the null space of $\mathbf{A}(x)^T$. Newton steps are only attempted in the neighborhood of stationary points that are expected to be minimizers. The step direction h_A is the solution to the problem

$$\underset{h}{\text{minimize}} \quad \nabla\psi_\epsilon(x,\mu)^T h + \frac{1}{2}h^T[\nabla^2\psi_\epsilon(x,\mu) - \sum_{r \in \mathbf{AC}(x,\epsilon)} \lambda_r \nabla^2 c_r(x)]h \tag{3.4.10}$$

$$\text{such that} \quad \nabla c_r(x)^T h = 0 \quad \text{for} \quad r \in \mathbf{AC}(x,\epsilon) \ .$$

The solution is computed as in the global case above, except that $\mathbf{H}_Z$ should now be a positive definite approximation to

$$\mathbf{Z}^T[\nabla^2\psi_\epsilon(x,\mu) - \sum_{r \in \mathbf{AC}(x,\epsilon)} \lambda_r \nabla^2 c_r(x)]\mathbf{Z} \ .$$

Since $h_A = \mathbf{Z}w$ for some vector w, this step direction is also a null space direction.

3.5. The Vertical Step Direction

At $x + h_A$, the constraints of $\mathbf{AC}(x,\epsilon)$ may no longer be within ϵ of zero. Using the vertical step direction, v, the constraints of $\mathbf{AC}(x,\epsilon)$ are brought more closely to a value of zero. The vertical step derives from taking a single Newton step toward the value of v that solves the nonlinear equation system

$$c_{\mathbf{AC}(x,\epsilon)}(x + h_A + v) = 0 \ ,$$

where $c_{\mathbf{AC}(x,\epsilon)}$ is the vector of constraint functions, ordered in accord with the columns of $\mathbf{A}(x)$. This means that the vertical step direction, v, is the solution to the system

$$\mathbf{A}(x)^T v = -c_{\mathbf{AC}(x,\epsilon)}(x + h_A) \ .$$

The computation of v uses the QR decomposition of $\mathbf{A}(x)$ as follows:

$$\text{solve} \quad \mathbf{R}^T u = -c_{\mathbf{AC}(x,\epsilon)}(x + h_A) \quad \text{for } u \ ,$$

$$\text{set} \quad v = \mathbf{Y}u \ .$$

The columns of the matrix $\mathbf{Y}$ are a basis for the range space of $\mathbf{A}(x)$; consequently, in the literature v is also referred to as a *range space step direction.* (The "vertical" terminology used by Coleman and Conn derives from the geometric view that, since the range space of $\mathbf{A}(x)$ is orthogonal to the plane tangent at the point x to the active constraint manifold, movement along the direction v corresponds to movement down to the manifold.)

3.6. The Dropping Step Direction

The direction that forces the constraint away from zero, locally and to first order, whose gradient appears as the r^{th} column in the matrix $\mathbf{A}(x)$ is the step direction d that satisfies the system of equations

$$\mathbf{A}(x)^T d = \sigma_r e_r \ , \tag{3.6.11}$$

where e_r is the r^{th} unit vector, and

$$\sigma_r = \begin{cases} -1 \text{ if } \lambda_r > +1 \\ +1 \text{ if } \begin{cases} \lambda_r < 0 \text{ and corresponds to an inequality constraint} \\ \lambda_r < -1 \text{ and corresponds to an equality constraint} \end{cases} \end{cases} .$$

3.7. Strategy for Choosing Step Directions

The steps described above are used, broadly speaking, as follows:

(1) When $\mathbf{Z}\mathbf{Z}^T \nabla \psi_\epsilon(x,\mu)$ is not "small enough" as indicated by the stationarity tolerance τ, then

$\bar{x} \leftarrow x + \alpha h_G$,

where a line search is used to determine $\alpha > 0$.

(2) When $\mathbf{Z}\mathbf{Z}^T \nabla \psi_\epsilon(x,\mu)$ is "small enough" as indicated by the stationarity tolerance τ, the multipliers λ_r, $r \in \mathbf{AC}(x,\epsilon)$, are approximated using the least squares solution to (3.2.6).

(a) If (3.1.4) and (3.1.5) are not satisfied, an index $r \in \mathbf{AC}(x,\epsilon)$ is chosen for which one of (3.1.4) or (3.1.5) is violated, and
$\bar{x} \leftarrow x + \alpha d$,
where a line search is used to determine $\alpha > 0$.

(b) If (3.1.4) and (3.1.5) are satisfied, then
$\bar{x} \leftarrow x + h_A + v$.

Under the standing assumptions on the f's and c's, the steps produced from the ϵ-approximation to the penalty function will produce descent for the penalty function itself, if ϵ and τ are correctly set. Since there is no *a priori* way of knowing how to set these tolerances, the algorithm must include some heuristics for adjusting ϵ and τ whenever a step produced from ψ_ϵ does not produce adequate descent for ψ. The failure of these heuristics must, in turn, be detected and lead to the orderly cessation of the algorithm.

