IMA Journal of Numerical Analysis (1986) **6**, 357–372

On the Accurate Determination of Search Directions for Simple Differentiable Penalty Functions

NICHOLAS IAN MARK GOULD

Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada

We present numerically reliable methods for the calculation of a search direction for use in sequential methods for solving nonlinear programming problems. The methods presented are easy to adapt to such problems as locating directions of negative curvature and linear infinite descent. Encouraging numerical results are included.

1. Introduction

IN this paper, we shall be concerned with the following continuous optimization problems:

NLP: minimize $f(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{R}^n$, subject to

$$c_i(\boldsymbol{x}) = 0 \quad (i \in \mathscr{E}), \qquad c_i(\boldsymbol{x}) \geqslant 0 \quad (i \in \mathscr{I});$$

SIP: minimize $f(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{R}^n$, subject to

$$\phi_i(\boldsymbol{x}, \boldsymbol{t}) \geqslant 0 \quad \text{for all } \boldsymbol{t} \in \mathscr{T}_i \quad (i \in \mathscr{I}), \qquad \mathscr{T}_i \subseteq \mathbb{R}^{p_i} \text{ closed, bounded } (i \in \mathscr{I}).$$

We shall assume that each problem function has sufficiently many continuous derivatives for any implicit assumptions that we make to hold. One of the most successful tools for solving problems of the form NLP and SIP is the penalty function. A penalty-function approach replaces the relevant problem by a suitably weighted combination of the objective function $f(\boldsymbol{x})$ and functions representing violations of constraints. This weighted combination is known as a penalty function. The unconstrained penalty function is normally minimized for a particular choice of the weighting and the weighting subsequently adjusted. The rationale behind such methods is the existence of powerful theoretical results which indicate how the weighting should be adjusted and when the minimizer of the penalty function is likely to converge to that of the relevant original problem (see Fiacco & McCormick, 1968; Pietrzykowski, 1970).

Early penalty functions for NLP include the quadratic loss function (form mixed equality and inequality constraints) and the barrier functions (for inequality constraints):

$$\text{Quadratic penalty function: } p_2(\boldsymbol{x}, \mu) = f(\boldsymbol{x}) + \frac{1}{2\mu} \sum_{i \in \mathscr{E}} c_i(\boldsymbol{x})^2 + \frac{1}{2\mu} \sum_{i \in \mathscr{I}} c_i(\boldsymbol{x})_-^2;$$

$$\text{Barrier function: } p_B(\boldsymbol{x}, \mu) = f(\boldsymbol{x}) + \mu \sum_{i \in \mathscr{I}} \varphi[c_i(\boldsymbol{x})];$$

where $c_i(\boldsymbol{x})_-$ indicates the lesser of $c_i(\boldsymbol{x})$ and zero and μ is a positive scalar parameter. The function $\varphi : \mathbb{R} \to \mathbb{R}$ included in the definition of the barrier function is any function such that $\varphi'(u) < 0$ and $\varphi''(u) > 0$ for $u > 0$ and such that $\lim_{u \to 0^+} \varphi(u) = +\infty$. (For instance, the logarithmic barrier function has $\varphi(u) = -\log_e u$ and the reciprocal barrier function chooses $\varphi(u) = u^{-s}$ for some $s > 0$).

In the same class of functions is the limiting penalty function $p_L(\boldsymbol{x}, \mu)$ for semi-infinite programming:

$$\textit{Limiting penalty function}: \; p_L(\boldsymbol{x}, \mu) = f(\boldsymbol{x}) - \frac{1}{\mu} \sum_{i \in \mathcal{S}} \int_{\mathcal{T}_i} \phi_i(\boldsymbol{x}, t)_- \mathrm{d}t.$$

All of these functions have the following properties.

(i) For a fixed value of the penalty parameter μ, they are continuously differentiable within their domains of definition. (The functions p_2 and, under suitable assumptions (Conn & Gould, 1985), p_L are continuously differentiable at all points for which the problem functions are. The barrier function p_B is continuously differentiable at all points strictly within the feasible region).

(ii) Under fairly mild conditions it can be shown that any local solution x^* to either the problem NLP or SIP is such that $\lim_{\mu \to 0^+} \boldsymbol{x}(\mu) = \boldsymbol{x}^*$ where $\boldsymbol{x}(\mu)$ is a local minimizer of the relevant penalty function (Pietrzykowski, 1970). In most cases, it is necessary that μ tends to zero. This has unfortunate repercussions as:

(iii) The second derivatives of the penalty functions are locally differentiable in a neighbourhood of $\boldsymbol{x}(\mu)$. However as $\mu \to 0^+$, the second derivative (Hessian) matrices become increasingly ill conditioned (see Lootsma, 1969; Murray, 1971).

This third property was thought to be catastropic as a simple-minded application of Newton's method to finding a stationary point of the penalty function would require the solution of many systems of linear equations, each of which has the Hessian matrix of the penalty function as its coefficient matrix. The ill conditioning of the matrix proved to make accurate determination of $\boldsymbol{x}(\mu)$ exceedingly difficult for small values of μ. During the 1970's the simple differentiable penalty functions fell from favour, with the more complicated exact penalty functions and augmented-Lagrangian functions taking their place (see for example Gill, Murray & Wright, 1981).

During the past ten years, there have been a number of attempts to overcome the difficulties discussed above. Gerencser (1974), Biggs (1975), and Broyden & Attia (1983) proposed methods which sought to isolate those terms in the penalty-function Hessian which caused the ill conditioning by a suitable transformation of variables. Indeed Broyden and Attia report considerable success with a method in which a form of iterative refinement is used to correct an initial attempt at solving Newton's equations for a stationary point of $p_2(\boldsymbol{x}, \mu)$. Likewise, Biggs has avoided the ill conditioning by isolating the dominant terms in $\nabla_{xx} p_2(\boldsymbol{x}, \mu)$ and by maintaining a positive definite approximation to the remaining second-derivative terms. Murray & Wright (1978) seek to follow the

trajectory of $x(\mu)$ as $\mu \to 0$ by solving a 'well conditioned' approximation to the Newton equations. Almost all of these methods rely on a transformation of the coordinate axes and have much in common with the so-called null-space methods which are often used to solve equality-constrained quadratic programs (see e.g. Gill, Murray, & Wright (1981)).

In this paper, we present several alternative methods for solving the system of equations which arises from an application of Newton's method to the solution of $\nabla_x p[x(\mu), \mu] = 0$. (Here, $p(x, \mu)$ is any of the previously mentioned penalty functions.) Given an estimate $x^{(k)}$ of $x(\mu)$, Newton's method gives a suitable correction $p^{(k)}$ as the solution of

$$\nabla_{xx} p(x^{(k)}, \mu) p^{(k)} = -\nabla_x p(x^{(k)}, \mu). \tag{1.1}$$

The methods we suggest do not suffer from the ill conditioning present in a direct solution of (1.1).

