AN INTERIOR–POINT ALGORITHM FOR NONCONVEX NONLINEAR PROGRAMMING

ROBERT J. VANDERBEI AND DAVID F. SHANNO

Statistics and Operations Research
Princeton University
SOR-97-21

ABSTRACT. The paper describes an interior–point algorithm for nonconvex nonlinear programming which is a direct extension of interior–point methods for linear and quadratic programming. Major modifications include a merit function and an altered search direction to ensure that a descent direction for the merit function is obtained. Preliminary numerical testing indicates that the method is robust. Further, numerical comparisons with MINOS and LANCELOT show that the method is efficient, and has the promise of greatly reducing solution times on at least some classes of models.

1. INTRODUCTION

In this paper, we describe modifications that we used to convert the quadratic programming (QP) solver LOQO into a general nonconvex nonlinear programming solver (of the same name). As a code for quadratic programming, LOQO implements an interior-point method. The complete details of the QP implementation can be found in [25]. In view of the dramatic success of modern interior-point methods for linear and quadratic programming, our guiding philosophy was to modify the QP interior-point method in LOQO as little as possible to make a robust and efficient general nonconvex nonlinear optimizer.

For notational simplicity, we begin by considering the following nonlinear programming problem:

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && h_i(x) \geq 0, \qquad i = 1, \ldots, m, \end{aligned} \tag{1}$$

where x is a vector of dimension n and $f(x)$ and the $h_i(x)$ are assumed to be twice continuously differentiable. This is a simplification of the general nonlinear programming problem, which can include equality constraints and bounds on the variables. In fact, the method developed in this paper and its implementation in LOQO handles all of these cases efficiently, and the exact way in which this is accomplished is discussed in detail later. For

1991 *Mathematics Subject Classification.* Primary 90C30, Secondary 49M37, 65K05.

Key words and phrases. Nonlinear programming, interior-point methods, nonconvex optimization.

Research of the first author supported by NSF grant CCR-9403789 and by ONR grant N00014-98-1-0036. Research of the second author supported by AFOSR grant F49620-95-1-0110.

the present, we consider this version of the problem, as it greatly simplifies the terminology, and the extension to the more general case is quite straightforward.

The interior-point approach taken in this paper is described as follows. First, add slack variables w_i to each of the constraints (1), reformulating the problem as

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && h(x) - w = 0, \\ &&& w \geq 0, \end{aligned} \tag{2}$$

where $h(x)$ and w represent the vectors with elements $h_i(x)$ and w_i respectively. We then eliminate the inequality constraints in (2) by placing them in a barrier term, resulting in the problem

$$\begin{aligned} &\text{minimize} && f(x) - \mu \sum_{i=1}^{m} \log(w_i) \\ &\text{subject to} && h(x) - w = 0, \end{aligned} \tag{3}$$

where the objective function

$$b(x, w; \mu) = f(x) - \mu \sum_{i=1}^{m} \log(w_i) \tag{4}$$

is the classical Fiacco-McCormick [8] logarithmic barrier function. The Lagrangian for this problem is

$$L(x, w, y, \mu) = f(x) - \mu \sum_{i=1}^{m} \log(w_i) - y^T(h(x) - w),$$

and the first-order conditions for a minimum are

$$\begin{aligned} \nabla_x L &= \nabla f(x) - \nabla h(x)^T y = 0, \\ \nabla_w L &= -\mu W^{-1} e + y \quad\;\; = 0, \\ \nabla_y L &= h(x) - w \quad\;\; = 0, \end{aligned} \tag{5}$$

where W is the diagonal matrix with elements w_i, e is the vector of all ones, and $\nabla h(x)$ is the Jacobian matrix of the vector $h(x)$. We now modify (5) by multiplying the second equation by W, producing the standard primal-dual system

$$\begin{aligned} \nabla f(x) - \nabla h(x)^T y &= 0, \\ -\mu e + WYe &= 0, \\ h(x) - w &= 0, \end{aligned} \tag{6}$$

where again Y is the diagonal matrix with elements y_i. Note that the second equation implies that y is nonnegative, which is consistent with the fact that y is the vector of Lagrange multipliers associated with what were originally inequality constraints.

The basis of the numerical algorithm for finding a solution to the primal-dual system (6) is Newton's method, which is well known to be very efficient for linear and convex quadratic programming. In order to simplify notation and at the same time highlight connections with linear and quadratic programming, we introduce the following definitions:

$$H(x, y) = \nabla^2 f(x) - \sum_{i=1}^{m} y_i \nabla^2 h_i(x)$$

and

$$A(x) = \nabla h(x).$$

The Newton system for (6) is then

$$\begin{bmatrix} H(x,y) & 0 & -A(x)^T \\ 0 & Y & W \\ A(x) & -I & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta w \\ \Delta y \end{bmatrix} = \begin{bmatrix} -\nabla f(x) + A(x)^T y \\ \mu e - WYe \\ -h(x) + w \end{bmatrix}. \tag{7}$$

The system (7) is not symmetric, but is easily symmetrized by multiplying the first equation by -1 and the second equation by $-W^{-1}$, yielding

$$\begin{bmatrix} -H(x,y) & 0 & A^T(x) \\ 0 & -W^{-1}Y & -I \\ A(x) & -I & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta w \\ \Delta y \end{bmatrix} = \begin{bmatrix} \sigma \\ -\gamma \\ \rho \end{bmatrix}, \tag{8}$$

where

$$\sigma = \nabla f(x) - A^T(x)y, \tag{9}$$

$$\gamma = \mu W^{-1}e - y, \tag{10}$$

$$\rho = w - h(x). \tag{11}$$

Note that ρ measures *primal infeasibility*. By analogy with linear programming, we refer to σ as *dual infeasibility*. Also, note that σ, γ, and ρ depend on x, y, and w even though we don't show this dependence explicitly.

It is (8), or a small modification of it, that LOQO solves at each iteration to find the search directions Δx, Δw, Δy. Since the second equation can be used to eliminate Δw without producing any off-diagonal fill-in in the remaining system, one normally does this elimination first. Hence, Δw is given by

$$\Delta w = WY^{-1}(\gamma - \Delta y)$$

and the resulting *reduced KKT system* is given by

$$\begin{bmatrix} -H(x,y) & A^T(x) \\ A(x) & WY^{-1} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \sigma \\ \rho + WY^{-1}\gamma \end{bmatrix}. \tag{12}$$

The algorithm then proceeds iteratively from an initial point $x^{(0)}$, $w^{(0)}$, $y^{(0)}$ through a sequence of points determined from the search directions described above:

$$\begin{aligned} x^{(k+1)} &= x^{(k)} + \alpha^{(k)}\Delta x^{(k)}, \\ w^{(k+1)} &= w^{(k)} + \alpha^{(k)}\Delta w^{(k)}, \\ y^{(k+1)} &= y^{(k)} + \alpha^{(k)}\Delta y^{(k)}. \end{aligned} \tag{13}$$

For linear programming different steplengths, $\alpha_{\mathrm{p}}^{(k)}$ and $\alpha_{\mathrm{d}}^{(k)}$, are used with the primal and dual directions, whereas for nonlinear programming a common steplength is employed.

