# Constrained Nonlinear Least Squares: An Exact Penalty Approach with 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 finite-dimensional, generally-constrained, nonlinear, least-squares (CNLLS) problems. The algorithm is an adaptation to the CNLLS problem of an exact penalty method for solving nonlinearly constrained optimization problems due to Coleman and Conn. The adaptation draws heavily on the methods of Dennis, Gay, and Welsch for handling the unconstrained, nonlinear, least-squares problem and upon the work of Murray and Overton for performing line searches. This method has been tested on a selection of CNLLS problems listed in the collection of Schittkowski and Hock [32].

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

## 1. Introduction

This paper surveys an adaptation of the exact penalty method due to Coleman and Conn [6,7,9] to the solving of finite-dimensional, generally-constrained, nonlinear, least-squares (CNLLS) problems. The adaptation draws upon the work of Dennis, Gay, and Welsch for solving unconstrained, nonlinear, least-squares problems [14] and upon the work of Murray and Overton for performing line searches [27]. The adaptation has been tested on a selection of CNLLS problems listed in the collection of Schittkowski and Hock [32].

We were primarily interested in two issues while doing this work: in the implementational details concerned with making the Coleman-Conn method somewhat more robust and somewhat more structured and in the behavior that would result in adapting the approach of Dennis, Gay, and Welsch in combination with that of Coleman and Conn. Robustness was expected to result from the incorporation of the careful line search due to Murray and Overton, by following the practical guidelines laid down in chapter 8 of [21], and by taking account in the code for places at which the assumptions in the Coleman-Conn theory might fail to hold.

Section 2 introduces the necessary notation and background, reviews some methods for solving unconstrained nonlinear least squares (UNLLS) problems, and lists updating strategies for the Hessian of the objective function. Section 3 reviews the exact-penalty approach due to Coleman and Conn for solving nonlinear programming problems. Section 4 is concerned with the CNLLS problems, and the adaptation of the Colemann-Conn algorithm is presented. Section 5 lists some practical issues relating to computer implementations of the algorithm. Computational results in section 6 suggest that the algorithm has the general behavior reported for the methods [6,7,9,14].

## 2. Unconstrained Nonlinear Least Squares

In this section we will review the unconstrained, nonlinear least-squares problem (UNLLS). In 2.1 the problem and the special structure of its Hessian and gradient will be given. In section 2.2 the common methods of solution will be outlined. In section 2.3 we will list some quasi-Newton methods that take account of the special structure of the problem.

### 2.1. Preliminaries

The UNLLS problem is

$$\underset{x}{\text{minimize}} \quad \phi(x) = \frac{1}{2}F(x)^T F(x) = \frac{1}{2}\sum_{\delta=1}^{k}[f_\delta(x)]^2 = \frac{1}{2}\|F(x)\|^2 \ , \tag{2.1.1}$$

where

$$x \in \mathbf{R}^n, \ F(x) \in \mathbf{R}^k, \ k \geq n \ .$$

(Throughout this paper $\|\cdot\|$ will denote the Euclidean vector norm and, when applied to a matrix, the corresponding subordinate matrix norm.) This problem arises in curve fitting where we are given values of a dependent variable, $y = y_\delta$, corresponding to values of an independent variable, $\theta = \theta_\delta$, for $\delta = 1, \ldots, k$, and a model $\eta(x, \theta_\delta)$ with unknown parameters $x_1, \ldots, x_n$. Often these parameters are chosen to minimize

$$\frac{1}{2}\sum_{\delta=1}^{k}(\eta(x,\theta_\delta) - y_\delta)^2$$

which is written in the form (2.1.1) with

$$f_\delta(x) = \eta(x,\theta_\delta) - y_\delta \ .$$

For a discussion, as well as numerous examples of how such fitting problems might arise, see [31].

The gradient and the Hessian of $\phi$ are

$$\nabla\phi(x) \ = \ J(x)^T F(x) \tag{2.1.2}$$

and

$$\nabla^2\phi(x) \;=\; J(x)^TJ(x) + \sum_{\delta=1}^{k} f_\delta(x)\nabla^2 f_\delta(x) \;=\; J(x)^TJ(x) + S(x) \qquad (2.1.3)$$

respectively, where $\nabla^2 f_\delta(x)$ is the Hessian matrix of $f_\delta(x)$, and $J(x)$ denotes the Jacobian of $F(x)$ (the $k\times n$ matrix whose $\delta$-th row is $\nabla f_\delta(x)^T$). We are using the notation $S(x)$ for the second order term:

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

The UNLLS problem is a special case of nonlinear minimization, and general nonlinear minimization techniques can be used, but they are most successfully applied in a modified format that takes the structure of the gradient and the Hessian into account.