If we view Newton's method as finding a search direction vector $p^{(k)}$ so that $p(x^{(k)} + p^{(k)}, \mu)$ is sufficiently less than $p(x^{(k)}, \mu)$, we have no reason to expect (1.1) will give rise to a descent direction for $p(x, \mu)$ if $x^{(k)}$ is far from $x(\mu)$. Under these circumstances, it is often more appropriate to calculate a 'direction of negative curvature' for $p(x, \mu)$ if such a direction exists. Our methods easily cope with such a possibility. Indeed, the technique used in this case is a minor modification of those methods discussed in Conn & Gould (1984). For simplicity of presentation, we shall focus our attention on $p_2(x, \mu)$ rather than the other penalty functions. The techniques discussed extend trivially to $p_B(x, \mu)$ (see Section 6). The extension to $p_L(x, \mu)$ is slightly more complicated because of the nature of the quadrature required to evaluate the penalty function. An algorithm for SIP based on some ideas here will be featured in a separate paper (Conn & Gould, 1985).

In Section 2, we describe our fundamental philosophy. In Section 3, we consider how to calculate directions of negative curvature for $p_2(x, \mu)$. In Section 4, we present an effective algorithm for solving NLP and give some numerical results in Section 5. Further extensions are considered in Section 6.

2. Solving Newton's equations

If we consider $p_2(x, \mu)$ and define the index set $\mathcal{V}(x) = \mathcal{E} \cup \{i \in \mathcal{I} : c_i(x) \leq 0\}$, we obtain

$$\nabla_x p_2(x, \mu) = g(x) - A(x)^T \lambda(x, \mu)$$

and

$$\nabla_{xx} p_2(x, \mu) = G(x) - \sum_{i \in \mathcal{V}(x)} \lambda_i(x, \mu) G_i(x) + \frac{1}{\mu} A(x)^T A(x),$$

where $g(x)$ and $G(x)$ are the first and second derivatives of $f(x)$, $c(x)$ is the vector whose components are the $c_i(x)$ with $i \in \mathcal{V}(x)$, $A(x)$ is the Jacobian matrix of $c(x)$, $G_i(x)$ is the Hessian matrix of $c_i(x)$, and the vector $\lambda(x, \mu)$ has

components

$$\lambda_i(x, \mu) = -c_i(x)/\mu \qquad (i \in \mathcal{V}(x)).$$

Under mild conditions as $\mu \to 0_+$, $x(\mu) \to x^*$ and $\lambda_i[x(\mu), \mu] \to \lambda_i^*$ where λ_i^* is a Lagrange multiplier associated with the solution x^* of NLP (see Fiacco & McCormick, 1968; Wright, 1976). Therefore, as $\mu \to 0$, it is clear that $\nabla_{xx} p_2[x(\mu), \mu]$ is dominated by the term $\mu^{-1} A[x(\mu)]^T A[x(\mu)]$ (the remaining terms converging to the Hessian matrix of the Lagrangian function $\ell(x, \lambda) = f(x) - \lambda^T c(x)$ at $x = x^*$, $\lambda = \lambda^*$). Thus, in most cases $\nabla_{xx} p_2[x(\mu), \mu]$ will become increasingly ill conditioned as μ decreases. (However if $\mathcal{V}(x)$ is empty or if $|\mathcal{V}(x)| \geq n$ and $A(x)$ is full rank, the ill conditioning will not occur.) This ill conditioning will also be reflected in $\nabla_{xx} p_2(x^{(k)}, \mu)$ where $x^{(k)}$ is an approximation to $x(\mu)$. Hence, a straightforward application of Newton's method (1.1) is likely to encounter severe numerical difficulties in calculating a search direction $p^{(k)}$ for small values of μ.

Let us define

$$\bar{G}(x, \mu) = G(x) - \sum_{i \in \mathcal{V}(x)} \lambda_i(x, \mu) G_i(x).$$

The tacit assumption here is that $\bar{G}(x, \mu)$ will be well behaved for points encountered in our minimization process as $\mu \to 0$. This assumption is reasonable provided we start our process with a largish value of μ (say $\mu = 10^{-1}$ for well scaled problems (see Section 5)). Our proposal is then to proceed in the fashion of the SUMT method of Fiacco & McCormick (1968) but to ensure that the search direction $p^{(k)}$ from (1.1) is accurately determined. Let $A^{(k)} = A(x^{(k)})$, $\bar{G}^{(k)} = \bar{G}(x^{(k)}, \mu)$, $g^{(k)} = g(x^{(k)})$, $\lambda^{(k)} = \lambda(x^{(k)}, \mu)$, and $c^{(k)} = c(x^{(k)})$. Consider the Newton equations (1.1), at $x^{(k)}$,

$$(\bar{G}^{(k)} + \mu^{-1} A^{(k)T} A^{(k)}) p^{(k)} = -(g^{(k)} + \mu^{-1} A^{(k)T} c^{(k)}). \tag{2.1}$$

Define the vector $r^{(k)} = \mu^{-1}(A^{(k)} p^{(k)} + c^{(k)})$. Then (2.1) may be rewritten as

$$\bar{G}^{(k)} p^{(k)} + A^{(k)T} r^{(k)} = -g^{(k)},$$
$$A^{(k)} p^{(k)} - \mu r^{(k)} = -c^{(k)},$$

i.e.

$$\begin{bmatrix} \bar{G}^{(k)} & A^{(k)T} \\ A^{(k)} & -\mu I \end{bmatrix} \begin{bmatrix} p^{(k)} \\ r^{(k)} \end{bmatrix} = -\begin{bmatrix} g^{(k)} \\ c^{(k)} \end{bmatrix}. \tag{2.2}$$

As $\mu \to 0_+$, there is no reason for the coefficient matrix of (2.2) to be poorly conditioned; indeed if we denote the Lagrangian function $\ell(x, \lambda) = f(x) - \lambda^T c(x)$ then

$$\lim_{\mu \to 0_+} \begin{bmatrix} \bar{G}[x(\mu), \mu] & A[x(\mu)]^T \\ A[x(\mu)] & -\mu I \end{bmatrix} = \begin{bmatrix} \nabla_{xx} \ell(x^*, \lambda^*) & A(x^*)^T \\ A(x^*) & 0 \end{bmatrix}. \tag{2.3}$$

The matrix in (2.3) is precisely that which reflects the sensitivity of the solution of NLP to small perturbations in the data (see Robinson, 1980). As such, ill conditioning in (2.3) is indicative of inherent ill conditioning in NLP. For small

values of μ and $x^{(k)}$ close to $x(\mu)$, the coefficient matrix of (2.2) is likely to be a small perturbation of (2.3). Therefore the conditioning of (2.3) will normally be reflected in the coefficient matrix of (2.2). In Section 3 we shall discuss properties of the matrix

$$\begin{bmatrix} \bar{G}^{(k)} & A^{(k)\mathsf{T}} \\ A^{(k)} & -\mu I \end{bmatrix} = K^{(k)}. \tag{2.4}$$

At present it is sufficient to notice that it is symmetric and (normally) indefinite. An appropriate way to solve (2.2) would therefore be by way of the Bunch–Parlett–Fletcher–Kaufman Generalized Cholesky factorization (Bunch & Parlett, 1971; Fletcher, 1976; Bunch & Kaufman, 1977). This has additional benefits which will be discussed in Section 3.