1.1. Formulas for search directions. While LOQO solves the reduced KKT system (12), it is nonetheless useful for subsequent analysis to give explicit formulas for Δx and Δw. Let

$$N(x,y,w) = H(x,y) + A^T(x)W^{-1}YA(x)$$

denote the *dual normal matrix*. In the following theorem, we drop the explicit indication of the dependence of A, N, and ∇f on x, y, and w.

Theorem 1. *If N is nonsingular, then* (8) *has a unique solution. In particular,*

$$\Delta x = -N^{-1}\nabla f + \mu N^{-1}A^T W^{-1}e + N^{-1}A^T W^{-1}Y\rho \tag{14}$$

$$\Delta w = -AN^{-1}\nabla f + \mu AN^{-1}A^T W^{-1}e - (I - AN^{-1}A^T W^{-1}Y)\rho. \tag{15}$$

Remark. The formulas for Δx and Δw involve three terms each. In both cases, the first term can be viewed as an *optimality direction*, the second term as a *centrality direction*, and the third term as a *feasibility direction*.

Proof. Solving the second block of equations in (12) for Δy and eliminating Δy from the first block of equations yields a system involving only Δx whose solution is

$$\Delta x = N^{-1}\left(-\sigma + A^T(W^{-1}Y\rho + \gamma)\right). \tag{16}$$

Using this formula, we can then solve for Δy and finally for Δw. The resulting formula for Δw is:

$$\Delta w = AN^{-1}A^T\gamma - \left(WY^{-1} - AN^{-1}A^T\right)W^{-1}Y\rho - AN^{-1}\sigma. \tag{17}$$

From the definitions of σ and γ, it follows that

$$\sigma - A^T\gamma = \nabla f - \mu A^T W^{-1}e.$$

Finally, we use this formula to eliminate σ and γ from (16) and (17). The formulas (14) and (15) for Δx and Δw then follow easily. □

1.2. Critical features. The critical features of the algorithm are the choices of $\alpha^{(k)}$ at each iteration and the modification of the system (8) in order to find a local minimizer of (1). For linear and convex quadratic programming, modification of the linear system is never required except possibly to deal with problems of numerical accuracy, and the step length at each iteration is determined by a simple ratio test. (See, for example, Lustig, Marsten and Shanno [18] and Vanderbei [25].) For convex nonlinear programming, again the linear system need not be modified, but the method for choosing $\alpha^{(k)}$ at each iteration becomes more complex, as it is well known that for general convex nonlinear problems, with a poor initial estimate Newton's method may diverge. In order to achieve convergence to a solution to the system (6), El Bakry et al. [7] introduced the merit function

$$\Psi_0(x, w, y) = \|\nabla f(x) - A(x)^T y\|^2 + \|WYe\|^2 + \|h(x) - w\|^2 \tag{18}$$

and showed that for a proper choice of μ, there exists a sequence of step lengths $\{\alpha^{(k)}\}$ such that Ψ_0 decreases monotonically and the iterates converge to a point at which $\Psi_0 = 0$ provided the Jacobian of the system (6) remains nonsingular. Shanno and Simantiraki [21] tested a variant of this algorithm on the Hock and Schittkowski set of test problems [16] and found that while the algorithm was often efficient, it also often converged to local maxima or saddle points, all of which satisfy the first-order necessary conditions. Further, the Jacobian of the system (6) sometimes becomes singular, causing the algorithm to fail. Thus it became apparent that a better merit function, and a way of ensuring a nonsingular Jacobian, would be necessary for a successful general algorithm. Our approach to this is discussed in detail in the next section.

1.3. Related work. It has recently come to our attention that Lasdon et al. [12] have also been studying this basic algorithm. While their algorithm differs from ours in many particulars, the general approach is similar. For related work on primal–dual interior–point methods see [10], [2], [13], [5]. An alternative barrier approach to inequality–constrained problems is discussed in [1].

2. ALGORITHM MODIFICATIONS FOR CONVEX OPTIMIZATION

As mentioned in the previous section, the search directions given by (8) together with a steplength selected simply to ensure that the vectors w and y remain component-wise nonnegative yield an efficient and robust algorithm for convex quadratic programming. However, for nonquadratic convex optimization problems, further reduction in the steplength may be necessary to guarantee convergence, as is trivially illustrated by the unconstrained minimization of the univariate function $f(x) = (1 + x^2)^{1/2}$ using an initial estimate $x^{(0)}$ with $|x^{(0)}| > 1$. Merit functions are used to guide one in deciding how much to shorten the steplength. In this section we describe the specific merit function that we have implemented.

2.1. The merit function. Merit functions for equality constrained nonlinear programming have been the subject of a great deal of research over the past twenty-odd years. The idea of a merit function is to ensure that joint progress is made both toward a local minimizer and toward feasibility. This progress is achieved by shortening the steplength along the search directions defined by (8) as necessary so that sufficient reduction in the merit function is made. One possible merit function is

$$\Psi_1(x, \beta) = f(x) + \beta \|\rho(x, w)\|_1. \tag{19}$$

This merit function is *exact*, which means that there exists a β_0 such that, for all $\beta \geq \beta_0$, a minimizer of (19) is guaranteed to be feasible and, under general conditions, a local minimizer of the original problem. While exactness is a useful property, the nondifferentiability of ℓ_1-norms can cause difficulties with numerical algorithms. The merit function defined by

$$\Psi_2(x, \beta) = f(x) + \frac{\beta}{2} \|\rho(x, w)\|_2^2 \tag{20}$$

is the penalty function for equality constrained nonlinear programming studied by Fiacco and McCormick [8]. It is differentiable everywhere. However, it has the theoretical disadvantage of requiring β to tend to infinity to ensure convergence to a feasible point, which again is hoped to be a local minimizer of the original problem. In spite of this apparent disadvantage, in practice we have had no such difficulty, and hence have chosen to use it.

The l_2 merit function (20) when applied to problems of the form (3) is

$$\Psi_{\beta,\mu}(x, w) = f(x) - \mu \sum_{i=1}^{m} \log(w_i) + \frac{\beta}{2} \|\rho(x, w)\|_2^2 \tag{21}$$

(recall that $\rho(x, w) = w - h(x)$). The following theorem shows, among other things, that for large enough β's the search directions defined by (8) are descent directions for $\Psi_{\beta,\mu}$ whenever the problem is strictly convex.