As a final point of terminology, we can call

$$\nabla^2 \psi_\epsilon(x,\mu)$$

the "global Hessian" and

$$\nabla^2 \psi_\epsilon(x,\mu) - \sum_{r \in \mathbf{AC}(x,\epsilon)} \lambda_r \nabla^2 c_r(x)$$

the "local Hessian." Note that the essence of the "global" designation is that the λ's are trivially estimated as zero. Thus, in summary, we wish to provide a quasi-Newton approximation $\mathbf{H}_Z$ to

$$\mathbf{Z}^T \left[\mu \mathbf{G}(x)\mathbf{G}(x)^T + \mu \mathbf{S}(x) + \sum_{i \in \mathbf{VE}(x,\epsilon)} \operatorname{sgn}(c_i(x)) \nabla^2 c_i(x) - \sum_{j \in \mathbf{VI}(x,\epsilon)} \nabla^2 c_i(x) - \sum_{r \in \mathbf{AC}(x,\epsilon)} \lambda_r \nabla^2 c_r(x) \right] \mathbf{Z} \ ,$$

which we will do by setting

$$\mathbf{H}_Z = \mu \mathbf{Z}^T \mathbf{G}(x)\mathbf{G}(x)^T\mathbf{Z} + \mathbf{B}_Z$$

and providing a quasi-Newton approximation

$$\mathbf{B}_Z \approx \mathbf{Z}^T \mathbf{S}(x,\lambda)\mathbf{Z} \quad ,$$

where

$$\begin{aligned}\mathrm{S}(x,\lambda) = \mu\mathrm{S}(x) \quad &+ \sum_{i\,\in\mathrm{VE}(x,\epsilon)} \mathrm{sgn}(c_i(x))\nabla^2 c_i(x) \\ &- \sum_{j\in\mathrm{VI}(x,\epsilon)} \nabla^2 c_i(x) \\ &- \sum_{r\,\in\mathrm{AC}(x,\epsilon)} \lambda_r \nabla^2 c_r(x)\end{aligned}$$

and the values of the λ_r will be taken to be zero in the "global" case.

The above considerations lead to a minimization algorithm for $\psi(x,\mu)$ as follows:

```
{choose $\epsilon>0$ and $\tau>0$};
optimal := false;
failure := false;
first := true;
while ( not optimal and not failure) do
    adequate := true;
    global := true;
    {determine $\mathbf{AC}(x,\epsilon)$, $\mathbf{VE}(x,\epsilon)$, and $\mathbf{VI}(x,\epsilon)$};
    {determine $\mathbf{A}(x)$, $\mathbf{Q}=[\mathbf{Y}\ \mathbf{Z}]$, and $\mathbf{R}$ correspondingly};
    if first then
        first := false;
        {choose initial $\mathbf{B}_Z$};
    endif;
    if ({$\mathbf{Z}\mathbf{Z}^T\nabla\psi_\epsilon(x,\mu)$, tested against $\tau$, is not small enough}) then
        {determine $h_G$};
        {determine $\alpha$ from a line search on $\psi(x,\mu)$};
        if ({sufficient decrease is indicated}) then
            $x := x + \alpha h_G$;
            {update $\mathbf{B}_Z$}
        else
            adequate := false
        endif
    else
        {determine the $\lambda$'s};
        {check whether $\lambda_r$ exists violating (3.1.4) or (3.1.5)};
        if ({$\lambda_r$ exists}) then
            {determine $d$};
            {determine $\alpha$ from line search on $\psi(x,\mu)$};
            if ({sufficient decrease is indicated}) then
                $x := x + \alpha d$;
                {update $\mathbf{B}_Z$}
            else
                adequate := false
            endif
        else
            global := false;
            {determine $h_A$};
            {determine $v$ to solve $\mathbf{A}(x)^T v = -c_{\mathrm{AC}(x,\epsilon)}(x+h_A)$};
```

```
                    p := h_A + v;
                    {check sufficient decrease on ψ(x ,μ)};
                    if ({sufficient decrease is indicated}) then
                          x := x + p;
                          {update B_Z};
                          {test optimality}
                    else
                          adequate := false
                    endif
               endif
          endif;
          if (not adequate) then
               if (global) then
                    if (AC(x,ε) ≠ AC(x,0)) then
                          {reduce ε to change AC(x,ε)}
                    endif
               else
                    {reduce τ so that ZZ^T∇ψ_ε(x,μ) becomes large tested against τ}
               endif;
               if ( (global and AC(x,ε) = AC(x,0))
                          or {ε too small}
                          or {τ too small} ) then
                    failure := true
               endif
          endif
     endwhile;
```

Of course, the "test optimality" step is expected to set the flag *optimal* to **true** when appropriate.

This differs from the flowchart in [6] in certain details, notably having to do with the tests on whether ϵ or τ become too small and on whether sufficient decrease is attained using the step directions h_G and d. Such evidences of failure are detected and result in setting the *failure* flag to **true** so that the algorithm "dies gracefully" when applied to a problem not meeting the assumptions made in [6,7].

The pseudo-code above is still not complete. A full implementation must take account of the special cases for which $\mathbf{AC}(x,\epsilon)$ contains 0, or n, or more than n elements. The former two cases require a number of *if* clauses that, for example, properly interpret $\mathbf{ZZ}^T\nabla\psi_\epsilon(x,\mu)$ in the case that $\mathbf{Z}$ is vacuous, or suppress the vertical step in case $\mathbf{AC}(x,\epsilon)$ is empty, or suppress the horizontal step in case $\mathbf{AC}(x,\epsilon)$ has n elements. The third case requires a degeneracy resolution process that we have provided in our computer implementation by introducing perturbations when required and removing them after resolution is obtained. These details greatly reduce the readability of the pseudo-code.