### 2.2. The UNLLS Techniques

For the UNLLS problem, *Newton's method* (N), ignoring line searches for simplicity, begins with an approximate minimizer $x \approx x^*$ and produces from it a new approximation $\bar{x} \approx x^*$ in the form

$$\bar{x} \;\leftarrow\; x + p_N \;\;,$$

where $p_N$ satisfies the equation

$$(J(x)^TJ(x) + S(x))p_N = -J(x)^TF(x) \;\;.$$

If $\|F\|$ tends to zero as $x$ approaches a minimizer $x^*$, the second derivative term $S(x)$ also tends to zero, and $p_N$ can be approximated by the solution of the equations

$$(J(x)^TJ(x))p_{GN} \;=\; -J(x)^TF(x) \;\;.$$

This produces the *Gauss-Newton method* (GN), and $p_{GN}$ can be interpreted as the solution of the linear least-squares problem,

$$\underset{p}{\text{minimize}} \;\; \|J(x)p + F(x)\| \;\;.$$

The Gauss-Newton method works quite well if $S(x^*)$ is "sufficiently small;" e.g. see Wedin [34,35,36], Boggs and Dennis [1], and Dennis [11]. Dennis' condition in this last-cited reference for being able to neglect $S(x)$ is that for all $x$ in some neighborhood of $x^*$ it is true that

$$\|(J(x) - J(x^*))^TF(x^*)\| \leq \rho\,\|x - x^*\|$$

for a scalar $\rho > 0$ less than the smallest eigenvalue of $J(x^*)^TJ(x^*)$. It is not hard to see that $\rho \geq \|S(x^*)\|$. A problem for which $S(x)$ is negligible is a *small residual* problem.

When $S(x)$ cannot be neglected, a well-known alternative is the *Levenberg-Marquardt method* (LM), [24,25], which computes $\bar{x}$ as $x + p_{LM}$, where $p_{LM}$ is the solution of

$$(J(x)^TJ(x) + \gamma I)p_{LM} = -J(x)^TF(x), \qquad (2.2.4)$$

with $\gamma$ a non-negative scalar. This is solved for trial values of $\gamma$ until $\phi(\bar{x}) < \phi(x)$. Fletcher [18] gives heuristic rules for increasing or decreasing $\gamma$ in an implementation of the Levenberg-Marquardt method. Moré [26] expands Fletcher's ideas and introduces his own tips on creating a "robust" and "efficient" implementation. Instead of $\gamma I$, Moré has $\gamma D^TD$ where $D$ is a diagonal matrix whose diagonal entries are appropriately updated at each iteration to scale the variables. Moré's algorithm is probably the best practical one for the Levenberg-Marquardt procedure.

Gill and Murray [20] account for the difference between $p_N$, the Newton direction, and $p_{GN}$, the Gauss-Newton direction by using the singular-values decomposition of $J(x)$

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

where $U$,$V$ are orthogonal and $\Sigma$ is the diagonal matrix of the ordered singular values of $J(x)$. This decomposition is partitioned according to a separation between the larger and smaller singular values

$$\begin{bmatrix} U_1 U_2 U_3 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} .$$

The *Gill-Murray method* (GM) produces a direction $p_{GM}$ that satisfies

$$p_{GM} = -V_1 \Sigma_1^{-1} U_1^T F(x)$$

and constitutes the component of the Gauss-Newton direction, $p_{GN}$, lying in the space spanned by the columns of $V_1$. ($p_{GM}$ will, in fact, be equal to $p_{GN}$ if all singular values of $J(x)$ are large, which makes $V_2$ and $\Sigma_2$ vacuous.) If sufficient decrease of $\phi$ is not achieved using this direction, $p_{GM}$ is augmented to solve the system

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

where $B$ is an approximation to $S(x)$. Gill and Murray tested three possibilities for $B$. The first used $S(x)$ itself. The other two used (1) finite differences taken along the directions defined by the columns of $V_2$, and (2) a quasi-Newton approximation. Of these latter two, the finite difference approach proved to be better in their tests. Their paper is quite negative in their assessment of (2), but their work precedes that of Dennis, Gay, and Welsch, which reports rather more success in using quasi-Newton updates in order to approximate $S(x)$.

### 2.3. Quasi-Newton Approach

A Taylor expansion of $\phi$ around $x$ is

$$\begin{aligned} \phi(\bar{x}) \approx Q(\bar{x}) = \tfrac{1}{2} F(x)^T F(x) + (\bar{x} - x)^T J(x)^T F(x) \\ + \tfrac{1}{2} (\bar{x} - x)^T [J(x)^T J(x) + S(x)] (\bar{x} - x). \end{aligned} \tag{2.3.5}$$

This means that, if

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

then $S$ will satisfy

$$\begin{aligned} S(\bar{x}) s &= \sum_{\delta=1}^{k} f_\delta(\bar{x}) \nabla^2 f_\delta(\bar{x}) (\bar{x} - x) \\ &\approx \sum_{\delta=1}^{k} f_\delta(\bar{x}) (\nabla f_\delta(\bar{x}) - \nabla f_\delta(x)) \\ &= \bar{J}^T \bar{F} - J^T \bar{F} . \end{aligned} \tag{2.3.6}$$

Let

$$y = \bar{J}^T \bar{F} - J^T \bar{F} .$$

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

$$\bar{B} s = y \tag{2.3.7}$$

to define an approximation $\bar{B} \approx S(\bar{x})$.

The choice of $y$ above is that given by Dennis, et. al. in [14], and so we will denote it by $y_D$. There are other reasonable choices for $y$. Dennis' summary in [13] is as follows:

(1) $y_{BD} = \bar{J}^T \bar{F} - J^T F - \bar{J}^T \bar{J} s$

(2) $y_B = \bar{J}^T \bar{F} - J^T F - J^T J s$

(3) $y_D = (\bar{J} - J)^T \bar{F}$

(4) $y_W = (\bar{J} - J)^T F$ .

Taking convex combinations of pairs of the above has been proposed, too:

(5) $y_G = \bar{J}^T \bar{F} - J^T F - [\alpha \bar{J}^T \bar{J} + (1-\alpha) J^T J]s$

(6) $y_{WD} = [\bar{J} - J]^T [\alpha \bar{F} + (1-\alpha)F]$.