Theorem 2. *Suppose that the dual normal matrix N is positive definite. Then, the search directions have the following properties:*

(1) *If $\rho = 0$, then*

$$\begin{bmatrix} \nabla_x b \\ \nabla_w b \end{bmatrix}^T \begin{bmatrix} \Delta x \\ \Delta w \end{bmatrix} \leq 0.$$

(2) *There exists $\beta_{\min} \geq 0$ such that, for every $\beta > \beta_{\min}$,*

$$\begin{bmatrix} \nabla_x \Psi_{\beta,\mu} \\ \nabla_w \Psi_{\beta,\mu} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta w \end{bmatrix} \leq 0$$

In both cases, equality holds if and only if (x, w) satisfies (6) *for some y.*

Proof. It is easy to see that

$$\begin{bmatrix} \nabla_x b \\ \nabla_w b \end{bmatrix} = \begin{bmatrix} \nabla f \\ -\mu W^{-1} e \end{bmatrix}.$$

From the expressions for Δx and Δw given in Theorem 1, we then get

$$\begin{aligned} \begin{bmatrix} \nabla_x b \\ \nabla_w b \end{bmatrix}^T \begin{bmatrix} \Delta x \\ \Delta w \end{bmatrix} &= -\left(\nabla f - \mu A^T W^{-1} e\right)^T N^{-1} \left(\nabla f - \mu A^T W^{-1} e\right) \\ &\quad + \mu e^T W^{-1} \rho \\ &\quad + \left(\nabla f - \mu A^T W^{-1} e\right)^T N^{-1} A^T W^{-1} Y \rho. \end{aligned}$$

Assuming that $\rho = 0$ and that N is positive definite, we see that

$$\begin{bmatrix} \nabla_x b \\ \nabla_w b \end{bmatrix}^T \begin{bmatrix} \Delta x \\ \Delta w \end{bmatrix} = -\left(\nabla f - \mu A^T W^{-1} e\right)^T N^{-1} \left(\nabla f - \mu A^T W^{-1} e\right) \le 0,$$

which completes the proof of the first property.

For the second property, first note that the merit function is the barrier function plus a constant times a measure of infeasibility:

$$\Psi_{\beta,\mu}(x, w) = b(x, w) + \frac{\beta}{2} \|\rho(x, w)\|^2.$$

We now address the infeasibility term. It is easy to check that

$$\begin{bmatrix} \nabla_x \beta \|\rho(x, w)\|_2^2/2 \\ \nabla_w \beta \|\rho(x, w)\|_2^2/2 \end{bmatrix} = \beta \begin{bmatrix} -A^T \\ I \end{bmatrix} \rho.$$

From this explicit expression for the gradient of the infeasibility term, it follows that

$$\begin{bmatrix} \nabla_x \beta \|\rho(x, w)\|_2^2/2 \\ \nabla_w \beta \|\rho(x, w)\|_2^2/2 \end{bmatrix}^T \begin{bmatrix} \Delta x \\ \Delta w \end{bmatrix} = -\beta \rho^T \rho.$$

Combining the inner products using the barrier function and the infeasibility term, we get

$$\begin{aligned} \begin{bmatrix} \nabla_x \Psi_{\beta,\mu} \\ \nabla_w \Psi_{\beta,\mu} \end{bmatrix}^T \begin{bmatrix} \Delta x \\ \Delta w \end{bmatrix} &= -\left(\nabla f - \mu A^T W^{-1} e\right)^T N^{-1} \left(\nabla f - \mu A^T W^{-1} e\right) \\ &\quad + \mu e^T W^{-1} \rho \\ &\quad + \left(\nabla f - \mu A^T W^{-1} e\right)^T N^{-1} A^T W^{-1} Y \rho \\ &\quad - \beta \|\rho\|_2^2. \end{aligned}$$

Suppose that (x, w) is not feasible ($\rho \neq 0$). For $(\Delta x, \Delta w)$ to fail to be a descent direction for the merit function, it is necessary that

$$\mu e^T W^{-1} \rho + \left(\nabla f - \mu A^T W^{-1} e\right)^T N^{-1} A^T W^{-1} Y \rho > 0.$$

When this is the case, setting

$$\begin{aligned} \beta_{\min} &= \Big(-\left(\nabla f - \mu A^T W^{-1} e\right)^T N^{-1} \left(\nabla f - \mu A^T W^{-1} e\right) \\ &\quad + \mu e^T W^{-1} \rho + \left(\nabla f - \mu A^T W^{-1} e\right)^T N^{-1} A^T W^{-1} Y \rho \Big) / \|\rho\|_2^2 \end{aligned} \tag{22}$$

ensures that $(\Delta x, \Delta w)$ is a descent direction for $\Psi_{\beta,\mu}$ for every $\beta > \beta_{\min}$.

Now suppose that (x, w) is feasible (i.e., $\rho = 0$). In this case, the barrier function's inner product and the merit function's inner product agree. Furthermore, either $(\Delta x, \Delta w)$ is a descent direction for both or

$$\left(\nabla f - \mu A^T W^{-1} e\right)^T N^{-1} \left(\nabla f - \mu A^T W^{-1} e\right) = 0.$$

In the latter case, we use the positive definiteness of N to conclude that

$$\left(\nabla f - \mu A^T W^{-1} e\right) = 0.$$

Introducing a dual variable $y = \mu W^{-1} e$, we see that this last equation together with the assumed feasibility comprise precisely the first-order optimality conditions (5) for the barrier problem. □

In LOQO, β is initialized to 0 and is unchanged as long as $(\Delta x, \Delta w)$ is a descent direction for $\Psi_{\beta,\mu}$. When $(\Delta x, \Delta w)$ fails as a search direction, then β is calculated from (22) using $\beta = 10\beta_{\min}$. This algorithm has proved quite satisfactory in practice, as β is changed very few times during the calculations on problems we have tried so far, and on many problems remains 0. While this may seem surprising, recall that the search direction is the Newton direction to try to find a feasible first-order point, and generally moves jointly toward a minimizer and a feasible point. To ensure that we will not simply approach an infeasible optimum very slowly, if the step along the search vector becomes very small, β is increased by a factor of 10. The current version of the code starts the next iteration from the new point with the increased value of β. In future work, we will test the efficiency gained by simply using the new β with the old search directions.