We note that the vector $-r^{(k)}$ will converge to the vector λ^* as $\mu \to 0$ and $k \to \infty$. It may be appropriate, therefore, to replace the vector $\lambda(x^{(k)}, \mu) = -\mu^{-1} c(x^{(k)})$, needed in calculating $\bar{G}$, by $-r^{(k-1)}$ as we approach the solution. This should avoid possible numerical inaccuracies associated with the calculation of $c(x^{(k)})/\mu$ as $\mu \to 0$, although we have never had to resort to such a substitution in practice.

We consider (2.2) to belong to the class of Lagrangian methods for solving NLP (see Fletcher, 1981). Indeed, in the limit as $\mu \to 0$, (2.2) gives the Newton–Lagrange method for solving the equations

$$A(x)^{\mathsf{T}} \lambda(x) = g(x), \qquad c(x) = 0.$$

If $\bar{G}^{(k)}$ is non-singular, the simplicity of the structure in (2.2) allows us to write $p^{(k)}$ in terms of $r^{(k)}$ in the first equations of (2.2) and substitute into the second equations to obtain the range-space equations

$$(A^{(k)} \bar{G}^{(k)-1} A^{(k)\mathsf{T}} + \mu I) r^{(k)} = c^{(k)} - A^{(k)} \bar{G}^{(k)-1} g^{(k)} \tag{2.5a}$$

$$\bar{G}^{(k)} p^{(k)} = -g^{(k)} - A^{(k)\mathsf{T}} r^{(k)}. \tag{2.5b}$$

We notice that these equations are very similar to those developed by Biggs (1972, 1975, 1978). Biggs' method replaces $\bar{G}^{(k)}$ in (2.5) by a suitable positive definite approximation. We note, however, that we do not require $\bar{G}^{(k)}$ to be positive definite in the methods we are suggesting here. Equations (2.5) may be appropriate when $|\mathcal{V}(x^{(k)})|$ is small, for then (2.5a) will be a small system of equations. The arguments for using a range-space rather than a Lagrangian method have been made elsewhere (Conn & Gould, 1984; Gill, Murray, & Wright, 1981) and will not be repeated here. Once again (2.5) should be solved using a Generalized Cholesky factorization of the matrices $\bar{G}^{(k)}$ and $A^{(k)} \bar{G}^{(k)-1} A^{(k)\mathsf{T}} + \mu I$ and this yields additional benefits (see Section 3).

If our interest is in solving large-structured nonlinear programs, the equation (2.2) may be preferred, since any sparsity in $\bar{G}^{(k)}$ and $A^{(k)}$ carries over into $K^{(k)}$. Our preferred method of solution of (2.2) would then be by a sparsity-exploiting variant of the Generalized Cholesky factorization (see e.g. Duff & Reid, 1983). In particular, if our problem is partially separable in the sense of Griewank & Toint (1982), the multifrontal elemental approach of Duff & Reid would seem

most appropriate. Moreover, it should be hoped that the data structures needed to solve (2.2) many times would remain more or less fixed.

3. Properties of the matrices and directions of infinite descent

If we wish to use our computed $\boldsymbol{p}^{(k)}$ as a search direction along which to reduce $p_2(\boldsymbol{x}, \mu)$, we need to ensure that $\boldsymbol{p}^{(k)}$ is a descent direction. This is immediate whenever $\nabla_{xx}p_2(\boldsymbol{x}^{(k)}, \mu)$ is positive definite. If $\nabla_{xx}p_2(\boldsymbol{x}^{(k)}, \mu)$ is indefinite, a local quadratic model of $p_2(\boldsymbol{x}, \mu)$ predicts directions of negative curvature along which the quadratic model decreases without bound. This is often the case far from the solution. If $\nabla_{xx}p_2(\boldsymbol{x}^{(k)}, \mu)$ is positive semidefinite but singular (1.1) may not have a solution. In this case there are directions of linear infinite descent along which the quadratic model again decreases without bound. When such directions of linear infinite descent or directions of negative curvature exist it is reasonable to locate one and use it, in place of the $\boldsymbol{p}^{(k)}$ obtained from (1.1), as a search direction. In this section we consider when such directions exist and how we might find one. We need the following formal definitions and elementary results.

DEFINITION. Any vector $\boldsymbol{p}$ such that $\boldsymbol{p}^{\mathrm{T}}\nabla_{xx}p_2(\boldsymbol{x}^{(k)}, \mu)\boldsymbol{p} < 0$ is a *direction of negative curvature* for $p_2(\boldsymbol{x}, \mu)$ at $\boldsymbol{x}^{(k)}$.

Clearly the quadratic approximation

$$Q(\boldsymbol{x}^{(k)} + \alpha\boldsymbol{p}) = p_2(\boldsymbol{x}^{(k)}, \mu) + \alpha\boldsymbol{p}^{\mathrm{T}}\nabla_x p_2(\boldsymbol{x}^{(k)}, \mu) + \tfrac{1}{2}\alpha^2\boldsymbol{p}^{\mathrm{T}}\nabla_{xx}p_2(\boldsymbol{x}^{(k)}, \mu)\boldsymbol{p} \quad (3.1)$$

of the penalty function about $\boldsymbol{x}^{(k)}$ is a decreasing function of α for all $\alpha > 0$ provided we choose the sign of the direction of negative curvature $\boldsymbol{p}$ so that $\boldsymbol{p}^{\mathrm{T}}\nabla_x p_2(\boldsymbol{x}^{(k)}, \mu) \leq 0$.

DEFINITION. Any vector $\boldsymbol{p}$ such that $\nabla_{xx}p_2(\boldsymbol{x}^{(k)}, \mu)\boldsymbol{p} = \boldsymbol{0}$ and $\boldsymbol{p}^{\mathrm{T}}\nabla_x p_2(\boldsymbol{x}^{(k)}, \mu) < 0$ is known as a *direction of linear infinite descent* for $p_2(\boldsymbol{x}, \mu)$ at $\boldsymbol{x}^{(k)}$.

Once again, the quadratic approximation (3.1) is a decreasing function of α whenever $\boldsymbol{p}$ is a direction of linear infinite descent.

DEFINITION. The *inertia* of the $m \times m$ real symmetric matrix M is the triple $\operatorname{In} M = (m_+, m_-, m_0)$ where m_+, m_-, m_0 are respectively the number of positive, negative, and zero eigenvalues of M (with multiplicities counted according to their occurrence). It is well known that $m_+ + m_- + m_0 = m$.