The value of $\alpha = \frac{1}{2}$ has been tested for these choices by Dennis, et. al. [14]. None of the above $y$ has proved to be superior in all situations, but $y_D$ seems to be slightly better than the other versions of $y$, as is claimed by Dennis [13,14].

The two most generally used update strategies are, respectively, the Davidon-Fletcher-Powell (DFP) update, [10,16], and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update, [2,17,23,33]. Each would start with some symmetric matrix $B \approx S(x)$ (for example, $B=0$ or $B=\gamma I$) and produce $\bar{B} \approx S(\bar{x})$, also symmetric, satisfying (2.3.7) above.

The update consistent with the DFP formula would be:

(a) $$\bar{B} = B + \frac{(y-Bs)y^T + y(y-Bs)^T}{(y^T s)} - \frac{(y-Bs)^T s y y^T}{(y^T s)^2} \quad .$$

The update consistent with the BFGS formula would be:

(b) $$\bar{B} = B - \frac{Bs(Bs)^T}{s^T Bs} + \frac{yy^T}{y^T s} \quad .$$

Dennis and Moré give a theoretical discussion of these two updating formulas in [12]. The practical comparisons that Powell makes between these two methods in [30] indicates that the BFGS can have much better properties in practice.

All of this points to the possibilities of using any one of (1)-(6) with either one of (a) or (b). We have been testing the use of $y_D$ with a version of the BFGS update.

## 3. Constrained Nonlinear Optimization

Here we will give a quick review of the exact penalty method for the nonlinear programming problem due to Coleman and Conn [6,7,9]. Our discussion will 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 [29], cited below, that provided the motivation for their work. Since we are merely surveying results at a low level of detail, we can afford the added complication.

In the first parts of this section we will outline the basic results and the "pure" algorithm, by which we mean one which demands that any $c_r(x)$ be exactly zero before it is considered to be an *active constraint*. We will conclude by outlining the modifications that are derived from the introduction of activity tolerances.

### 3.1. The Exact Penalty Method

If we ignore the least-squares structure and add constraints, the problems becomes

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

Both $\phi$ and the $c$'s are functions from $\mathbf{R}^n$ to $\mathbf{R}^1$, and they are usually assumed to be twice continuously differentiable.

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

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

where the *penalty parameter* $\mu$ is a positive number. Early mentions of this penalty function are to be found in Zangwill [37], Fiacco and McCormack [15], and Pietrzykowski [29]. The result to be found in Pietrzykowski guarantees that $\Psi$ is an *exact* penalty function. That is, under proper conditions and if $\mu$ is small enough, every isolated local minimum of $\Psi$ is also an isolated local minimum of (3.1.1). More precisely:

*Result 3.1.1*

> Let $x^*$ denote an isolated minimum to (3.1.1) and assume that $\phi$, $c_i$, $i=1, \ldots, l+m$ are continuously differentiable in a neighborhood of $x^*$. Assume that the gradients of the active constraints at $x^*$ are linearly independent. Then there exists a real number $\mu^*>0$ such that for each $\mu$, $0 < \mu \leq \mu^*$, there is an isolated local minimum $x(\mu)$ of $\Psi(x,\mu)$ for which $x(\mu) = x^*$.

Other exact-penalty approaches exist; e.g. see [19] for a broader discussion.

Coleman and Conn [6,7,9] take the result of Pietrzykowski as a starting point and propose an algorithm for minimizing $\Psi$ for a fixed value of $\mu$. Under certain conditions (most notably: (1) two continuous derivatives are required, (2) the points produced by the algorithm lie in a compact set, (3) the gradients of all active constraints at $x$ are linearly independent for all $x$, (4) and line search conditions, conditions of positive-definiteness, and second-order sufficiency conditions hold), their method will converge globally to an isolated minimizer of $\Psi$, and the ultimate rate of convergence will be two-step superlinear.

The solution approach given by Coleman and Conn for (3.1.1), which uses their minimization algorithm for $\Psi$ as the basic tool, is:

```
{choose μ>0};
feasible := false;
failure := false;
while ({μ is large enough} and not feasible and not failure) do
      {reduce μ};
      {minimize Ψ(x,μ) with respect to x};
      if (not failure) then
            {test feasibility}
      endif
endwhile;
```

It is expected that the "minimize" step of the above will set the "failure" flag if it is unable to succeed. The "test feasibility" step of the algorithm is expected to set the logical variable "feasible" appropriately. The proof of success, of course, comes from ending the **while** loop with $\mu$ large enough, "feasible" equal to **true**, and "failure" equal to **false**.

The case in which problem (3.1.1) is an infeasible one is assessed by detecting infeasibility for a sequence of values of $\mu$ tending to zero. On a computer, using finite-precision arithmetic, an infinite sequence is not required to arrive at a conclusion. It is enough to reduce $\mu$ 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 $\mu$. They constitute difficult situations, in general, for nonlinear programming algorithms; see Coleman and Conn [5].

Situations of unboundedness in the general nonlinear programming problem 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 $\Psi$ is carried out using several alternative step directions:

(i) the *global horizontal direction*;

(ii) the *dropping direction*;

(iii) the *Newton direction*;

and the Newton direction is composed of two components:

(iii′) the *asymptotic horizontal direction*;

(iii″) the *vertical direction*.