Thus at each step an $\alpha_{\max}$ is chosen by the standard ratio test to ensure that nonnegative variables remain nonnegative. In the context of the discussion of this section, this means that, for $i = 1, 2, \ldots, m$, $w_i + \alpha \Delta w_i$ and $y_i + \alpha \Delta y_i$ must remain strictly positive so

$$\alpha_{\max} = 0.95 \left(\max \left\{ -\frac{\Delta w_i}{w_i}, -\frac{\Delta y_i}{y_i} : i = 1, \ldots, m \right\} \right)^{-1}.$$

Then the interval $[0, \alpha_{\max}]$ is searched by successive halving, using merit-function evaluations only, for a value of α that produces a reduction in the merit function satisfying an Armijo condition.

We conclude this section with two comments. First, the above discussion assumed that N is positive definite. Subsequent sections will demonstrate how the algorithm can be modified to relax this assumption.

Second, the method listed here is admittedly ad hoc. A convergence proof for a method of this kind will certainly require a more carefully developed algorithm in terms of selection of the parameter β. Yet in preliminary testing this method works surprisingly well, while attempts at more rigorous algorithms were less successful. Thus it can be viewed similarly to the recent method of Fletcher and Leyffer [9], which essentially accepts any point that improves either optimality or infeasibility. While much remains for further study, our experience to date is that Newton's method should be hindered as little as possible.

3. ALGORITHM MODIFICATIONS FOR NONCONVEX OPTIMIZATION

Using the merit function described in the previous section to guide the selection of a steplength (together with the stabilization technique of the next section) yields an efficient and robust algorithm for convex optimization. For nonconvex optimization problems, there is another important issue to consider, namely, that the matrix $N(x, y, w) = H(x, y) +$

$A^T(x)W^{-1}YA(x)$ may fail to be positive semidefinite. In this section, we discuss the impact of indefiniteness and describe the algorithmic changes that we made to address this issue.

3.1. The search direction. In Theorem 2 we showed that the search directions $(\Delta x, \Delta w)$ have desirable descent properties for both the barrier function and the merit function provided that N is positive definite. When N is indefinite, the algorithm might still converge to something, but it might not be a local minimum for the problem one wishes to solve. Consider, for example, the problem of minimizing the concave function $4x(1-x)$ subject to the bound constraint $0 \le x \le 1$. The algorithm presented so far when applied to this problem and initialized with $0.4 \le x^{(0)} \le 0.6$ has the very undesirable property of converging to the global maximum at 0.5.

These results suggest replacing $H(x, y)$ in the definition of the search directions with a diagonal perturbation thereof:

$$\hat{H} = H + \lambda I, \qquad \lambda \ge 0.$$

If λ is chosen so that N is positive definite, then Theorem 2 applies and ensures descent properties for the barrier and merit functions. Of course, the following question naturally arises: what is the impact of such a perturbation on other important properties such as reduction in primal and dual infeasibility? The following theorem addresses this question. Henceforth, we assume that the search directions are computed using $\hat{H}$ instead of H.

Theorem 3. *Let*

$$\begin{aligned} &\bar{x} = x + t\Delta x, \quad \bar{y} = y + t\Delta y, \quad \bar{w} = w + t\Delta w, \\ &\bar{\rho} = \rho(\bar{x}, \bar{w}), \quad \bar{\sigma} = \sigma(\bar{x}, \bar{y}). \end{aligned}$$

Then

$$\begin{aligned} \bar{\rho} &= (1-t)\rho + o(t), \\ \bar{\sigma} &= (1-t)\sigma - t\lambda\Delta x + o(t), \\ \bar{w}^T\bar{y} &= (1 - t(1-\delta))w^T y + o(t), \end{aligned}$$

where $\delta = m\mu / w^T y$.

Proof. We begin with $\bar{\rho}$:

$$\begin{aligned} \bar{\rho} &= \bar{w} - h(\bar{x}) \\ &= w + t\Delta w - h(x + t\Delta x) \\ &= w + t\Delta w - h(x) - tA(x)\Delta x + o(t) \\ &= w - h(x) + t(\Delta w - A(x)\Delta x) + o(t) \\ &= (1-t)\rho + o(t), \end{aligned}$$

where the last equality follows from the third block of equations in (8). For $\bar{\sigma}$, the analysis is similar but one must use the first block of equations in (8) with $\hat{H}$ replacing H:

$$\begin{aligned} \bar{\sigma} &= \nabla f(\bar{x}) - A^T(\bar{x})\bar{y} \\ &= \nabla f(x + t\Delta x) - A^T(x + t\Delta x)(y + t\Delta y) \\ &= \nabla f(x) + t\nabla^2 f(x)\Delta x - A^T(x)y - t\nabla_x (y^T A(x))^T \Delta x - tA^T(x)\Delta y + o(t) \\ &= \nabla f(x) - A^T(x)y + t(H(x, y)\Delta x - A^T(x)\Delta y) + o(t) \\ &= (1-t)\sigma - t\lambda\Delta x + o(t). \end{aligned}$$

Finally, the analysis of $w^T y$ follows closely the analogous development for linear programming. It starts as follows:

$$\begin{aligned} \bar{w}^T \bar{y} &= (w + t\Delta w)^T (y + t\Delta y) \\ &= w^T y + t(w^T \Delta y + \Delta w^T y) + o(t). \end{aligned} \tag{23}$$

Then, we rewrite the linear term in t as follows:

$$\begin{aligned} w^T \Delta y + \Delta w^T y &= e^T (W\Delta y + Y\Delta w) \\ &= e^T (\mu e - WYe) \\ &= m\mu - w^T y \\ &= -(1-\delta) w^T y. \end{aligned} \tag{24}$$

Substituting (24) into (23) we get the desired result. □

As for linear programming, μ is usually chosen so that $0 < \delta < 1$ and therefore $w^T y$ decreases provided that the step length is short enough. However, if $\lambda > 0$, Theorem 3 shows that dual infeasibility may fail to decrease even with arbitrarily small steps. This seems a small price to pay for the other desirable properties and, even though we don't have a proof of convergence for the perturbed method, empirical evidence suggests that λ is zero most of the time (using the rules given shortly) and that the dual infeasibility does eventually decrease to zero once the iterates are in a neighborhood of the optimal solution (where N should be positive definite).

In our nonlinear LOQO, we pick $\lambda > 0$ whenever necessary to keep N positive definite. By doing this, we guarantee that the algorithm, if convergent, converges to a local minimum.