4. Accommodating the Projected Least Squares Structure

In [30] Nocedal and Overton have discussed an approach to the quasi-Newton updating of projected Hessian approximations for general nonlinear programming. In this section we will adapt their approach to a consideration of the matrix $\mathbf{S}(x_k,\lambda_k)$ where we interpret the multipliers λ_k to be zero or estimated according to (3.2.6), whichever is appropriate at the k^{th} stage of the algorithm.

Consider the asymptotic case. We assume that the final active set has been identified, so that, for all further k,

$$\mathbf{AC}(x_k,\epsilon) = \mathbf{AC}(x,0)\ ,\ \ \mathbf{VE}(x_k,\epsilon) = \mathbf{VE}(x,0)\ ,\ \ \mathbf{VI}(x_k,\epsilon) = \mathbf{VI}(x,0)\ \ .$$

Suppose

$$\mathbf{B}_{Z,k} \approx \mathbf{Z}_k^T\mathbf{S}(x_k,\lambda_k)\mathbf{Z}_k\ \ ,$$

and we wish to update $\mathbf{B}_{Z,k}$ to $\mathbf{B}_{Z,k+1}$ approximating

$$\mathbf{B}_{Z,k+1} \approx \mathbf{Z}_{k+1}^T \mathrm{S}(x_{k+1},\lambda_{k+1})\mathbf{Z}_{k+1} \quad .$$

To derive a secant relationship for $\mathbf{B}_Z$, we resolve the difference in x along the subspaces defined by $\mathbf{Z}_{k+1}$ and $\mathbf{Y}_{k+1}$. We have

$$\bar{x}-x = \overline{\mathbf{Y}}p_Y + \overline{\mathbf{Z}}p_Z \quad ,$$

where, in order to simplify the notation, the presence of a bar above a quantity indicates that it is taken at step $k+1$, and the absence of a bar indicates step k. If the constraints are linear, then $p_{\overline{Y}} = 0$ for $k>0$. In the nonlinear case, asymptotically, we expect $\overline{\mathbf{Y}}p_{\overline{Y}}$ to become negligible. This follows the observations that the approach to an optimum is tangential to the manifold of active constraints, and that the tangent to this manifold corresponding to $\bar{x}$ approaches the tangent to the manifold corresponding to x as k becomes large; i.e., the manifold is suitably approximated by linear functions in the limit.

Let

$$s = \overline{\mathbf{Z}}^T(\bar{x}-x) = p_Z$$

so that

$$\overline{\mathbf{Z}}s = \overline{\mathbf{Z}}p_Z = \bar{x} - x - \overline{\mathbf{Y}}p_Y \quad .$$

Then

$$\begin{aligned}\overline{\mathbf{Z}}^T\mathrm{S}(\bar{x},\bar{\lambda})\overline{\mathbf{Z}}s &= \overline{\mathbf{Z}}^T\mathrm{S}(\bar{x},\bar{\lambda})(\bar{x}-x-\overline{\mathbf{Y}}p_Y) \\ &= \overline{\mathbf{Z}}^T\mathrm{S}(\bar{x},\bar{\lambda})(\bar{x}-x) - \overline{\mathbf{Z}}^T\mathrm{S}(\bar{x},\bar{\lambda})\overline{\mathbf{Y}}p_Y \quad .\end{aligned}$$

If the second term is negligible, then we are left with

$$\begin{aligned}\overline{\mathbf{Z}}^T\mathrm{S}(\bar{x},\bar{\lambda})\overline{\mathbf{Z}}s &\approx \overline{\mathbf{Z}}^T\mathrm{S}(\bar{x},\bar{\lambda})(\bar{x}-x) \\ &= \overline{\mathbf{Z}}^T\Bigg[\mu\sum_{\delta=1}^{\ell} f_\delta(\bar{x})\nabla^2 f_\delta(\bar{x}) \\ &\qquad + \sum_{i\in\mathrm{VE}(x,0)} \mathrm{sgn}(c_i(\bar{x}))\nabla^2 c_i(\bar{x}) \\ &\qquad - \sum_{j\in\mathrm{VI}(x,0)} \nabla^2 c_j(\bar{x}) \\ &\qquad - \sum_{r\in\mathrm{VE}(x,0)} \bar{\lambda}_r\nabla^2 c_r(\bar{x})\Bigg](\bar{x}-x) \\ &\approx \overline{\mathbf{Z}}^T\Big[\mu(\overline{\mathbf{G}}-\mathbf{G})\overline{F} + (\overline{\mathbf{E}}-\mathbf{E})\bar{\sigma} - (\overline{\mathbf{I}}-\mathbf{I})e - (\overline{\mathbf{A}}-\mathbf{A})\bar{\lambda}\Big]\end{aligned}$$

Since $\overline{\mathbf{Z}}^T\overline{\mathbf{A}} = 0$, this can be reformulated as

$$\begin{aligned}&\overline{\mathbf{Z}}^T\mathrm{S}(\bar{x},\bar{\lambda})\overline{\mathbf{Z}}s \\ &\quad = \overline{\mathbf{Z}}^T\Big[\mu(\overline{\mathbf{G}}-\mathbf{G})\overline{F} + (\overline{\mathbf{E}}-\mathbf{E})\bar{\sigma} - (\overline{\mathbf{I}}-\mathbf{I})e + \mathbf{A}\bar{\lambda}\Big] \quad .\end{aligned}$$