These directions are described in terms of the following index sets:

the set of *active-constraint indices*

$$\mathbf{AC}(x) \;=\; \{r : c_r(x)=0 \textbf{ and } 1\le r\le l+m\} \;,$$

the set of *violated equality-constraint indices*

$$\mathbf{VE}(x) \;=\; \{i : |c_i(x)|>0 \textbf{ and } 1\le i\le l\} \;,$$

and the set of *violated inequality-constraint indices*

$$\mathbf{VI}(x) \;=\; \{j : c_j(x)<0 \textbf{ and } l+1\le i\le l+m\} \;.$$

We will use $t$ to denote the number of elements in $\mathbf{AC}(x)$, and our own assumption of simplicity for ease in presentation will be that $0<t<n$. The cases $t=0$ and $t=n$ require some separate attention in implementing the Coleman-Conn algorithm, but the details are not interesting. The case of *degeneracy*, $t>n$, is obviously ruled out of the presentation. For a discussion of some techniques useful in handling this case, see Busovaca [3].

Note that $\Psi(x,\mu)$ can be written in the form

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

The following vectors and matrices, defined in terms of these index sets, play a role in the algorithm: the *active constraint matrix*

$$A(x) \;=\; \left[\nabla c_{r_1}(x),\ldots,\nabla c_{r_t}(x)\right] \;,$$

where $r_1,\ldots,r_t$ are the members of $\mathbf{AC}(x)$, the *effective gradient* of $\Psi$

$$\nabla\Psi(x,\mu) \;=\; \mu\nabla\phi(x) + \sum_{i\in\mathbf{VE}(x)} \operatorname{sgn}(c_i(x))\nabla c_i(x) - \sum_{j\in\mathbf{VI}(x)} \nabla c_j(x) \;,$$

and the *effective Hessian* of $\Psi$

$$\nabla^2\Psi(x,\mu) \;=\; \mu\nabla^2\phi(x) + \sum_{i\in\mathbf{VE}(x)} \operatorname{sgn}(c_i(x))\nabla^2 c_i(x) - \sum_{j\in\mathbf{VI}(x)} \nabla^2 c_j(x) \;.$$

A necessary condition for $x$ to be an isolated local minimizer for $\Psi$ is the following:

*Result 3.1.2*

Assume that $\phi$, $c_i$, $i=1, \ldots, l+m$ are continuously differentiable and that $\{\nabla c_r(x): r \in \mathbf{AC}(x)\}$ is a linearly independent set, then necessary conditions for $x$ to be a local minimizer of $\Psi(x,\mu)$ are that there exist *multipliers*, $\lambda_r$ for $r \in \mathbf{AC}(x)$,

(a) such that

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

(b) and such that

$$-1 \leq \lambda_r \leq 1, \ r \in \mathbf{AC}(x) \cap \{1,...,l\} \tag{3.1.4}$$

$$0 \leq \lambda_i \leq 1, \ r \in \mathbf{AC}(x) \cap \{l+1,...,l+m\} \ . \tag{3.1.5}$$

A point, $x$, for which (a) above is satisfied is a *stationary point* of $\Psi$. An optimal point, then, is a stationary point that satisfies (b).

### 3.1.1. The Multiplier Estimates

The estimates of the numbers $\lambda_r$ 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 $\lambda_r$ are taken to be the least-squares solution to

$$A(x)\lambda = \nabla\Psi(x,\mu) \ . \tag{3.1.1.6}$$

In practice the $QR$ decomposition of $A(x)$ is used to solve the least-squares problem:

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

Since $t$ is the number of columns in $A(x)$, $Z$ is an $n \times (n-t)$ matrix satisfying

$$A(x)^T Z = 0$$

and

$$Z^T Z = I_{(n-t)\times(n-t)}.$$

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

$$Z Z^T \nabla\Psi(x,\mu)$$

is deemed "small enough" according to this tolerance. This will be discussed further in section 5 below.

### 3.1.2. The Global Horizontal Direction

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

$$\underset{h}{\text{minimize}} \ \nabla\Psi(x,\mu)^T h + \tfrac{1}{2} h^T \nabla^2\Psi(x,\mu) h \tag{3.1.2.7}$$

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

Using the $QR$ decomposition of $A(x)$, if we set $h_G = Zw$ for some $w \in \mathbf{R}^n$, then (3.1.2.7) implies that $w$ is to be found by solving

$$(Z^T[\nabla^2\Psi(x,\mu)]Z)w = -Z^T\nabla\Psi(x,\mu) \ . \tag{3.1.2.8}$$

The direction $h_G$ is a descent direction for $\Psi$ at $x$ provided that $(Z^T[\nabla^2\Psi(x,\mu)]Z)$ is positive definite and $Z^T\nabla\Psi(x,\mu) \neq 0$.

Exact second derivatives are replaced by approximations in the algorithm. Thus, (3.1.2.8) is actually solved as

$$\hat{H}w \;=\; (Z^T H Z)w \;=\; -Z^T \nabla\Psi(x,\mu) \;\;.$$

In the material presented in [6,7,9] it is stressed that $\hat{H} = Z^T H Z$ should be a single matrix, a positive-definite approximation to $Z^T[\nabla^2\Psi(x,\mu)]Z$ that is updated according to a quasi-Newton formula. This avoids some difficulties with the algorithm in the event that $Z^T[\nabla^2\Psi(x,\mu)]Z$ is positive definite in a neighborhood of a stationary point whereas $\nabla^2\Psi(x,\mu)$ is indefinite there. The alternative, updating $H$ alone as a quasi-Newton approximation to $\nabla^2\Psi(x,\mu)$ and then explicitly forming $Z^T H Z$ whenever needed, is simpler to implement but theoretically less robust.