LEMMA 3.1 (Haynsworth, 1968). *Suppose* $M = \begin{bmatrix} M_1 & M_2 \\ M_2^{\mathrm{T}} & M_3 \end{bmatrix}$ *where M is real and symmetric. Then if M_1 is non-singular*

$$\operatorname{In} M = \operatorname{In} M_1 + \operatorname{In}(M_3 - M_2^{\mathrm{T}}M_1^{-1}M_2).$$

If M_3 is non-singular

$$\operatorname{In} M = \operatorname{In} M_3 + \operatorname{In}(M_1 - M_2M_3^{-1}M_2^{\mathrm{T}}).$$

Where it is convenient we shall omit the superscripts k: all matrices and vectors

are assumed to be evaluated at $x^{(k)}$. We shall use Lemma 3.1 to tell us when $\nabla_{xx}p_2(x^{(k)}, \mu) \equiv \bar{G} + \mu^{-1}A^TA$ is positive definite, positive semidefinite, or singular.

DEFINITION. Let In $(\bar{G} + \mu^{-1}A^TA) = (g_+, g_-, g_0)$, and In $K = (k_+, k_-, k_0)$ where K is as given by (2.4). Furthermore, if $\bar{G}$ is non-singular, let In $\bar{G} = (h_+, h_-, 0)$ and In $(A\bar{G}^{-1}A^T + \mu I) = (a_+, a_-, a_0)$.

THEOREM 3.2. *Let the inertias of $\bar{G} + \mu^{-1}A^TA$ and K be as given. Then, if A is $t \times n$ and $\mu > 0$,*

$$(k_+, k_-, k_0) = (g_+, g_-, g_0) + (0, t, 0).$$

Furthermore, if $\bar{G}$ is non-singular and the inertias of $\bar{G}$ and $A\bar{G}^{-1}A^T + \mu I$ are as given,

$$(g_+, g_-, g_0) + (0, t, 0) = (h_+, h_-, 0) + (a_-, a_+, a_0).$$

Proof. The result follows from Lemma 3.1 with $M = K$, $M_1 = \bar{G}$, and $M_3 = -\mu I$. □

COROLLARY 3.3. *$\bar{G} + \mu^{-1}A^TA$ is positive definite if and only if $k_0 = 0$ and $k_- = t$. If $\bar{G}$ is nonsingular, $\bar{G} + \mu^{-1}A^TA$ is positive definite if and only if $a_0 = 0$ and $a_- = h_-$.*

COROLLARY 3.4. *If the inertias of $\bar{G}$, $A\bar{G}^{-1}A^T + \mu I$ and K are as given, there is a direction of negative curvature for $\bar{G} + \mu^{-1}A^TA$ if and only if $k_- > t$ which (when $\bar{G}$ is nonsingular) is if and only if $h_- + a_+ > t$.*

COROLLARY 3.5. *The following are equivalent:*

(i) *$\bar{G} + \mu^{-1}A^TA$ is positive semidefinite and singular and* (1.1) *has no solution.*

(ii) *$k_- = t$, $k_0 > 0$, and* (2.2) *has no solution.*

If $\bar{G}$ is nonsingular, (i) and (ii) are equivalent to

(iii) *$a_- + a_0 = h_-$, $a_0 > 0$, and* (2.5a) *has no solution.*

Finally each of (i)–(iii) imply the existence of a direction of linear infinite descent.

Proof. The equivalence of (i)–(iii) follows from Theorem 3.2 and from the equivalence of (1.1), (2.2), and (2.5). If there were no direction of linear infinite descent, $p^T\nabla_x p_2(x^{(k)}, \mu) = 0$ whenever $\nabla_{xx}p_2(x^{(k)}, \mu)p = 0$. Thus $\nabla_x p_2(x^{(k)}, \mu)$ lies in the range of $\nabla_{xx}p_2(x^{(k)}, \mu)$ and (1.1) has a solution. This contradicts (i). □

Corollaries 3.3–3.5 indicate when we should expect to find a direction of linear infinite descent or (preferably) a direction of negative curvature. We now indicate how such a direction of negative curvature may be obtained. The methods are almost identical to those given in Conn & Gould (1984). Indeed the only real difference is that the methods given there are appropriate when matrices of the form

$$\begin{bmatrix} \bar{G} & A^T \\ A & 0 \end{bmatrix} \quad \text{or} \quad A\bar{G}^{-1}A^T \text{ and } \bar{G}$$

are present. In the present case we have the slightly more general

$$K = \begin{bmatrix} \bar{G} & A^T \\ A & -\mu I \end{bmatrix} \quad \text{or} \quad A\bar{G}^{-1}A^T + \mu I \text{ and } \bar{G}.$$

For completeness, we briefly explain the procedures used. Suppose we have the matrix K and wish to find a direction of negative curvature for $\bar{G} + \mu^{-1}A^TA$. Then if we can find vectors $\boldsymbol{p}$ and $\boldsymbol{r}$ such that

$$\begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{r} \end{bmatrix}^T \begin{bmatrix} \bar{G} & A^T \\ A & -\mu I \end{bmatrix} \begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{r} \end{bmatrix} < 0, \qquad A\boldsymbol{p} = \mu \boldsymbol{r}, \tag{3.2a, b}$$

we obtain, on expanding (3.2a),

$$0 > \boldsymbol{p}^T(\bar{G} + \mu^{-1}A^TA)\boldsymbol{p} - \frac{1}{\mu}(A\boldsymbol{p} - \mu\boldsymbol{r})^T(A\boldsymbol{p} - \mu\boldsymbol{r}) = \boldsymbol{p}^T(\bar{G} + \mu^{-1}A^TA)\boldsymbol{p} \text{ from (3.2b)}.$$

Thus $\boldsymbol{p}$ is a direction of negative curvature. To satisfy (3.2a), recall that we are assuming that $\bar{G} + \mu^{-1}A^TA$ is indefinite and hence K has more than t negative eigenvalues. Let $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{k_-}$ be such that $\boldsymbol{v}_i^TK\boldsymbol{v}_j = 0$ $(i \neq j)$ and $\boldsymbol{v}_i^TK\boldsymbol{v}_i < 0$. (For instance, the $\boldsymbol{v}_i$ could be eigenvectors corresponding to the negative eigenvalues of K.) It is clear that any nontrivial linear combination $\boldsymbol{v} = \sum \alpha_i \boldsymbol{v}_i$ satisfies $\boldsymbol{v}^TK\boldsymbol{v} < 0$. On letting $\boldsymbol{v}^T = [\boldsymbol{p}^T \quad \boldsymbol{r}^T]$, we therefore satisfy (3.2a). Moreover, it is possible to find a nontrivial solution to

$$A\boldsymbol{p} - \mu\boldsymbol{r} = [A \quad -\mu I]\boldsymbol{v} = [A \quad -\mu I]\sum_{i=1}^{k_-} \alpha_i \boldsymbol{v}_i = \sum_{i=1}^{k_-} \alpha_i [A \quad -\mu I]\boldsymbol{v}_i = 0$$

as the vectors $[A \quad -\mu I]\boldsymbol{v}_i$ are of length $t < k_-$. Thus we satisfy (3.2b). The mechanics behind finding the $\boldsymbol{v}_i$ and computing the α_i when a Generalized Cholesky factorization of K is known are given by Conn & Gould (1984). In general, the computation may be performed extremely efficiently and in a stable fashion. Note that the method does not require that we find the eigenvalues or eigenvectors of K.