Adding and subtracting $\mathbf{A}\lambda$ yields

$$\begin{aligned}&\overline{\mathbf{Z}}^T\mathrm{S}(\bar{x},\bar{\lambda})\overline{\mathbf{Z}}s \\ &\quad = \overline{\mathbf{Z}}^T\Big[\mu(\overline{\mathbf{G}}-\mathbf{G})\overline{F} + (\overline{\mathbf{E}}-\mathbf{E})\bar{\sigma} - (\overline{\mathbf{I}}-\mathbf{I})e + \mathbf{A}(\bar{\lambda}-\lambda) + \mathbf{A}\lambda\Big] \\ &\quad \approx \overline{\mathbf{Z}}^T\Big[\mu(\overline{\mathbf{G}}-\mathbf{G})\overline{F} + (\overline{\mathbf{E}}-\mathbf{E})\bar{\sigma} - (\overline{\mathbf{I}}-\mathbf{I})e + \mathbf{A}\lambda\Big] \quad .\end{aligned}$$

The final formula is consistent with formula (e), page 832, of [30], suitably adapted to the exact penalty function and the least squares structure.

Summarizing, we propose using BFGS to update $\mathbf{B}_Z$ to $\overline{\mathbf{B}}_Z$ using the secant equation

$$\overline{\mathbf{B}}_Z s = y \quad ,$$

where

$$s = \overline{\mathbf{Z}}^T(\overline{x}-x)$$

and

$$y = \overline{\mathbf{Z}}^T\left[\mu(\overline{\mathbf{G}}-\mathbf{G})\overline{F} + (\overline{\mathbf{E}}-\mathbf{E})\overline{\sigma} - (\overline{\mathbf{I}}-\mathbf{I})e + \mathbf{A}\lambda\right] \quad ,$$

and where the full projected Hessian required by the Coleman and Conn algorithm is taken to be

$$\overline{\mathbf{H}}_Z = \mu\overline{\mathbf{Z}}^T\overline{\mathbf{G}}\overline{\mathbf{G}}^T\overline{\mathbf{Z}} + \overline{\mathbf{B}}_Z \quad .$$

It is worth mentioning that, in the unconstrained case, y reduces to the right hand side of the secant equation used in [15]. Consistent with Nocedal and Overton, the BFGS update used to obtain $\overline{\mathbf{B}}_Z$ is to be carried out only if $\overline{\mathbf{Y}}p_Y$ has become negligible (i.e. small relative to $\| s \|$). This means that the update is omitted when

$$\| g \| \geq \frac{\eta}{(k+1)^{1+\nu}} \| s \|$$

for fixed constants η and ν as suggested in [30], where

$$g = \overline{\mathbf{Y}}^T(\overline{x}-x) \quad .$$

Since $\mathbf{B}_Z$ is symmetric, only a triangular portion needs to be stored.

In the global case; i.e. in the case where the set of active constraints has not stabilized, we continue to apply the update so long as the number of active constraints remains constant. When the number of active constraints is *reduced* from the k^{th} to the $k+1^{st}$ step, the "update $\mathbf{B}_Z$" step of the algorithm resets $\mathbf{B}_Z$ to the identity of the appropriate dimensions. When the number of active constraints is *increased* from the k^{th} to the $k+1^{st}$ step, the "update $\mathbf{B}_Z$" step of the algorithm reduces $\mathbf{B}_Z$ by *deleting* an appropriate number of columns from the stored triangular portion of $\mathbf{B}_Z$. Finally, the initial matrix $\mathbf{B}_Z$ is taken to be the identity.

The resetting and initializing steps just described are quite naive. While they usually worked well in the tests, it was sometimes advantageous to initialize and reset using the zero matrix instead of the identity. There is room for improvement in the management of $\mathbf{B}_Z$ for global steps.

5. Computational Considerations for Robustness

In this section we survey the measures taken to implement the foregoing in a robust fashion. Some of what is described in this section is a simple application of the suggestions laid down by Gill, Murray, and Wright in Chapter 8 of [22], the rest of what is described involves computational issues that were left untreated in [6,7].

The projected gradient, $\mathbf{Z}\mathbf{Z}^T\nabla\psi_\epsilon(x,\mu)$, must be tested to determine nearness to stationarity, and it must also be sufficiently small in order for the minimization to be terminated. Since the columns of $\mathbf{Z}$ are orthonormal, it is sufficient to test $\| \mathbf{Z}^T\nabla\psi_\epsilon(x,\mu) \|$ for smallness. Nearness to stationarity is to be determined relative to the tolerance, τ, and acceptability for termination is determined relative to a much smaller tolerance, θ, which must be defined by the user. The test for stationarity takes the form of a relative magnitude test

$$\| \mathbf{Z}^T\nabla\psi_\epsilon(x,\mu) \| \leq \tau \ \text{reference}_1\{ \| \nabla\psi_\epsilon(x,\mu) \| \} \quad , \tag{5.1}$$

where a reference value for $\| \nabla\psi_\epsilon(x,\mu) \|$ is used on the right hand side of the inequality. Coleman and Conn describe their algorithm throughout using the test in which

$$\text{reference}_1\{ \| \nabla\psi_\epsilon(x,\mu) \| \} = \| \nabla\psi_\epsilon(x,\mu) \| \quad .$$

There are many alternatives to this reference value, depending upon the extent to which the implementor wishes to account for orders of magnitude of difference in the values of the norm encountered during the course of the minimization. For any scalar value *val* we are using

$$\text{reference}_1\{val\} = \max(1,val) \ .$$

This avoids underflow, or a too stringent test, when *val* becomes close to zero. The tolerance θ, of course, replaces τ in (5.1) when the tests for convergence are being made.