The value of λ is computed as follows. We first do an LDL^T factorization of a symmetric permutation of the reduced KKT matrix in (12). The permutation is computed at the beginning based solely on structure/fill-in considerations; see [25] for details. If $H(x, y)$ is positive definite, then the reduced KKT matrix is *quasidefinite* as defined in [24] and hence, as shown in [24], the factorization is guaranteed to exist. Furthermore, it has the property that every diagonal element of D associated with an original diagonal element of $H(x, y)$ must be positive and every element associated with an original diagonal element of $-WY^{-1}$ must be negative. Hence, after factoring, we scan the diagonal elements of D looking for elements of the wrong sign and noting the one with largest magnitude. Let λ_0 denote the largest magnitude. If all of the diagonal elements have the correct sign, then $H(x, y)$ must be positive definite and no perturbation is required. Otherwise, we do an initial perturbation with $\lambda = 1.2\lambda_0$. For the trivial case of a 1×1 matrix, such a perturbation is guaranteed to produce a positive definite matrix. However, for larger matrices, there is no such guarantee. Therefore, we factor the perturbed matrix and check as before. If the perturbation proves to be insufficient, we keep doubling it until a perturbation is found that gives a positive definite matrix. If, on the other hand, the initial perturbation is sufficiently large, then we do successive halving until we find a perturbation that is too small. Finally, we double once. It is not clear whether this halving process is necessary. We plan to investigate this issue in the future. The method described here is very simple, and can undoubtedly be improved, but it has in general worked very well in practice. Note that as w and y are always positive, only $H(x, y)$ ever needs to be modified.

4. BOUNDS AND RANGES

As noted in the introduction, the general nonlinear programming problem may have equality constraints as well as inequality constraints. Furthermore, simple bounds on the variables, while they may be considered as inequality constraints, are generally handled separately for efficiency. All of these cases were treated specifically in the quadratic solver LOQO and are described in [25]. Our nonlinear modification continues to accept all such formulations. We describe the algorithmic issues only briefly here.

A more general form of inequality constraint is a range constraint of the form

$$0 \le h_i(x) \le r_i.$$

This is easily converted to a system of constraints

$$\begin{aligned} h_i(x) - w_i &= 0, \\ w_i + p_i &= r_i, \\ w_i,\, p_i &\ge 0, \end{aligned}$$

where $0 \le r_i < \infty$. Equality constraints are treated by simply declaring them to be range constraints with $r_i = 0$. Note that this does not increase the number of nonlinear constraints, but does add a linear constraint and an extra variable. In [23], Vanderbei shows how to reduce the resulting KKT system to one identical in size and character to (12), but with modifications to the diagonal of matrix $W^{-1}Y$.

Bounds on the variables are handled similarly. The algorithm assumes that each variable x_j has (possibly infinite) upper and lower bounds:

$$l_j \le x_j \le u_j,$$

where

$$\begin{aligned} -\infty &\le l_j < \infty \\ -\infty &< u_j \le \infty. \end{aligned}$$

As in interior-point methods for linear programming, the finite inequalities are converted to equalities by adding slack variables, and the resulting system of equations is again reduced to one of the exact form and size of (12) with the only change being the addition of a diagonal matrix to the $H(x, y)$ block of the reduced KKT matrix. One should note that, in contrast to the lower-bound shift employed by others, we treat lower bounds exactly the same way that we treat upper bounds; such a symmetric treatment of upper and lower bounds was first suggested by Gill, et al., in [14].

The only addition to the discussion of this section is that all nonnegative variables added through either range or bound constraints must be included in the logarithmic barrier term in the merit function and any extra linear equalities are included in the penalty term. Thus the final form of the reduced KKT system solved is

$$\begin{bmatrix} -(H(x, y) + E_n) & A(x)^T \\ A(x) & E_m \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \Phi_1 \\ \Phi_2 \end{bmatrix}, \tag{25}$$

where E_n is an $n \times n$ nonnegative diagonal matrix, E_m is an $m \times m$ positive diagonal matrix and Φ_1 and Φ_2 are appropriately modified right-hand sides; see [25] for explicit expressions. If every variable has either a finite upper bound or a finite lower bound, i.e. if there are no free variables, then E_n too is a positive diagonal matrix.

4.1. Free variables and stabilization. A diagonal element of the nonnegative matrix E_n introduced in the previous section is positive if the corresponding variable has a finite upper or lower bound and it is zero if the variable is a free variable. Therefore as things stand, for linear programming problems having free variables, the reduced KKT matrix is not quasidefinite. To get a quasidefinite matrix even for this case, we write each free variable x_j as a difference of two nonnegative variables:

$$x_j - t_j + g_j = 0,$$
$$t_j, g_j \geq 0.$$

Such a *splitting* of free variables is a common trick but the next step is usually to eliminate them from the problem. We do not do this. Instead, we retain each free x_j and add the constraint shown above. We then do the algebra to reduce the system again to the usual reduced KKT system. The result is exactly the same as before except that now E_n is strictly positive in every component, even those associated with free variables. This improves the conditioning of the reduced KKT system and thereby helps stabilize the accuracy of the factorization. Of course, the new nonnegative variables, t_j and g_j, must be incorporated in the logarithmic barrier term in the merit function and the linear equations in the penalty term.

This implicit handling of free variables ensures that the matrix in (25) is quasidefinite whenever the problem is a convex optimization problem (in particular, whenever it is a linear programming problem). Furthermore, by adding something positive to $H(x, y)$ we increase the likelihood that the system is quasidefinite even when the problem is nonconvex and hence we greatly reduce the number of times that we need to add diagonal perturbations to $H(x, y)$. In fact, on the Hock and Schittkowski test set on which we report in Section 6 LOQO took a total of 2174 Newton iterations with 3298 matrix factorizations for an average of 1.52 factorizations per iteration. As previously stated, our search algorithm for λ is very simplistic and could lead to significantly more factorizations than a more sophisticated version. However, these results seem to verify that the addition of the positive diagonal matrix E_n and the semidefinite matrix $A^T(x)W^{-1}YA(x)$ to $H(x, y)$ greatly reduces the number of perturbations required.

For full details of the computation of E_n and the modification of the right-hand side, as well as full details from the previous section, the reader is referred to [23] and [25].

5. Implementation Details

This section provides implementation details for the nonlinear version of LOQO that we have coded and tested. In particular, we deal with some specific algorithmic choices. In the next section we give the results of computational testing, with comparison to MINOS [20] and LANCELOT [6] on a test set that includes all of the well-known Hock and Schittkowski problems [16] as well as several other large-scale real-world problems.

5.1. Choice of the barrier parameter μ. So far, we have not specified any particular choice for the barrier parameter μ. Most theoretical analyses of interior-point algorithms choose μ as

$$\mu = \gamma \frac{w^T y}{m},$$

where $0 \leq \gamma < 1$. In [7], El Bakry et al. show that for this choice of μ, their algorithm converges to a zero of the merit function (18). Computational experience has shown that

the algorithm performs best when the complementary products $w_i y_i$ approach zero at a uniform rate. We measure the distance from uniformity by computing

$$\xi = \frac{\min_i w_i y_i}{y^T w / m}.$$

Clearly, $0 < \xi \leq 1$ and $\xi = 1$ if and only if $w_i y_i$ is a constant over all values of i. When far from uniformity, a larger μ promotes uniformity for the next iteration. Consequently, we use the following heuristic for our choice of μ, which has proven very effective in practice:

$$\mu = \gamma \min\left((1-r)\frac{1-\xi}{\xi}, 2\right)^3 \frac{w^T y}{m},$$

where $0 < r < 1$ denotes the steplength parameter, which by default is 0.95, and γ is a settable scale factor, which defaults to 0.1.