In this particular matter we opted for simplicity in the implementation we used for testing. We update $H$ rather than $\hat{H}$. Various suggestions have been made regarding projected Hessian updates, see [4,9,28], the choice of which is the best to use is not yet clear, and we have left an investigation of such matters to the next phase of our studies.

### 3.1.3. The Asymptotic Horizontal Direction

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

$$\underset{h}{\text{minimize}} \;\; \nabla\Psi(x,\mu)^T h + \tfrac{1}{2} h^T [\nabla^2\Psi(x,\mu) - \sum_{r\in\mathbf{AC}(x)} \lambda_r \nabla^2 c_r(x)] h \qquad (3.1.3.9)$$

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

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

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

or alternatively, $H$ should be a positive-definite approximation to

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

### 3.1.4. The Vertical Step Direction

At $x + h_A$, the constraints of $\mathbf{AC}(x)$ may no longer be active. By means of a vertical step, $v$, the constraints of $\mathbf{AC}(x)$ are brought more closely to a value of zero. The vertical step is based upon a Newton step toward the solution of the nonlinear system $c_{\mathbf{AC}(x)}(x+h_A) = 0$, where $c_{\mathbf{AC}(x)}$ is the vector of constraint functions, ordered in accord with the columns of $A(x)$. The vertical step is the solution to the system

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

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

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

$$\text{set} \;\; v \;=\; W u \;\;.$$

### 3.1.5. The Dropping Direction

The direction that releases the constraint from activity, locally and to first order, whose gradient appears as the $r$-th column in the matrix $A(x)$ is the direction $d$ that satisfies the system of equations

$$A(x)^T d \;=\; \sigma_r e_r \;\;, \qquad (3.1.5.10)$$

where $e_r$ is the $r$-th unit vector, and

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

### 3.1.6. Direction Choice Strategy

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

(1) When $Z Z^T \nabla \Psi(x,\mu)$ is not "small enough" as indicated by the stationarity tolerance $\tau$, then

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

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

(2) When $Z Z^T \nabla \Psi(x,\mu)$ is "small enough" as indicated by the stationarity tolerance $\tau$, the multipliers $\lambda_r$, $r \in \mathbf{AC}(x)$, are estimated.

(a) If (3.1.4) and (3.1.5) are not satisfied, an index $r \in \mathbf{AC}(x)$ 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
$\bar{x} \leftarrow x + h_A + v$ .

### 3.1.7. The Activity Tolerance

The stationarity tolerance has already been mentioned. It serves the purpose of preventing the estimation of values of the multipliers according to (3.1.1.6) until there is a reasonable chance that the estimated values will indicate whether the inequalities of (3.1.4) and (3.1.5) are satisfied. If this tolerance is not used, or if it is set too large, the dropping step can be chosen inappropriately, causing indices to be shifted in and out of $\mathbf{AC}(x)$ repeatedly. This zig-zagging phenomenon will also arise if the directions $h_G$, $h_A$, $d$, and $v$ are determined using the true penalty function $\Psi$; that is, using the index sets $\mathbf{AC}(x)$, $\mathbf{VE}(x)$, and $\mathbf{VI}(x)$. In this case, the zig-zagging will be a result of the fact that $c_r(x)$ must be precisely zero before $r$ can be included in the index set $\mathbf{AC}(x)$ and that any subsequent step will usually cause $c_r(x)$ to depart from zero again. The algorithm should consider as active any constraints that are within a tolerance of being zero; that is, it should determine $h_G$, $h_A$, $d$, and $v$ based upon an *$\epsilon$-approximate* version of the penalty function

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

in which $\mathbf{VE}(x)$ and $\mathbf{VI}(x)$ are replaced by

$$\mathbf{VE}_\epsilon(x) = \{i : |c_i(x)| \text{ is "large enough" as indicated by epsilon } \textbf{and } 1 \le i \le l\}$$

and

$$\mathbf{VI}_\epsilon(x) = \{j : c_j(x) \text{ is "negative enough" as indicated by epsilon } \textbf{and } l+1 \le i \le l+m\}$$

respectively, and $\mathbf{AC}(x)$ is replaced by

$$\mathbf{AC}_\epsilon(x) = \{r : |c_r(x)| \text{ is "small enough" as indicated by epsilon } \textbf{and } 1 \le r \le l+m\} ,$$

and all of the discussion of sections 3.1 through 3.1.6 is changed so that $\mathbf{AC}_\epsilon(x)$, $\mathbf{VE}_\epsilon(x)$, and $\mathbf{VI}_\epsilon(x)$ play the roles of $\mathbf{AC}(x)$, $\mathbf{VE}(x)$, and $\mathbf{VI}(x)$ respectively. The quantity $\epsilon$ is the *activity tolerance*, and it is to be positive. We will be more specific about its use in section 5.