Alternatively suppose we are given the matrices $\bar{G}$ and $A\bar{G}^{-1}A^T + \mu I$ and wish to find a direction of negative curvature for $\bar{G} + \mu^{-1}A^TA$ using these matrices. Suppose we can find vectors $\boldsymbol{r}$ and $\boldsymbol{s}$ which satisfy

$$\boldsymbol{s}^T\bar{G}\boldsymbol{s} < 0, \qquad \boldsymbol{r}^T(A\bar{G}^{-1}A^T + \mu I)\boldsymbol{r} \geq 0, \tag{3.3a}$$

and

$$A\boldsymbol{s} - A\bar{G}^{-1}A^T\boldsymbol{r} - \mu\boldsymbol{r} = \boldsymbol{0}. \tag{3.3b}$$

Now if we define $\boldsymbol{p} = \boldsymbol{s} - \bar{G}^{-1}A^T\boldsymbol{r}$, we obtain

$$\boldsymbol{p}^T(\bar{G} + \mu^{-1}A^TA)\boldsymbol{p} = \boldsymbol{s}^T\bar{G}\boldsymbol{s} - \boldsymbol{r}^T(A\bar{G}^{-1}A^T + \mu I)\boldsymbol{r} < 0.$$

Thus $\boldsymbol{p}$ is a direction of negative curvature. To satisfy (3.3a) let $\boldsymbol{s}_i$ $(i = 1, \ldots, h_-)$ and $\boldsymbol{r}_i$ $(i = 1, \ldots, a_+)$ be chosen so that $\boldsymbol{s}_i^T\bar{G}\boldsymbol{s}_j = 0$ $(i \neq j)$, $\boldsymbol{s}_i^T\bar{G}\boldsymbol{s}_i < 0$, $\boldsymbol{r}_i^T(A\bar{G}^{-1}A^T + \mu I)\boldsymbol{r}_j = 0$ $(i \neq j)$, and $\boldsymbol{r}_i^T(A\bar{G}^{-1}A^T + \mu I)\boldsymbol{r}_i > 0$. (For instance, the $\boldsymbol{s}_i$ could be eigenvectors corresponding to the negative eigenvalues of $\bar{G}$ and the $\boldsymbol{r}_i$ could be eigenvectors corresponding to the positive eigenvalues of $A\bar{G}^{-1}A^T + \mu I$.

We note that a_+ may be zero in which case we choose $\boldsymbol{r} = \boldsymbol{0}$.) It is clear that any linear combinations

$$\boldsymbol{r} = \sum_{i=1}^{a_+} \alpha_i \boldsymbol{r}_i \quad \text{and} \quad \boldsymbol{s} = \sum_{i=1}^{h_-} \beta_i \boldsymbol{s}_i$$

satisfy (3.3a). In order to satisfy (3.3b) we note that

$$A\boldsymbol{s} - A\bar{G}^{-1}A^{\mathsf{T}}\boldsymbol{r} - \mu\boldsymbol{r} = \sum_{i=1}^{h_-} \beta_i(A\boldsymbol{s}_i) - \sum_{i=1}^{a_+} \alpha_i(A\bar{G}^{-1}A^{\mathsf{T}} + \mu I)\boldsymbol{r}_i = \boldsymbol{0}$$

is a system of t equations in the $h_- + a_+$ $(>t)$ unknown α_i and β_i. Thus there must be a nontrivial solution to these equations. Once again, Conn & Gould (1984) show how this may be achieved in an efficient and stable fashion when Generalized Cholesky factorizations of $\bar{G}$ and $A\bar{G}^{-1}A^{\mathsf{T}} + \mu I$ are known.

Lastly, directions of linear infinite descent are very easy to obtain whenever the Generalized Cholesky factorization of the matrices relevant to Corollary 3.5 are known. The methods are essentially identical to those given in Conn & Gould (1984, §2.2). The interested reader should consult that paper. The procedures described also indicate how weak solutions to (1.1), (2.2), and (2.5a, b) may be found when these equations are consistent.

4. The algorithm

In this section, we propose the following algorithmic framework for solving the nonlinear programming problem NLP.

ALGORITHM (for fixed μ). Given initial data $\boldsymbol{x}^{(0)}$, an estimate of the solution $\boldsymbol{x}(\mu)$, $k = 0$, iterate:

STEP 1. Compute $\bar{G}^{(k)}$, $\boldsymbol{g}^{(k)}$, $\boldsymbol{c}^{(k)}$, $A^{(k)}$, and t, the number of rows of $A^{(k)}$. Find a Generalized Cholesky factorization of

$$K^{(k)} = \begin{bmatrix} \bar{G}^{(k)} & A^{(k)\mathsf{T}} \\ A^{(k)} & -\mu I \end{bmatrix}.$$

Obtain k_-, the number of negative eigenvalues and k_0, the number of zero eigenvalues, of the matrix from the factorization. If $\|\boldsymbol{g}^{(k)} + \mu^{-1}A^{(k)\mathsf{T}}\boldsymbol{c}^{(k)}\|$ is sufficiently small, and $k_- = t$ and $k_0 = 0$, stop.

STEP 2. If $k_- = t$ and $k_0 = 0$ calculate $p^{(k)}$ and $r^{(k)}$ from

$$\begin{bmatrix} \bar{G}^{(k)} & A^{(k)\mathsf{T}} \\ A^{(k)} & -\mu I \end{bmatrix} \begin{bmatrix} \boldsymbol{p}^{(k)} \\ \boldsymbol{r}^{(k)} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{g}^{(k)} \\ \boldsymbol{c}^{(k)} \end{bmatrix}. \tag{4.1}$$

If $k_- > t$, calculate a direction of negative curvature $\boldsymbol{p}^{(k)}$ as described in Section 3.

Otherwise calculate a weak solution to (4.1) or find a direction of linear infinite descent.

STEP 3. Perform a line search along $p^{(k)}$ from $x^{(k)}$ for the penalty function $p_2(x, \mu)$ so that $p_2(x^{(k)} + \alpha^{(k)}p^{(k)}, \mu)$ is sufficiently smaller (in the sense of, for instance, Armijo, 1966) than $p_2(x^{(k)}, \mu)$.