5.2. The initial point. Nonlinear programming traditionally requires that a starting point be given as part of the problem data, and comparative numerical testing is done using these traditional starting points. For an interior-point code, more is required, however, as both slack variables and split variables are added to the problem. Thus even with initial values for the x_j's, initial values for the w_i's, y_i's etc. must be determined by the program. For slack variables, for instance the w_i's, given an $x^{(0)}$ we can compute $h_i(x^{(0)})$ and set

$$w_i^{(0)} = h_i(x^{(0)}). \tag{26}$$

There are two difficulties with (26). First, if $x^{(0)}$ is not feasible, (26) gives an initial negative value to some of the w_i's, which is not allowed for interior-point methods. Second, even if $x^{(0)}$ is feasible, it may lie on the boundary of the feasible region or so close to it that some of the initial w_i's are very close to zero and progress is hindered. Much as in interior-point methods for linear programming, it is necessary to specify a $\theta > 0$ so that all initial values of variables constrained to be nonnegative are at least as large as θ. Hence w_i is initially set as follows:

$$w_i^{(0)} = \max(h_i(x^{(0)}), \theta). \tag{27}$$

An analogous method can be used to set the slacks on the range constraints used to handle equality constraints, and also the slack variables that transform simple bounds into equalities.

The latter brings up another interesting point. Some of the standard test problems have simple bounds on the variables and give an initial point that violates these bounds. Such an initial point seems unrealistic and can be computationally dangerous, as sometimes bounds on variables are used to ensure that the function value can actually be calculated, such as enforcing $x_j \geq 0$ if $\sqrt{x_j}$ appears somewhere in the calculations. In view of this, we have added an option to the code that can force bounds to be honored. In this case, any initial point that lies outside the bounds is altered to lie inside the bounds and is set exactly θ from the bound if there is only one bound and, if there are both upper and lower bounds, is set to a 90%-10% mixture of the two bounds with the higher value placed on the nearer bound. Finally, the split parts associated with a free variable x_j, t_j and g_j, are set so that the difference is equal to x_j and the smaller of the two is equal to θ.

The remaining point of interest is the choice of θ. For linear programming, $\theta = 100$ has proved computationally efficient. For the nonlinear problems we have tested to date, however, $\theta = 1$ has proved by far the best choice. A good problem-dependent choice for θ remains a topic for further study.

5.3. Other issues. Issues such as stopping rules and matrix reordering heuristics are handled in exactly the same manner as documented in [23].

6. COMPUTATIONAL RESULTS

The LOQO implementation described in this paper, in addition to its ability to read industry-standard MPS files for expressing linear and quadratic programming problems, is interfaced with two of the most popular mathematical programming languages, AMPL [11] and GAMS [4]. Currently, only AMPL can provide a solver with second-order information, i.e. Hessians, and therefore all of our testing was performed using AMPL together with LOQO. Many other solvers are also interfaced with AMPL, including MINOS [20] and LANCELOT [6]. For the tests described here, we used LOQO version 3.10 (19971027), MINOS version 5.4 (19940910), and LANCELOT version 20/03/1997. Both MINOS and LANCELOT were run using default options although we did have to increase the number of superbasics and the maximum number of major/minor iterations on some of the larger problems.

All tests were performed on an R4600 SGI workstation with 160 MBytes of real memory, 16 Kbytes of data cache, and a 133 MHz clock. There are three components to the stopping rule for LOQO: (i) primal feasibility, (ii) dual feasibility, and (iii) lack of duality gap. The default rule declares a solution primal/dual feasible if the relative infeasibility is less than $1.0e-6$ and declares the problem optimal if, in addition, there are 8 or more digits of agreement between the primal and dual objective function values.

6.1. Hock and Schittkowski suite. Table 1 shows the solution times in seconds for the Hock and Schittkowski set of test problems [16]. Table 2 shows how many interior-point iterations LOQO used to solve each problem. Each solver was run with default parameters. Since LOQO is still under development some of the defaults might change in the future. Therefore, we list here those parameters that might change and give the current default settings (for nonlinear problems):

- **`mufactor`**, called γ above, is set to 0.1.
- **`bndpush`**, called θ above, is set to 100.
- **`honor_bnds`**, a boolean flag indicating whether bounds on variables are to be enforced throughout the iteration process as opposed to only at optimality, is set to 1 (i.e., true).

All of the problems in the Hock and Schittkowski set are very small. For linear programming, the simplex method is usually more efficient than interior-point methods on small problems. One would expect the same for nonlinear programming. As expected, MINOS, which is simplex-based, was faster on a majority of the problems. Nonetheless, we were pleasantly surprised to see how well LOQO did compared with MINOS. Since all of the times are small, direct time comparisons have limited value and so we have summarized in Table 3 how many times each solver came in first, second, and third place. Finally, we computed the total time for LOQO and MINOS on problems that both codes solved to optimality. The total time for LOQO was 21.9 seconds whereas for MINOS it was 28.1 seconds. It should be noted here that MINOS uses only first derivatives. It is possible that a true Newton variant of MINOS might improve speed. However, our experience to date is that for small dense problems coded in AMPL the cost of evaluating second derivatives is significant and may well offset any other improved algorithmic efficiency. It is our understanding that the default LANCELOT uses exact second derivatives.