The theoretical properties of the algorithm will remain intact, under reasonable conditions, so long as the line search mentioned in section 3.1.6 is designed to reduce the value of the true penalty function, $\Psi(x,\mu)$ rather than $\Psi_\epsilon(x,\mu)$, and so long as the the tolerances $\tau$ and $\epsilon$ are chosen reasonably. Since there is no *a priori* way in which the values of $\tau$ and $\epsilon$ can be known, some positive values are chosen initially, and the algorithm is adjusted to reduce the values of $\tau$ and $\epsilon$ 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), (2,a), or (2,b) given in section 3.1.6. Such cases indicate that $\Psi_\epsilon(x,\mu)$ is not an adequate predictor of the behavior of the true penalty function $\Psi(x,\mu)$, or that the proximity to the nearest stationary point is being misjudged using the stationarity tolerance $\tau$, or both.

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 3.1.6 whenever an appropriate measure of sufficient decrease in the true penalty function is not achieved. Lack of sufficient decrease in steps (2,a) and (2,b) 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.

The revision of $\epsilon$ and $\tau$ are undertaken together in the Coleman and Conn algorithm. It might be hoped that they could be adjusted separately, but the roles that they play are intertwined. The tolerance $\tau$ is used to determine nearness to stationarity so that the trustworthiness and advisability of computing the $\lambda$'s can be judged. The tolerance $\epsilon$ is used to determine the set $\mathbf{AC}_\epsilon(x)$, and this set, in turn, specifies what $\lambda$'s can be computed and what model, $\Psi_\epsilon(x,\mu)$, of the penalty function will be used to predict a descent direction. Insufficient decrease can occur if the choice about whether to compute the $\lambda$'s is wrongly made, because $x$ is too far from the nearest stationary point for the values of the $\lambda$'s to be accurate guides, or if the wrong $\lambda$'s are computed even though $x$ is close enough, because $\epsilon$ is too large and $\mathbf{AC}_\epsilon(x) \neq \mathbf{AC}(x)$, or if $\Psi_\epsilon(x,\mu)$ does not reliably predict the decrease of $\Psi(x,\mu)$, also because $\epsilon$ is too large. It is not known how to assign the blame for insufficient decrease to $\tau$ or to $\epsilon$ alone. It is known, however, that the insufficiency can be overcome, under reasonable assumptions, by reducing both together.

### 3.1.8. Algorithm Outline

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

```
{choose ε>0 and τ>0};
{choose H};
optimal := false;
while ( not optimal and {ε is large enough} and {τ is large enough}) do
    adequate := true;
    {determine AC_ε(x), VE_ε(x), and VI_ε(x)};
    {determine A(x), Q=[WZ], and R correspondingly};
    if ({Z Z^T ∇Ψ_ε(x,μ), tested against τ, is not small enough}) then
        {determine h_G};
        {determine α from a line search on Ψ(x,μ)};
        if ({sufficient decrease is indicated}) then
            x := x + α h_G;
            {update H ≈ ∇²Ψ_ε(x,μ)}
        else
            adequate := false
        endif
    else
        {determine the λ's};
        {check whether λ_r exists violating (3.1.4) or (3.1.5)};
        if ({λ_r exists}) then
            {determine d};
            {determine α from line search on Ψ(x,μ)};
            if ({sufficient decrease is indicated}) then
                x := x + α h_G;
                {update H ≈ ∇²Ψ_ε(x,μ)}
            else
                adequate := false
            endif
```

**else**
{determine $h_A$};
{determine $v$ to solve $A(x)^T v = -c_{\mathbf{AC}_\epsilon(x)}(x+h_A)$};
$p := h_A + v$;
{check sufficient decrease on $\Psi(x,\mu)$};
**if** ({sufficient decrease is indicated}) **then**
$x := x + p$;
{update $H \approx \nabla^2\Psi_\epsilon(x,\mu) - \sum_{r\in\mathbf{AC}_\epsilon(x)} \lambda_r \nabla^2 c_r(x)$};
{test optimality}
**else**
adequate := **false**
**endif**
**endif**
**endif**;
**if** (**not** adequate) **then**
**repeat**
{reduce $\epsilon$};
{reduce $\tau$}
**until** ({$\mathbf{AC}_\epsilon(x)$ changes}
**or** {$ZZ^T\nabla\Psi_\epsilon$ becomes large tested against $\tau$}
**or** {$\epsilon$ becomes too small}
**or** {$\tau$ becomes too small})
**endif**;
failure := **not** optimal
**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 $\epsilon$ or $\tau$ become too small and on whether sufficient decrease is attained using the directions $h_G$ and $d$. Such evidences of failure must be detected if the algorithm is to "die gracefully" when applied to a problem not meeting the assumptions made in [6,7].

## 4. Accommodating the Least-Squares Structure

For CNLLS, the minimization problem becomes

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \phi(x) = \tfrac{1}{2}F(x)^T F(x) = \tfrac{1}{2}\sum_{\delta=1}^{k}[f_\delta(x)]^2 \\ & = \tfrac{1}{2}\|F(x)\|^2 \\ \text{such that} \quad & c_i(x) = 0 \qquad i = 1,\ldots,l \\ & c_j(x) \geq 0 \qquad j = l+1,\ldots,l+m \quad . \end{aligned} \tag{4.1}$$