By the same token, activity is determined by

$$|c_r(x)| \leq \epsilon \ \text{reference}_\psi\{c_r(x)\}$$

and the violation of an equality constraint is determined by the reverse of this inequality. If c_r defines an inequality constraint, it is regarded as violated if

$$c_r(x) < -\epsilon \ \text{reference}_\psi\{c_r(x)\} \ .$$

We are using

$$\text{reference}_\psi\{c_r(x)\} = reference_1 \left\{ \frac{\|F(x)\| + \sum_{r=1}^{\kappa} m \ |c_r(x)|}{\kappa+m+1} \right\} ,$$

which served us as a general purpose, average function value. At more cost in space, a separate reference value for each of the constraint functions would have been more robust for problems in which the constraints have significantly different scales.

Tests for feasibility are carried out exactly as for activity, except that a much smaller, user defined tolerance, γ, is used in place of ϵ.

The values of 10^{-1} and 10^{-2} chosen initially for ϵ and τ, respectively, have served well. The initial value of $\mu=1$ is usually adequate, though other values are sometimes advantageous to use. Changes to μ are naive. It is reduced through division by 8 each time a minimizing x for $\psi(x,\mu)$ is found that is not feasible for (3.1.1). A more sophisticated management of μ would be welcome.

We have been using 10^{-6} for γ and 10^{-4} for θ for all of the problems reported on below.

The value of λ_r must be tested to determine whether it is inside or outside of the interval $[-1,+1]$ or of the interval $[0,+1]$, depending upon whether $1\leq r\leq\kappa$ or $\kappa+1\leq r\leq\kappa+m$, respectively. It is advisable to recognize three cases, in fact, whether λ_r is strictly within its interval, strictly outside, or probably on the boundary. The last case represents a critical situation in which it is impossible to be certain about stationarity *vs.* optimality without gathering higher order information about the objective function and the constraints. To distinguish the cases we have asked, for example, whether

$$\lambda_r < -1 - \theta \ ,$$

or

$$\lambda_r > -1 + \theta \ ,$$

or

$$-1 - \theta \leq \lambda_r \leq -1 + \theta \ ,$$

and similarly for the other critical values, 0 and +1.

The value of μ is regarded as too small when

$$\mu \ \|F(x)\| \leq macheps \ \text{reference}_\psi\{c_r(x)\} \ ,$$

where *macheps* is the "machine epsilon". We regard ϵ to be too small when

$$\epsilon \leq \gamma \ ,$$

and τ is regarded to be too small when

$$\tau \leq \theta \ .$$

The difference in penalty function values, $\psi(\bar{x},\mu)-\psi(x,\mu)$, is to be tested to determine whether sufficient decrease has been obtained. Ideally, a separate test should be applied for each of the step directions h_G, d, and h_A+v. From the theory developed by Coleman and Conn we choose the condition for sufficient decrease on the Newton step to be

$$\psi(x+h_A+v,\mu) - \psi(x,\mu) \le -\beta \text{ reference}_1\{ \| \mathbf{Z}^T \nabla \psi_\epsilon(x,\mu) \|_2^2 + \sum_{r \in \mathbf{A}\mathrm{C}(x,\epsilon)} |c_r(x)| \} \ .$$

The tolerance β was taken to be 10^{-8}.

Sufficient decrease for the other two steps is demanded by setting the parameter *eta* to 0.9 in the line search of [29], which represents a stringent requirement for function decrease. If the line search reports failure to achieve this requirement, this is taken as insufficient decrease. Relying on the line search to report problems proved to be a significant departure from the algorithm charted in [6,7], since that algorithm assumes that sufficient decrease will be obtained when the step h_G is used, and it assumes that sufficient decrease is predictable when the step d is used.

In the event that degeneracy is detected, we have simply added a random perturbation, of the order of magnitude of $\epsilon\,|c_r(x)|$ to $c_r(x)$, to resolve the situation. We remove this perturbation as soon as $\| \bar{x}-x \|$ becomes larger than $\sqrt{macheps}\ \| \bar{x} \|$. This was easy to implement for the tests, but the more sophisticated techniques given in [3] should be kept in mind.

There has been recent discussion in the literature about the continuity of the QR factors of the matrix of active constraint functions in algorithms such as the one being considered here; e. g., see [4,9,23]. The general result has been that the matrices $\mathbf{Y}(x)$ and $\mathbf{Z}(x)$ derived from

$$\mathbf{A}(x) = \begin{bmatrix} \mathbf{Y}(x) & \mathbf{Z}(x) \end{bmatrix} \begin{bmatrix} \mathbf{R}(x) \\ 0 \end{bmatrix}$$

are not necessarily continuous in x unless special care is taken in the process that computes the factorization. We have taken no special care. The algorithm appears not to suffer as a result of this neglect, even though the convergence theory for the algorithm assumes the continuity of $\mathbf{Z}(x)$. This is consistent with evidence that others; e. g. Coleman, have gathered to the effect that the continuity of $\mathbf{Z}(x)$ is more of a theoretical concern than a practical one.