	Time in Seconds				Time in Seconds				Time in Seconds		
Name	Minos	Lancelot	Loqo	Name	Minos	Lancelot	Loqo	Name	Minos	Lancelot	Loqo
hs001	0.02	0.11	0.06	hs040	0.01	0.04	0.03	hs080	0.04	0.06	0.04
hs002	0.00	0.04	0.04	hs041	0.00	0.04	0.04	hs081	0.05	0.07	0.07
hs003	0.00	0.05	0.03	hs042	0.01	0.04	0.03	hs083	0.01	0.07	0.05
hs004	0.00	0.05	0.03	hs043	0.05	0.08	0.03	hs084	0.03	(3)	0.06
hs005	0.01	0.04	0.02	hs044	0.00	0.05	0.03	hs085	0.49	(7)	1.19
hs006	0.08	0.11	0.05	hs045	0.00	0.01	0.09	hs086	0.01	0.14	0.06
hs007	0.08	0.06	0.03	hs046	0.09	0.08	0.05	hs087	0.05	0.23	0.08
hs008	0.01	0.04	0.03	hs047	0.10	0.08	0.10	hs088	0.50	(3)	1.06
hs009	0.00	0.06	0.03	hs048	0.01	0.02	0.04	hs089	1.02	(3)	2.88
hs010	0.04	0.06	0.03	hs049	0.02	0.08	0.06	hs090	(6)	(3)	1.34
hs011	0.03	0.05	0.03	hs050	0.01	0.04	0.04	hs091	(4)	10.36	1.98
hs012	0.03	0.07	0.03	hs051	0.00	0.04	0.06	hs092	9.74	23.84	1.93
hs013	0.03	0.11	(4)	hs052	0.00	0.03	0.03	hs093	0.06	(1)	0.04
hs014	0.01	0.05	0.03	hs053	0.00	0.03	0.03	hs095	0.01	0.16	0.06
hs015	0.01	0.11	0.06	hs054	0.01	0.04	0.06	hs096	0.01	0.17	0.07
hs016	0.01	0.08	0.04	hs055	0.00	0.03	0.04	hs097	0.09	0.13	0.06
hs017	0.01	0.07	0.08	hs056	0.05	0.04	0.04	hs098	0.04	0.13	0.06
hs018	0.13	0.39	0.04	hs057	0.04	0.08	0.09	hs099	0.05	(4)	0.14
hs019	0.04	0.19	0.06	hs059	0.09	0.83	0.04	hs100	0.11	0.25	0.03
hs020	0.01	0.08	0.05	hs060	0.09	0.05	0.05	hs101	2.03	(4)	0.61
hs021	0.00	0.05	0.04	hs061	0.04	0.05	0.03	hs102	2.04	(4)	0.26
hs022	0.02	0.04	0.02	hs062	0.01	0.09	0.03	hs103	4.48	(4)	0.24
hs023	0.04	0.13	0.04	hs063	0.12	0.05	0.03	hs104	0.32	(1)	0.05
hs024	0.00	0.05	0.04	hs064	0.09	0.09	0.05	hs105	2.59	(4)	6.78
hs025	0.01	0.06	0.24	hs065	0.06	0.17	0.04	hs106	0.17	(4)	0.11
hs026	0.07	0.09	0.04	hs066	0.02	0.04	0.04	hs107	0.04	(4)	0.25
hs027	0.09	0.07	0.20	hs067	(5)	(5)	(5)	hs108	0.13	0.33	0.12
hs028	0.00	0.03	0.03	hs068	(2)	(2)	0.12	hs109	0.30	(4)	0.35
hs029	0.04	0.05	0.03	hs069	(2)	(2)	0.04	hs110	0.02	0.06	0.05
hs030	0.03	0.05	0.03	hs070	0.17	4.73	0.42	hs111	0.41	0.34	0.16
hs031	0.02	0.04	0.03	hs071	0.04	0.06	0.03	hs112	0.05	0.29	0.06
hs032	0.01	0.03	0.06	hs072	0.07	(3)	0.05	hs113	0.11	0.91	0.05
hs033	0.01	0.04	0.05	hs073	0.02	0.07	0.05	hs114	0.09	3.43	0.15
hs034	0.05	0.05	0.03	hs074	0.03	0.07	0.04	hs116	0.37	(4)	0.18
hs035	0.00	0.04	0.03	hs075	0.02	(3)	0.04	hs117	0.13	0.55	0.10
hs036	0.00	0.04	0.03	hs076	0.01	0.05	0.03	hs118	0.03	0.23	0.07
hs037	0.01	0.04	0.03	hs077	0.10	0.08	0.04	hs119	0.04	0.29	0.30
hs038	0.03	0.12	0.09	hs078	0.04	0.05	0.03				
hs039	0.06	0.05	0.04	hs079	0.04	0.06	0.03				

TABLE 1. Solution times on Hock-Schittkowski problems. Legend: (1) could not find a feasible solution, (2) erf() not available, (3) step got too small, (4) too many iterations, (5) could not code model in AMPL, (6) unbounded or badly scaled, (7) core dump.

In order to interpret the results of Table 1, we note first that hs067 was not run with any algorithm. This problem contains an internal fixed-point calculation which is difficult to code in AMPL. LOQO failed to converge for only one case of the remaining test set, hs013. This is a classic problem where the optimal point does not satisfy the KKT conditions. As LOQO currently only looks for KKT points, it was unable to solve this problem. Identifying optima that do not satisfy the KKT conditions as well as correctly determining unboundedness and infeasibility remain topics for further study.

Another point of interest is the relatively high iteration count and solution time for LOQO on hs027. These can be dramatically reduced by increasing β early in the iteration sequence, but all methods we have tried to date that significantly help with hs027 hurt enough other problems so as to be less efficient overall. Thus an optimal means of choosing β, which one hopes would be part of a provably globally convergent algorithm, also remains for further study.

Name	Iters	Name	Iters	Name	Iters	Name	Iters	Name	Iters
hs001	32	hs025	15	hs048	15	hs073	20	hs098	19
hs002	19	hs026	15	hs049	24	hs074	13	hs099	20
hs003	11	hs027	55	hs050	16	hs075	15	hs100	11
hs004	10	hs028	11	hs051	18	hs076	11	hs101	40
hs005	10	hs029	10	hs052	10	hs077	13	hs102	24
hs006	17	hs030	11	hs053	13	hs078	9	hs103	24
hs007	10	hs031	9	hs054	18	hs079	9	hs104	14
hs008	9	hs032	25	hs055	13	hs080	11	hs105	21
hs009	10	hs033	17	hs056	12	hs081	19	hs106	33
hs010	15	hs034	14	hs057	19	hs083	15	hs107	56
hs011	12	hs035	11	hs059	17	hs084	18	hs108	23
hs012	10	hs036	14	hs060	18	hs085	48	hs109	49
hs014	11	hs037	11	hs061	11	hs086	15	hs110	12
hs015	33	hs038	44	hs062	14	hs087	25	hs111	17
hs016	20	hs039	15	hs063	10	hs088	25	hs112	17
hs017	36	hs040	9	hs064	28	hs089	34	hs113	16
hs018	17	hs041	17	hs065	14	hs090	25	hs114	31
hs019	31	hs042	9	hs066	13	hs091	29	hs116	33
hs020	24	hs043	11	hs068	41	hs092	27	hs117	22
hs021	16	hs044	11	hs069	15	hs093	10	hs118	17
hs022	9	hs045	38	hs070	19	hs095	18	hs119	33
hs023	16	hs046	20	hs071	12	hs096	22		
hs024	16	hs047	37	hs072	21	hs097	18		

TABLE 2. Iteration counts on Hock-Schittkowski problems.