STEP 4. Set $x^{(k+1)} = x^{(k)} + \alpha^{(k)}p^{(k)}$ and $k := k + 1$. Return to step 1. □

As we have taken care to ensure that $p^{(k)}$ is always a descent direction and that our step size $\alpha^{(k)}$ gives us sufficient decrease on the penalty function $p_2(x, \mu)$, the above algorithm should produce a sequence of iterates $x^{(k)}$ which either converge to a local minimizer $x(\mu)$ of $p_2(x, \mu)$ or decrease $p_2(x, \mu)$ without bound. We would also expect the method to converge quadratically in some neighbourhood of $x(\mu)$ since (4.1) are just Newton's equations for minimizing $p_2(x, \mu)$. The dominant costs per iteration are those incurred in calculating the problem functions and derivatives and those incurred in factorizing $K^{(k)}$ (and perhaps in obtaining a direction of infinite descent). The factorization will require about $\frac{1}{6}(n + t)^3$ flops (floating point multiplications or divisions). The calculation of a direction of negative curvature may cost as many as $\frac{1}{3}t^3$ flops (although this is in the worst case, see Conn & Gould (1984)). All other costs are $O(n^2)$ flops.

5. Numerical experience

A prototype implementation of a method for solving NLP based upon the algorithm given in Section 4 has been written in Double Precision Fortran 77 and tested on a Honeywell 6066 at the University of Waterloo. The algorithm in Section 4 is used as a subroutine in the following scheme:

ALGORITHM MINISH:
Given μ_0 and $\mu_{\min}$ such that $0 < \mu_{\min} < \mu_0$, set $\mu = \mu_0$. Then

$$\begin{aligned} \text{until} \quad & \mu < \mu_{\min} \\ & \text{solve:} \quad \text{minimize} \quad p_2(x, \mu), x \in \mathbb{R}^n, \\ & \mu := \mu \times 10^{-2}. \end{aligned}$$

In all cases μ_0 was taken as 10^{-1} and $\mu_{\min}$ was set to 10^{-12}. For a particular value of μ, we considered $x^{(k)}$ an adequate approximation to $x(\mu)$ when $\| \nabla_x p_2(x^{(k)}, \mu) \|_2 \leq 10^{-15}/\mu$ and when $k_- = t$. The dependence of this test on μ merely indicates the expected accuracy which we believed possible given that we have to form $\mu^{-1}\nabla_x c_i(x^{(k)})$ numerically in order to find $\nabla_x p_2(x^{(k)}, \mu)$ and that small inaccuracies in forming $\nabla_x c_i(x^{(k)})$ will be magnified by μ^{-1}. MINISH has been used to solve a subset of the (nontrivial) test problems given by Hock & Schittkowski (1980) from the prescribed starting points and the results of the tests are given in Table 1. We note, however, that the constraints in all problems were *rescaled* according to the following scheme:
Pick a 'typical' value of x, x_{typ} (in the absence of any better value, the starting point for the test problem). Evaluate the constraint functions at x_{typ} and at $x_{\text{typ}} + y$, where the ith element of y satisfies

$$y_i = 10^{-2} \max \{1, |(x_{\text{typ}})_i|\} r_i$$

TABLE 1
Tests results of the algorithm on a subset of test problems from Hock & Schittkowski (1980)

Test problem (Hock & Schittkowski)	Number of variables	Number of constraints	Number of gradient evaluations made	Approximate residual of active constraints (unscaled)
43	4	3	30	10^{-12}
64	3	4	32	10^{-5}
68(I)	4	10	128	10^{-10}
69(I)	4	10	60	10^{-9}
74	4	13	27	10^{-4}
75	4	13	21	10^{-4}
78(I)	5	3	34	10^{-10}
80(I)	5	13	25	10^{-13}
81(I)	5	13	24	10^{-13}
83(I)	5	16	34	10^{-9}
86(I)	5	15	24	10^{-10}
106(S)	8	22	168	10^{-11}
111(I)	10	23	48	10^{-10}
112(I)	10	13	23	10^{-9}
117(IS)	15	20	92	10^{-9}

I indicates an indefinite matrix was encountered
S indicates a singular matrix was observed.

and r_i is a random number uniformly distributed in $[-1, 1]$. Compute

$$\Delta c_i = |c_i(x_{\text{typ}}) - c_i(x_{\text{typ}} + y)|, \qquad \Delta c_{\min} = \min\{\Delta c_i \in \mathbb{R}_+ : 1 \leqslant i \leqslant m\}.$$

Then solve the rescaled problem

$$\overline{\text{NLP}}: \quad \text{minimize} \quad f(x), x \in \mathbb{R},$$
$$\text{s.t.} \quad \bar{c}_i(x) = 0 \ (i \in \mathscr{E}), \quad \bar{c}_i(x) \geqslant 0 \ (i \in \mathscr{I}),$$

where

$$\bar{c}_i(x) = \frac{\Delta c_{\min}}{\Delta c_i} c_i(x),$$

if $\Delta c_i > 0$, and $\bar{c}_i(x) = c_i(x)$ otherwise.

The idea is just to try to ensure that 'typical' perturbations in the values of x produce 'similar' perturbations in all of the constraints. As penalty functions in general, and the quadratic penalty function in particular, are sensitive to the relative sizes of the constraints, it seems sensible to incorporate some sort of pre-scaling. We note that this scaling is done automatically. Clearly the above scheme is not foolproof. However, it has performed very well on our admittedly limited testing.

In Table 1 we have included details of the problems solved, the number of gradient/Hessian evaluations made (1 gradient evaluation includes all of the gradients of the currently violated constraints) and the 2-norm of the residuals of the unscaled constraints active at the solution found. In all cases the solution

identified was that quoted in Hock & Schittkowski (1980) (excepting that the misprint in the 3rd constraint in problem 112, where the constant -1 is omitted was corrected). The implementation of our algorithm is relatively crude, at present.

In particular a simple backtracking line search is used starting with the estimate $\alpha^{(k)} = 1$ and reducing $\alpha^{(k)}$ by 0.5 until $p_2(\boldsymbol{x}^{(k)} + \alpha^{(k)}\boldsymbol{p}^{(k)}, \mu)$ is sufficiently less than $p_2(\boldsymbol{x}^{(k)}, \mu)$ (in the sense of Armijo, 1966). If $\boldsymbol{p}^{(k)}$ is not the Newton direction (4.1) and $p_2(\boldsymbol{x}^{(k)} + \boldsymbol{p}^{(k)}, \mu)$ is significantly less than $p_2(\boldsymbol{x}^{(k)}, \mu)$, we allow $\alpha^{(k)}$ to *increase* by a factor of 2 until $p_2(\boldsymbol{x}^{(k)} + 2\alpha^{(k)}\boldsymbol{p}^{(k)}, \mu)$ is no longer acceptable. A more sophisticated line search would, in all likelihood, enhance the performance of the algorithm.

In no instance was there any indication of a poorly determined search direction. The expected asymptotic quadratic convergence of the iterates was observed for all values of μ encountered. The test problems indicated by (I) and (S) respectively encountered regions in which $\nabla_{xx}p_2(\boldsymbol{x}, \mu)$ was indefinite or singular and progress towards $\boldsymbol{x}(\mu)$ using directions of infinite descent was made. For those problems whose solutions are known analytically, the computed and known solutions differ, typically by 10^{-10}. For the other problems, our solutions agree with those of Hock & Schittkowski to the accuracy given in their report.