The numerical positive definiteness of the matrix $\mathbf{H}_Z$ is enforced by using the modified Cholesky factorization described on page 111 of [22] during the process of solving (3.3.9).

The requirements for optimality are that, for reasonable choices of reference values,

$$\| \mathbf{Z}^T \nabla \psi_\epsilon(x,\mu) \| \le \theta \text{ reference}_1\{ \| \nabla \psi_\epsilon(x,\mu) \| \} \ ,$$
$$-1+\theta < \lambda_r < +1-\theta \text{ for all } r \in \mathbf{A}\mathrm{C}(x,\epsilon) \text{ and } 1 \le r \le \kappa \ ,$$
$$\theta < \lambda_r < +1-\theta \text{ for all } r \in \mathbf{A}\mathrm{C}(x,\epsilon) \text{ and } \kappa+1 \le r \le \kappa+m \ ,$$
$$|\psi(\bar{x},\mu) - \psi(x,\mu)| \le \theta \text{ reference}_1\{ |\psi(\bar{x},\mu)| \} \ ,$$
$$\| \bar{x} - x \| \le \gamma \text{ reference}_1\{ \| \bar{x} \| \} \ ,$$

In this regard, γ is being used in the spirit of τ_F and θ is being used in the spirit of $\sqrt{\tau_F}$ in the notation of Chapter 8 of [22], and we are avoiding, to a certain extent, the assumptions made in that reference that the problem might be well scaled. One might also be prepared to accept as a case of optimal termination the event that one or more of the above fail to be satisfied, but notice should be given to the user whenever this happens.

The results presented in [6,7]; i.e., that $\psi(x,\mu)$ can be globally minimized and that a rapid asymptotic rate of convergence is achieved, depend upon accurately predicting the behavior of $\psi(x,\mu)$ using $\psi_\epsilon(x,\mu)$. This, in turn, depends upon having reasonable values of ϵ and τ. Since there is no *a priori* way in which the values of τ and ϵ can be known, some positive values are chosen initially, and the algorithm is adjusted to reduce these values whenever the value of the true penalty function at $\bar{x}$ does not show sufficient decrease over its value at x for any of the steps (1), (2a), or (2b) given in Section 3.7.

Indications that $\psi_\epsilon(x,\mu)$ may not be an adequate predictor of the behavior of $\psi(x,\mu)$ are provided at each of the steps described in Section 3.7 whenever an appropriate measure of sufficient decrease in the true penalty function is not achieved. Lack of sufficient decrease in steps (2a) and (2b) were covered by Coleman and Conn, but their assumptions on the objective and constraint functions removed the case of insufficient decrease in step (1). Through the use of a careful line search algorithm, however, sufficient decrease in step (1) can also be monitored, and revisions of tolerances can be made if it is not attained. We have taken the view that insufficient decrease in $\psi(x,\mu)$ on a Newton step is an indication that proximity to a stationary point is being misjudged, and reduce the value of μ in response. We wait until insufficient decrease in $\psi(x,\mu)$ is encountered on h_G or d to reduce the value of ϵ. Cases in which attempts are made to reduce ϵ even though $\mathbf{AC}(x,\epsilon) = \mathbf{AC}(x,0)$; i.e., $\psi(x,\mu) = \psi_\epsilon(x,\mu)$ are signaled as failure. These cases can arise, for example, when insufficient decrease on a line search is detected at a point with no activities.

6. Computational Results

Thirty CNLLS problems were taken from Hock and Schittkowski [35]:

1, 2, 6, 13, 14, 15, 16, 17, 18, 20,
22, 23, 26, 27, 28, 30, 31, 32, 42,
46, 48, 49, 50, 51, 52, 53, 60, 65,
77, 79.

These were chosen because they were least squares problems or could be conveniently cast in that form. The number of variables in these problems varies from 1 to 5, and the number of constraints varies from 1 to 13. The algorithm has successfully solved all of them. The convergence on all problems clearly showed a superlinear rate.

The algorithm was coded in a portable subset of FORTRAN IV and run in double precision arithmetic on the f77 compiler of the 4.2BSD version of UNIX on a VAX 11/750 at York University. Linear algebra services were provided by the LINPACK version of the Basic Linear Algebra Subroutines (BLAS) augmented with a small number of matrix factorization routines provided by Gill, Murray, Wright, and Saunders. The line searches were carried out by the algorithm due to Murry and Overton [29] using a code provided by Michael Overton.

The following table presents the results. The column headed "Problem Number" uses the numbering as given in [35]. The function value obtained by our algorithm is found in the column headed "Function Value" for comparison with those listed. It should be noted that our objective functions corresponded to those reported in [35] multiplied by $\frac{1}{2}$. This was simply a matter of convenience for us in arranging the test problems in least squares format. The number given in the "Number of Iterations" column counts the number of steps through the **while** loop of the minimization for $\psi(x,\mu)$. The "Number of Function Evaluations" column counts the number of times the values of $f_\delta(x)$, $c_i(x)$, and $c_j(x)$ were collectively computed. Each execution of the **while** loop requires one evaluation of each f and c, and many executions of the line search will require one or more additional evaluations of the f's and c's collectively. The spread between the number of iterations and the number of function evaluations provides an impression of how much work was required by the line search. The numbers listed in the column headed "Hock and Schittkowski" give the number of function evaluations required by the best method to attain a function value at least as good as the one we attained. We believe that our method of counting function evaluations is consistent with Hock and Schittkowski's, so that this column serves as a basis for comparison.