The structure of the objective function is to be reflected in the "choose $H$" and "update $H$" steps of the algorithm. In all phases of the algorithm

$$H \approx \mu J(x)^T J(x) + S(x) \quad .$$

In the *global phase*; i. e., for the steps determined by a line search using the directions $h_G$ or $d$,

$$S(x) = \mu\sum_{\delta=1}^{k} f_\delta(x)\nabla^2 f_\delta(x) + \sum_{i\in\mathbf{VE}_\epsilon(x)} \mathrm{sgn}(c_i(x))\nabla^2 c_i(x) - \sum_{j\in\mathbf{VI}_\epsilon(x)} \nabla^2 c_j(x) \quad ,$$

and in the *asymptotic phase*; i. e., for the steps determined by $h_A + v$,

$$S(x) = \mu \sum_{\delta=1}^{k} f_\delta(x)\nabla^2 f_\delta(x) + \sum_{i\in \mathbf{VE}_\epsilon(x)} \mathrm{sgn}(c_i(x))\nabla^2 c_i(x) - \sum_{j\in \mathbf{VI}_\epsilon(x)} \nabla^2 c_j(x) - \sum_{r\in \mathbf{AC}_\epsilon(x)} \lambda_r \nabla^2 c_r(x) \ .$$

This means that $H$ can be assumed to have the structure

$$\mu\, J(x)^T J(x) + B$$

where

$$B \approx S(x) \ ,$$

and we only need to choose $B$ and update it according to a quasi-Newton formula, varying the vectors $s$ and $y$ of (2.3.7) appropriately depending upon the phase and step taken by the algorithm. In particular, the formula for $y_D$ as given in section 2 may be extended in the obvious way. Under the simplifying assumption that the values of the multipliers $\lambda_r$ may be regarded as constant over the interval between $\bar{x}$ and $x$, the BFGS updating formula that uses this $y_D$ results in the following version of the "update $H$" steps:

$y := \mu[J(\bar{x})-J(x)]^T F(\bar{x}) + \sum_{i\in \mathbf{VE}_\epsilon(x)} \mathrm{sgn}(c_i(\bar{x}))[\nabla c_i(\bar{x})-\nabla c_i(x)] - \sum_{j\in \mathbf{VI}_\epsilon(x)} [\nabla c_j(\bar{x})-\nabla c_j(x)]$

**if** ({Newton step}) **then**

$y := y - \sum_{r\in \mathbf{AC}_\epsilon(x)} \lambda_r [\nabla c_r(\bar{x})-\nabla c_r(x)]$;

$s := \bar{x} - x$;

{use $y$ and $s$ in the BFGS update of $B$};

$H := \mu\, J(\bar{x})^T J(\bar{x}) + B$;

## 5. Software Issues

Much 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 [21]. Specific choices that we have made in implementing the algorithm are listed here to aid the reader in interpreting the computational results outlined in the next section. The following quantities were involved:

(1) $Z Z^T \nabla \Psi_\epsilon(x,\mu)$

(2) $c_r(x)$, for $r = 1,...,l+m$

(3) $\lambda_r$, for $r \in \mathbf{AC}_\epsilon(x)$

(4) $\Psi(\bar{x},\mu) - \Psi(x,\mu)$ .

The projected effective gradient, $Z Z^T \nabla \Psi_\epsilon(x,\mu)$, 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 $Z$ are orthonormal, it is sufficient to test $\| Z^T \nabla \Psi_\epsilon(x,\mu) \|$ for smallness. Nearness to stationarity is to be determined relative to the tolerance, $\tau$, and acceptability for termination is determined relative to a much smaller tolerance, $\theta$, which must be defined by the user. The test for stationarity takes the form of a relative-magnitude test

$$\| Z^T \nabla \Psi_\epsilon(x,\mu) \| \ \leq \ \tau \ \mathrm{reference}\{ \| \nabla \Psi_\epsilon(x,\mu) \| \} \ , \tag{5.1}$$

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 straightforward relative test in which

$$\mathrm{reference}\{ \| \nabla \Psi_\epsilon(x,\mu) \| \} \ = \ \| \nabla \Psi_\epsilon(x,\mu) \| \ .$$

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. We are using

$$\mathrm{reference}\{ \| \nabla \Psi_\epsilon(x,\mu) \| \} \ = \ \max(1, \| \nabla \Psi_\epsilon(x,\mu) \|) \ .$$

This avoids underflow, or a too stringent test, when the value of the norm becomes close to zero. The tolerance $\theta$, of course, replaces $\tau$ in (5.1) when the tests for convergence are being made.

By the same token, activity is determined by

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

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

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

We are using

$$\text{reference}\{c_r(x)\} = \max(1, \frac{\|F(x)\| + \sum_{r=1}^{l+m} |c_r(x)|}{l+m+1}) \ ,$$

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, $\gamma$, is used in place of $\epsilon$.

The values of 1, $10^{-1}$, and $10^{-2}$ chosen initially for $\mu$, $\epsilon$, and $\tau$, respectively, have served well. They are reduced, respectively, by dividing by 8, 2, and 2. We have been using $10^{-6}$ for $\gamma$ and $10^{-4}$ for $\theta$ for all of the problems reported on below.

The value of $\lambda_r$ 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 $1 \leq r \leq l$ or $l+1 \leq r \leq l+m$, respectively. It is advisable to recognize three cases, in fact, whether $\lambda_r$ 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 $\mu$ is regarded as too small when

$$\mu \|F(x)\| \leq \textit{macheps} \max(1, \frac{\|F(x)\| + \sum_{r=1}^{l+m} |c_r(x)|}{l+m+1}) \ ,$$

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

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

and $\tau$ is regarded to be too small when

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

Item (4) in the list of quantities above 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

$$\Psi(\bar{x},\mu) - \Psi(x,\mu) \;\leq\; -\kappa \text{ reference}\{\, \| Z^T \nabla \Psi_\epsilon(x,\mu) \|_2^2 + \sum_{r \in \mathbf{AC}_\epsilon(x)} |c_r(x)| \,\} \;,$$

where the reference value used is the maximum of 1 and the value of the formula within braces above. The tolerance $\kappa$ 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 [27], 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 simply add a random perturbation, of the order of magnitude of $\epsilon\,|c_r(x)|$ to $c_r(x)$ in order to resolve degeneracy, and 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.