The rather large residuals for some active constraints (Problems 64, 74, 75) were a consequence of the high nonlinearity of the constraints for these problems and the scaling procedure used. By reducing μ further, more accurate answers were possible. For instance, the residuals of Problem 74 were forced down to 10^{-14} by reducing μ to 10^{-24} at the cost of an extra 16 gradient evaluations. For most of the problems solved, our scaling procedure did not significantly affect the efficiency of the algorithm. However, this is not always the case; when Problems 68, 106, and 117 were solved unscaled, the number of iterations increased dramatically. None the less, the algorithm still converged. The dominant extra cost appears to have been in finding $\boldsymbol{x}(10^{-1})$. Thereafter the observed convergence is always fast.

6. Extensions

As we indicated in the introduction, the ideas presented in Section 2 carry over to other penalty functions in a straightforward way. If we consider the barrier function $p_B(\boldsymbol{x}, \mu)$ the first and second derivatives at the feasible point $\boldsymbol{x}$ are given by

$$\nabla_x p_B(\boldsymbol{x}, \mu) = \boldsymbol{g}(\boldsymbol{x}) - \bar{A}(\boldsymbol{x})^T \bar{\boldsymbol{\lambda}}(\boldsymbol{x}, \mu)$$

and

$$\nabla_{xx} p_B(\boldsymbol{x}, \mu) = G(\boldsymbol{x}) - \sum_{i \in \mathcal{I}} \bar{\lambda}_i(\boldsymbol{x}, \mu) G_i(\boldsymbol{x}) + \bar{A}(\boldsymbol{x})^T \bar{D}(\boldsymbol{x}, \mu) \bar{A}(\boldsymbol{x}),$$

where $\bar{\boldsymbol{\lambda}}(\boldsymbol{x}, \mu)$ is the vector whose ith element $\bar{\lambda}_i(\boldsymbol{x}, \mu)$ is $-\mu\varphi'[c_i(\boldsymbol{x})]$, $\bar{A}(\boldsymbol{x})$ is the Jacobian matrix of the constraint functions $c_i(\boldsymbol{x})$, and $\bar{D}(\boldsymbol{x}, \mu)$ is the positive definite diagonal matrix whose ith diagonal element is $\mu\varphi''[c_i(\boldsymbol{x})]$. Under very mild conditions, $\boldsymbol{x}(\mu)$ (a minimizer of $p_B(\boldsymbol{x}, \mu)$) converges to $\boldsymbol{x}^*$ and $\boldsymbol{\lambda}[\boldsymbol{x}(\mu), \mu]$

converges to λ^*, the vector of Lagrange multipliers for NLP, as μ tends to zero (see Fiacco & McCormick, 1968). However as μ tends to zero, some of the diagonal elements of $\bar{D}(x, \mu)$ may increase without bound. Hence the Newton equations

$$\nabla_{xx} p_B(x^{(k)}, \mu) p^{(k)} = -\nabla_x p_B(x^{(k)}, \mu) \tag{6.1}$$

for finding a correction $p^{(k)}$ to the iterate $x^{(k)}$ are likely to be badly conditioned when μ is small. To avoid this problem we must first isolate the large and small components of $\bar{D}(x, \mu)$. To this end, let τ be a given positive number ($\tau = 1$, say) and define index sets $\mathcal{S}$ and $\mathcal{L}$ from

$$\mathcal{S}(x, \mu) = \{i \in \mathcal{I} : \mu\varphi''[c_i(x)] < \tau\} \quad \text{and} \quad \mathcal{L}(x, \mu) = \mathcal{I} \setminus \mathcal{S}(x, \mu).$$

Under the usual conditions we have

$$\lim_{\mu \to 0_+} \mu\varphi''(c_i[x(\mu)]) = \begin{cases} 0 & \text{if } c_i(x^*) > 0, \\ +\infty & \text{if } c_i(x^*) = 0. \end{cases}$$

Therefore, as

$$\lim_{\mu \to 0_+} \mathcal{S}[x(\mu), \mu] = \{i \in \mathcal{I} : c_i(x^*) > 0\}$$

and

$$\lim_{\mu \to 0_+} \mathcal{L}[x(\mu), \mu] = \{i \in \mathcal{I} : c_i(x^*) = 0\},$$

we see that $\mathcal{L}(x, \mu)$ and $\mathcal{S}(x, \mu)$ are predictions of the 'active' and 'strictly satisfied' constraints at x^* and that the ill-conditioning in (6.1) is due to the diagonal elements in $\bar{D}(x, \mu)$ corresponding to those constraints which are active at x^*. We may then write (6.1) as

$$[\bar{G}(x^{(k)}, \mu) + A(x^{(k)})^T D(x^{(k)}, \mu) A(x^{(k)})] p^{(k)} = -[\bar{g}(x^{(k)}, \mu) - A(x^{(k)})^T \lambda(x^{(k)}, \mu)], \tag{6.2}$$

where

$$\bar{G}(x, \mu) = G(x) - \sum_{i \in \mathcal{I}} \bar{\lambda}_i(x, \mu) G_i(x) + \sum_{i \in \mathcal{S}(x, \mu)} \mu\varphi''[c_i(x)] \nabla_x c_i(x) \nabla_x c_i(x)^T,$$

$$\bar{g}(x, \mu) = g(x) - \sum_{i \in \mathcal{S}(x, \mu)} \bar{\lambda}_i(x, \mu) \nabla_x c_i(x),$$

and where A, D, and λ comprise the rows, rows and columns, and entries (respectively) of $\bar{A}$, $\bar{D}$, and $\bar{\lambda}$ indexed by $\mathcal{L}(x, \mu)$. Defining $r^{(k)}$ by

$$r^{(k)} = D(x^{(k)}, \mu) A(x^{(k)}) p^{(k)} - \lambda(x^{(k)}, \mu),$$

equations (6.2) give rise to

$$\begin{bmatrix} \bar{G}(x^{(k)}, \mu) & A(x^{(k)})^T \\ A(x^{(k)}) & -D(x^{(k)}, \mu)^{-1} \end{bmatrix} \begin{bmatrix} p^{(k)} \\ r^{(k)} \end{bmatrix} = -\begin{bmatrix} \bar{g}(x^{(k)}, \mu) \\ b^{(k)} \end{bmatrix}, \tag{6.3}$$

where $b^{(k)}$ is the vector whose components are $\varphi'[c_i(x^{(k)})]/\varphi''[c_i(x^{(k)})]$ with $i \in \mathcal{L}(x^{(k)}, \mu)$. Equations (6.3) are well conditioned in the sense that as $\mu \to 0$ and

$x^{(k)} \to x^*$, the coefficient matrix of (6.3) approaches

$$\begin{bmatrix} \nabla_{xx}\ell(x^*, \lambda^*) & A(x^*)^T \\ A(x^*) & 0 \end{bmatrix};$$

the very matrix which indicates the conditioning of NLP (see Section 2). The only modification required in the remaining analysis of Sections 2 and 3 is to replace μI by $D(x^{(k)}, \mu)^{-1}$ whenever the former appears.