Problem Number	Function Value	Number of Iterations	Number of Function Evaluations	Hock and Schittkowski
1	7.307639d–5	18	22	24
2	2.470615d+0	5	7	18
6(1)	1.004591d–22	8	12	10
13(2)	4.921779d–1	9	14	45
14	6.967324d–1	5	8	6
15(3)	1.532500d+2	3	3	5
16(2)	1.256487d–1	11	41	89
17	5.000000d–1	16	23	12
18	2.500000d+0	25	61	8
20	2.009937d+1	32	44	20
22	5.000000d–1	6	9	9
23	1.000000d+1	5	7	7
26	6.939574d–9	24	48	19
27	2.000002d–2	14	24	25
28	9.629650d–35	2	2	5
30	5.000000d–1	2	2	14
31	3.000000d+0	15	22	10
32	5.000000d–1	8	10	3
42	6.055556d+0	3	4	10
46	6.336040d–7	18	33	14
48	1.232600d–32	2	2	7
49	3.662794d–6	11	17	9
50	9.937394d–10	8	9	18
51	2.311116d–23	2	2	5
52	2.663324d+0	7	8	8
53	2.046512d+0	8	9	8
60(1)	1.628410d–2	14	20	9
65	4.767644d–1	11	22	—
77(3)	1.207531d–1	28	55	16
79	3.938841d–2	17	33	10

There is no entry in the last column for problem 65, which serves to indicate that our program reached a minimum value significantly lower than that of any of the methods reported on in [35]. In the first column of the table a superscritp (1) indicates that μ was set to 100 initially; all other problems were started with μ set to 1 initially. The two problems in question, 6 and 60, had constraint functions with values that dominated the penalty function, forcing the the minimization process to take an unusually long time in attempting to maintain the constraints within feasibility. The larger value of μ compensated for the imbalance in scale between the objective function and the constraints. The superscript (2) indicates that μ was set initially to 0.001. In the problems for which this was done, 13 and 16, the imbalance of scale was far on the side of the objective function, causing the minimization to waste time registering a sequence of infeasible stationary points for a sequence of decreasing values for μ. The final superscript (3) points out the problems for which setting $\mathbf{B}_Z$ to zero rather than to the identity, initially and for each reduction in the number of active constraints, had a marked effect on the time to solve the problem. When $\mathbf{B}_Z$ was initialized and reset using the identity, for example, the results for problem 77 were

Problem Number	Function Value	Number of Iterations	Number of Function Evaluations	Hock and Schittkowski
77	1.207531d–1	25	85	16

As already mentioned, the management of $\mathbf{B}_Z$ during the global steps is quite simplistic. The management of the penalty parameter μ is also quite naive. Improvement could be made in both areas. Despite this, the method we have described shows itself very favorably in comparison with those tested by Hock and Schittkowski. It did as well or significantly better than the best of the methods they tested on nearly two thirds of the test problems.

7. Acknowledgements

We are grateful for the assistance of P. E. Gill, W. Murray, and M. L. Overton in providing us with the code for the line search and the Cholesky factorization. S. Busovaca, R. H. Byrd, T. F. Coleman, and A. R. Conn have all contributed helpful suggestions.

8. References

1. P. T. Boggs and J. E. Dennis, Jr., A Stability Analysis for Perturbed Nonlinear Iterative Methods, *Mathematics of Computation* **30** pp. 1-17 (1976).
2. C. G. Broyden, The Convergence of a Class of Double-Rank Minimization Algorithms, *J. Inst. Maths. Applications* **6** pp. 76-90 (1970).
3. Ph.D., *Handling Degeneracy in a Nonlinear L-1 Algorithm,* University of Waterloo, Waterloo, Ontario, Canada (1985).
4. R. H. Byrd and R. B. Schnabel, Continuity of the Null Space Basis and Constrained Optimization, CU-CS-272-84, The University of Colorado, Boulder, Colorado, USA 80309 (1984).
5. T. F. Coleman and A. R. Conn, Second-order Conditions for an Exact Penalty Function, *Mathematical Programming* **19** pp. 155-177 North-Holland, (1980).
6. T. F. Coleman and A. R. Conn, Nonlinear Programming via an Exact Penalty Function: Global Analysis, *Mathematical Programming* **24** pp. 137-161 North-Holland, (1982).
7. T. F. Coleman and A. R. Conn, Nonlinear Programming via an Exact Penalty Function: Asymptotic Analysis, *Mathematical Programming* **24** pp. 123-136 North-Holland, (1982).
8. T. F. Coleman and A. R. Conn, On the Local Convergence of a Quasi-Newton Method for the Nonlinear Programming Problem, *SIAM Journal on Numerical Analysis* **21** pp. 755-769 (1984).
9. T. F. Coleman and D. C. Sorenson, A Note on the Computation of an Orthonormal Basis for the Null Space of a Matrix, *Mathematical Programming* **29** pp. 234-242 North-Holland, (1984).
10. W.C. Davidon, Variable Metric Method for Minimization, ANL-5990 Rev., Argonne National Laboratory (1959).
11. J. E. Dennis, Jr. and K. M. Brown, A New Algorithm for Nonlinear Least Squares Curve Fitting, pp. 348-368 in *Mathematical Software*, ed. J. R. Rice, Academic Press, New York, N. Y. (1971).
12. J. E. Dennis, Jr., Nonlinear Least Squares and Equations, in *the State of the Art of Numerical Analysis*, ed. D. Jacobs, Academic Press, London (1977).
13. J. E. Dennis, Jr. and J. J. Moré, Quasi-Newton Methods: Motivation and Theory, *SIAM Review* **19** pp. 46-89 (1977).
14. J. E. Dennis, Jr., Techniques for Nonlinear Least Squares and Robust Regression, *Commun. Statist.-Simula. Comput. B* **7**(4) pp. 345-359 (1978).