For the purpose of simplicity we chose to carry out quasi-Newton updates on a full, $n \times n$ matrix and then to use $Z$ to explicitly project. This means that the occurrence of indefinite Hessians, in certain circumstances, could prove fatal to the implementation we use for our testing, whereas a more carefully crafted code, designed to update approximated projected Hessians, might succeed. We monitored all of the failures reported in the next section, and we point out which of them are associated with the occurrence of an indefinite or singular Hessian.

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 [4,8,22]. The general result has been that the matrices $W(x)$ and $Z(x)$ derived from

$$A(x) \;=\; \Big[\, W(x)\, Z(x) \,\Big] \begin{bmatrix} R(x) \\ 0 \end{bmatrix}$$

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 $Z(x)$. This is consistent with the evidence others, e. g. Coleman, have gathered to the effect that the continuity of $Z(x)$ is more of a theoretical concern than a practical one.

When any of the denominators in the BFGS formula becomes exactly zero, we avoid division by zero by ignoring the corresponding term in the update. This action is derived from consideration of the case of linear constraints.

The numerical positive definiteness of the matrix

$$Z^T (J(x)^T J(x) + B) Z$$

is enforced by using the modified Cholesky factorization described on page 111 of [21] during the process of solving (3.1.2.8).

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

$$\begin{aligned}
&\| Z^T \nabla \Psi_\epsilon(x,\mu) \| \;\leq\; \theta \text{ reference}\{\, \| \nabla \Psi_\epsilon(x,\mu) \| \,\} \;, \\
&-1+\theta \;<\; \lambda_r \;<\; +1-\theta \text{ for all } r \in \mathbf{AC}_\epsilon(x) \text{ and } 1 \leq r \leq l \;, \\
&\theta \;<\; \lambda_r \;<\; +1-\theta \text{ for all } r \in \mathbf{AC}_\epsilon(x) \text{ and } l+1 \leq r \leq l+m \;, \\
&|\Psi_\epsilon(\bar{x},\mu) - \Psi_\epsilon(x,\mu)| \;\leq\; \theta \text{ reference}\{\, |\Psi_\epsilon(\bar{x},\mu)| \,\} \;, \\
&\|\bar{x} - x\| \;\leq\; \gamma \text{ reference}\{\, \|\bar{x}\| \,\} \;,
\end{aligned}$$

In this regard, $\gamma$ is being used in the spirit of $\tau_F$ and $\theta$ is being used in the spirit of $\sqrt{\tau_F}$ in the notation of chapter 8 of [21], 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 fails to be satisfied, but notice should be given to the user

whenever this happens.

## 6. Computational Results

Thirty CNLLS problems were taken from Hock and Schittkowski [32]:

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.

The number of variables in these problems varies from 1 to 5, and the number of constraints varies from 1 to 13.

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 [27] using a code provided by Michael Overton.

Problems

6, 23, 26, 27, 46, 60, and 79

have all failed to converge because they encountered a rank-deficient approximate Hessian.

Problems

13, 15, and 16

have all converged, at an ultimately two-step superlinear rate, to an optimizer at which one or more multipliers were at a critical value (0 or +1 for an inequality constraint, −1 or +1 for an equality constraint).

All other problems have been solved successfully with an ultimate rate of convergence that was two-step superlinear.

## 7. Acknowledgements

We are grateful for the help 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 suggestions.

## 8. References

1. P. T. Boggs and J. E. Dennis, Jr., A Stability Analysis for Perturbed Nonlinear Iterative Methods, *Math. Comp.* **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. S. Busovaca, *Handling Degeneracy in a Nonlinear L-1 Algorithm,* PhD Thesis, Computer Science Department, University of Waterloo, Waterloo, Ontario N2L 3G1, 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 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).

9. 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).

10. W.C. Davidon, Variable Metric Method for Minimization, ANL-5990 Rev., Argonne National Laboratory (1959).

11. 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).

12. J. E. Dennis, Jr. and J. J. Moré, Quasi-Newton Methods: Motivation and Theory, *SIAM Review* **19** pp. 46-89 (1977).

13. J. E. Dennis, Jr., Techniques for Nonlinear Least Squares and Robust Regression, *Commun. Statist.-Simula. Comput. B* **7**(4) pp. 345-359 (1978).

14. 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).

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

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

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

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

19. 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).

20. 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).

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

22. 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).

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

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

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

26. 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).

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

28. J. Nocedal and M. Overton, Projected Hessian Updating Algorithms for Nonlinearly Constrained Optimization, 95, New York University, 256 Mercer Street, New York, NY (1983).

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

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

31. D. A. Ratkowsky, *Nonlinear Regression Modeling: A Unified Practical Approach,* Marcel Dekker (1983).

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

33. D. F. Shanno, Conditioning of Quasi-Newton Methods for Function Minimization, *Math. Comput.* **24** pp. 647-656 (1970).

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

35. P-Å. 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).

36. P-Å. 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).

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