The same analysis may also be applied to the limiting penalty function $p_L(x, \mu)$ for semi-finite programming. Complications are introduced because the penalty function contains integral terms. For this reason we shall not consider $p_L(x, \mu)$ further here but refer the interested reader to Conn & Gould (1985).

An obvious extension is to allow approximations to $\bar{G}(x^{(k)}, \mu)$ in (2.2) or (2.5) rather than use the exact matrices. We have in mind obtaining secant approximations to $\bar{G}$. Indeed, much work has already been done in this area, since the matrix we are trying to approximate approaches $\nabla_{xx}\mathcal{L}(x^*, \lambda^*)$ as $x \to x^*$. Methods for approximating such matrices have been suggested by Biggs (1975), Han (1976), and Powell (1978). We note that our general framework does not require that $\bar{G}(x^{(k)}, \mu)$ be approximated by a positive definite matrix. Although our current implementation does not include such possibilities, we intend to consider them in the future.

Finally, we believe that a special purpose line search algorithm for the quadratic penalty function would be appropriate. Special purpose line searches for problems involving sums of squares have already been suggested (Lindström & Wedin, 1984). As $p_2(x, \mu)$ is merely a sum of squares with (possibly) one extra term ($f(x)$) it would seem reasonable to devise a special line search for this problem. There are, of course, already special purpose line searches for the barrier functions (see, for example, Fletcher & McCann, 1969, Murray & Wright, 1976).

7. Acknowledgements

The author would like to thank Charles Broyden for the many fruitful discussions during his visit to Waterloo in October 1984 and Andrew Conn for his usual encouragement and helpful comments. Many thanks are also due to Joan Selwood for her excellent preparation of the typescript. Finally, the author is grateful to Mike Powell and his referees for suggesting a number of helpful improvements.

This research was made possible by a Natural Sciences and Engineering Research Council of Canada Grant A-8442.

References

Armijo, L. 1966 Minimization of function have continuous partial derivatives. *Pacific J. Math* **16**, 1–3.

Biggs, M. C. 1972 Constrained minimization using recursive equality quadratic program-

ming. In: F. A. Lootsma, ed., *Numerical Methods for Nonlinear Optimization.* New York: Academic Press.

BIGGS, M. C. 1975 Constrained minimization using recursive quadratic programming. In: L. C. W. Dixon and G. P. Szego, eds., *Towards Global Optimization.* North Holland.

BIGGS, M. C. 1978 On the convergence of some constrained minimization algorithms based on quadratic programming. *J. Inst. Maths Its Applics* **21,** 67–81.

BROYDEN, C. G., & ATTIA, N. F. 1983 A smooth sequential penalty function method for solving nonlinear programming problems, preprint, (1983) Department of Computer Science, University of Essex, England; presented at the IFIP Conference, Copenhagen, July.

BUNCH, J. R., & KAUFMAN, L. C. 1977 Some stable methods for calculating inertia and solving symmetric linear equations. *Mathematics of Computation* **31,** 163–179.

BUNCH, J. R., & PARLETT, B. N. 1971 Direct methods for solving symmetric indefinite systems of linear equations. *SIAM J. Numerical Analysis* **8,** 639–655.

CONN, A. R., & GOULD, N. I. M. 1984 On the location of directions of infinite descent for nonlinear programming algorithms. *SIAM J. Numerical Analysis* **21,** 1162–1179.

CONN, A. R., & GOULD, N. I. M. 1985 Solving semi-infinite programs via a differentiable penalty function. Dept. of Computer Science, University of Waterloo, in preparation.

DUFF, I. S., & REID, J. K. 1983 The multifrontal solution of indefinite sparse symmetric linear systems, *ACM Trans. Math. Software* **9,** 302–325.

FIACCO, A. V., & MCCORMICK, G. P. 1968 *Nonlinear Programming: Sequential Unconstrained Minimization Techniques.* New York: John Wiley and Sons.

FLETCHER, R. 1976 Factorizing symmetric indefinite matrices. *Linear Algebra and its Applics* **14,** 257–272.

FLETCHER, R. 1981 *Practical Methods of Optimization, Vol. 2: Constrained Optimization.* New York: John Wiley and Sons.

FLETCHER, R., & MCCANN, A. P. 1969 Acceleration techniques for nonlinear programming. In: R. Fletcher, ed., *Optimization.* New York: Academic Press, 37–49.

GERENCSER, L. 1974 A second-order technique for the solution of nonlinear optimization problems. *Colloquia Mathematica Societatis Janos Bolyai* **12,** Progress in Operations Research, EGER Hungary.

GILL, P. E., MURRAY, W., & WRIGHT, M. H. 1981 *Practical Optimization.* London and New York: Academic Press.

GRIEWANK, A., & TOINT, P. L. 1982 On the unconstrained optimization of partially separable function, in: M. J. D. Powell, ed., *Nonlinear Optimization 1981.* New York: Academic Press.

HAN, S. P. 1976 Superlinearly convergent variable metric algorithms for general non-linear programming problems. *Mathematical Programming* **11,** 263–282.

HAYNSWORTH, E. V. 1968 Determination of the inertia of a partitioned Hermitian matrix. *Linear Algebra and its Applics* **1,** 73–81.

HOCK, W., & SCHITTKOWSKI, K. 1980 *Test Examples for Nonlinear Programming Codes.* Lecture Notes in Economics and Mathematical Systems 187. Berlin: Springer-Verlag.

LINDSTRÖM, P., & WEDIN, P.-A. 1984 A new line search algorithm for nonlinear least squares problems. *Mathematical Programming* **29,** 268–296.

LOOTSMA, F. A. 1969 Hessian matrices of penalty functions for solving constrained optimization problems. Phillips Research Reports **24,** 322 331.

MURRAY, W. 1971 Analytical expressions for eigenvalues and eigenvectors of the Hessian matrices of barrier and penalty functions. *J. Optimiz. Theor. Applics* **7,** 189–196.

MURRAY, W., & WRIGHT, M. H. 1976 Efficient linear search algorithms for the logarithmic barrier function, Report SOL 76-18 Stanford University, California.

MURRAY, W., & WRIGHT, M. H. 1978 Projected Lagrangian methods based on the trajectories of penalty and barrier functions, Report SOL 78-23 Stanford University, California.

PIETRZYKOWSKI, T. 1970 The potential method for conditioned maxima in the locally compact metric spaces. *Numerische Math.* **14,** 325–329.

POWELL, M. J. D. 1978 A fast algorithm for nonlinearly constrained optimization calculations. In: *Numerical Analysis, Dundee 1977,* Lecture Notes in Mathematics 630. Berlin: Springer-Verlag, 144–157.

ROBINSON, S. M. 1980 Strongly regular generalized equations. *Mathematics of Operations Research* **5,** 43–62.

WRIGHT, M. H. 1976 Numerical methods for nonlinearly constrained optimization. Report CS-76-566 Stanford University, California.