15. J. E. Dennis, Jr., D. M. Gay, and R. E. Welsch, An Adaptive Nonlinear Least-Squares Algorithm, *ACM Transactions on Mathematical Software* **7** pp. 348-383 (1981).

16. A. V. Fiacco and G. P. McCormick, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques,* John Wiley and Sons, New York (1968).

17. R. Fletcher and M. J. D. Powell, A Rapidly Convergent Descent Method for Minimization, *The Computer Journal* **6** pp. 163-168 (1963).

18. R. Fletcher, A New Approach to Variable Metric Algorithms, *The Computer Journal* **13** pp. 317-322 (1970).

19. R. Fletcher, A Modified Marquardt Subroutine for Nonlinear Least Squares, 6799, Atomic Energy Research Establishment, Harwell, England (1971).

20. U. M. García-Palomares, Connections Among Nonlinear Programming, Minimax and Exact Penalty Functions, TM-20, Argonne National Laboratory, 9700 South Cass Ave. Argonne, IL 60439 (1983).

21. P. E. Gill and W. Murray, Algorithms for the Solution of the Nonlinear Least-Squares Problem, *SIAM Journal of Numerical Analysis* **15**(5) pp. 977-992 (1978).

22. P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization,* Academic Press, London (1981).

23. P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, On the Representation of a Basis for the Null Space, SOL 83-19, Stanford University, Stanford, California, USA 94305 (1983).

24. D. Goldfarb, A Family of Variable Metric Updates Derived by Variational Means, *Mathematics of Computation* **24** pp. 23-26 (1970).

25. S-P. Han, Variable Metric Methods for Minimizing a Class of Nondifferentiable Functions, *Mathematical Programming* **20** pp. 1-13 North-Holland, (1981).

26. K. Levenberg, A Method for the Solution of Certain Problems in Least Squares, *Q. Appl. Math.* **2** pp. 164-168 (1944).

27. D. Marquardt, An Algorithm for Least Squares Estimation for Nonlinear Parameters, *SIAM J. Appl. Math.* **11** pp. 431-441 (1963).

28. J. J. Moré, The Levenberg-Marquardt Algorithm: Implementation and Theory Numerical Analysis, pp. 105-116 in *Lecture Notes in Mathematics 630,* ed. G.A. Watson, Springler-Verlag, New York (1977).

29. Walter Murray and Michael L. Overton, Steplength Algorithms for Minimizing a Class of Nondifferentiable Functions, STAN-CS-78-679, Stanford University, Stanford, California (1978).

30. Jorge Nocedal and Michael L. Overton, Projected Hessian Updating Algorithms for Nonlinearly Constrained Optimization, *SIAM Journal on Numerical Analysis* **22**(5) pp. 821-850 (1985).

31. T. Pietrzykowski, An Exact Potential Method for Constrained Maxima, *SIAM J. Anal.* **6** pp. 299-304 (1969).

32. M. J. D. Powell, R. M. Chamberlain, C. Lemarechal, and H. C. Pedersen, The Watchdog Technique for Forcing Convergence in Algorithms for Constrained Optimization, *Tenth International Symposium on Mathematical Progamming,* (1979).

33. M. J. D. Powell, How Bad Are the BFGS and DFP Methods When the Objective Function Is Quadratic?, *Mathematical Programming* **27** pp. 34-47 (1986).

34. D. E. Salane, A Continuation Approach for Solving Large-Residual Nonlinear Least Squares Problems, *SIAM Journal on Scientific and Statistical Computing* **8**(4) pp. 655-671 (1987).

35. K. Schittkowski and W. Hock, *Test Examples for Nonlinear Programming Codes,* Springer-Verlag (1981). Lecture Notes in Economic and Mathematical Systems #187

36. D. F. Shanno, Conditioning of Quasi-Newton Methods for Function Minimization, *Mathematics of Computation* **24** pp. 647-656 (1970).

37. P. L. Toint, On Large Scale Nonlinear Least Squares Calculations, *SIAM Journal on Scientific and Statistical Computing* **8**(3) pp. 393-415 (1987).

38. P-A. Wedin, The Nonlinear Least Squares Problem from a Numerical Point of View, Technical Memoranda I and II, Lund University, Lund, Sweden (1972).

39. P-A. Wedin, On the Gauss-Newton Method for the Non-linear Least Squares Problem, ITM Arbetsrapport No. 24, Inst. for Tellampad Matematik, Box 5073, Stockholm 5, Sweden (1974).

40. P-A. Wedin, On Surface Dependent Properties of Methods for Separable Nonlinear Least Squares Problems, ITM Arbetsrapport No. 23, Inst. for Tellampad Matematik, Box 5073, Stockholm 5, Sweden (1974).

41. W. I. Zangwill, Non-linear Programming via Penalty Functions, *Management Science* **13**(5) pp. 344-358 (1967).