Solver	1st	2nd	3rd
MINOS	72	31	9
LANCELOT	4	30	78
LOQO	44	55	13

TABLE 3. Rankings

Nonconvex problems often have alternative optima, which is certainly the case with some problems in the Hock and Schittkowski test set. In our experiments, LOQO found alternative optima that were worse than the reported optima on 8 problems (hs002, hs016, hs020, hs025, hs059, hs070, hs097, hs098) and better on 7 problems (hs047, hs088, hs089, hs090, hs091, hs092, hs109). While finding optima that are worse is generally to be expected, finding optima that are better is quite a surprise because those reported by Hock and Schittkowski represent the best optima known from years of experimentation with the test set. For the suboptimal problems the desired optimal solution could generally be obtained by altering θ, thereby changing the initial solution.

In addition to the default value for θ, we tried two other values. For $\theta = 0.1$, LOQO fails on two problems and is noticeably less efficient on others. For $\theta = 10$, it fails on seven problems and is again noticeably less efficient on others.

Name	m	n	nonz(A)	nonz(H)	LOQO	MINOS	LANCELOT
antenna	167	49	8014	2303	1m1s	*2m34s	*19m41s
fekete2	50	150	150	22500	1m18s	24s	*6m 1s
markowitz2	201	1200	201200	200	4m43s	1m56s	13m35s
minsurf	172	1849	172	12601	21s	*13m37s	2m7s
polygon2	195	42	766	880	0.7s	*60m0s	1m1s
sawpath	198	5	784	25	2s	5s	*8s
structure4	720	1536	5724	20356	2m40s	*43m6.89s	**
trafequil2	628	1194	5512	76	5.1s	5.3s	2m39s

TABLE 4. Preliminary computational results for several application areas. An asterisk indicates that the solution obtained was either infeasible or suboptimal. The double asterisk indicates not enough memory to solve the problem.

We also did a run in which we did not force variables to remain within their bounds; i.e., **`honor_bnds=0`**. On this run, three problems in addition to hs013 failed to converge. We did not find this very surprising for interior-point methods, where we feel that bounds should be honored from the beginning.

As a final note, the iteration counts in Table 2 represent the total number of times first and second partial derivatives were computed. A few extra function evaluations were used in reducing the merit function, but as the time taken to calculate them is very small compared to other components of the algorithm, we chose to report execution times rather than function counts.

6.2. Large-scale real-world problems. We are assembling a collection of real-world problems encoded in AMPL. The collection is available from the first author's web site [22]. Table 4 gives a brief summary of problem statistics and computational results for some of the larger problems that we have encountered. Most of these problems are quite difficult. As the table shows, LOQO can be a very efficient and robust code on large difficult problems when compared to the other solvers. We made naive attempts to set parameters to appropriate non-default values in all codes. Clearly, we were better able to adjust parameters in LOQO than in the other codes. Note that we set the time limit for each run at one hour. Our inability to run LANCELOT due to memory limitations on **`structure4`**, is the result of not knowing how to adjust the memory estimates made by AMPL when invoking LANCELOT.

As these problems have not been previously part of any standard test set, we give here a brief description of some of the interesting ones. Non-default LOQO parameters, when used, are noted below with the problem descriptions.

6.2.1. *Antenna Array Synthesis (antenna).* An important problem in electrical engineering is to determine how to combine the signals from an array of antennas so as to reinforce the signal from a desired direction and suppress the signal from undesired ones. There are various formulations of this problem, some of which fall under convex optimization. The details of one such formulation are given in [17]. To solve this problem with LOQO, we set parameters as follows: **`sigfig=5, bndpush=2, convex, inftol=1.0e-1`**.

6.2.2. *Electrons on a Sphere (fekete2).* Given n electrons placed on a conducting sphere, the problem is to find a distribution of these electrons that minimizes the total Coulomb

potential. This problem is nonconvex and has a large number of local minima. Problem fekete2 was solved with $m = 50$.

6.2.3. *Portfolio optimization (markowitz2).* The Markowitz model in portfolio optimization seeks the portfolio of investments that optimizes a linear combination of expected return and expected risk. Risk is modeled as the variance of the return. Problem markowitz2 is a separable convex quadratic formulation of the problem with linear constraints. Expected reward and risk are computed using historical data.

6.2.4. *Minimal Surfaces (minsurf).* Given a domain in $\mathbb{R}^2$ and boundary data, the minimal surface problem is to find an interpolation of the boundary data into the interior of the domain so that the surface so generated has minimal surface area. It is a convex optimization problem.

6.2.5. *Largest Small Polygon (polygon2).* Given n, the problem is to find the n-sided polygon of maximal area whose diameter does not exceed 1. This problem sounds trivial, but Graham showed in [15] that the optimal hexagon is not the regular hexagon. To solve polygon2 with LOQO, we set parameters as follows: **convex**.

6.2.6. *Saw Path Tracking (sawpath).* Given a list of points describing the center line of a wood piece, the problem is to fit the best polynomial that minimizes the sum of the squares of errors subject to three sets of side constraints: (1) the polynomial must go through the first point, (2) the polynomial must have a specified initial slope, and (3) the radius of curvature must never exceed a given value. This problem comes from F. Grondin of Forintek Canada Corp.

6.2.7. *Structural Optimization (structure4).* In structural optimization, the problem is to decide how to build a structure to minimize weight or compliance, subject to the constraint that the structure can support one or more possible loadings. There are various models, some of which are convex while others are not. Also, some of these models are for pin-jointed truss-like structures while others are based on finite-element models. Problem structure4 is to find an optimal bracket design obtained using a convex finite-element model. The corresponding truss-like model is described in [3].

6.2.8. *Traffic Equilibrium (trafequil2).* The problem is to find flows through a network that minimize a nonlinear function, called the Beckman objective.

6.3. Mittelmann's quadratic programming set. As a service to the Operations Research community, Hans Mittelmann has done extensive testing of optimization software. We summarize here some of his results [19]. He used an AMPL model to generate a number of random, feasible, quadratic programming problems. These problems are determined by specifying values for five parameters, **n**, **m**, **p**, **s**, and **pf**; see [19] for their definitions.

He used two versions of the random model generator; one that generates convex problems and one that does not. The results for the convex problems are shown in Table 5 and for the nonconvex problems the results are shown in Table 6.

The first table shows how efficient interior-point methods are compared to other methods on convex quadratic programming problems. Regarding the second table in which problems were nonconvex, inevitably the different algorithms found different optima. We were also encouraged by the fact that LOQO successfully solved six of the nine problems efficiently. However, LOQO did fail on three of the problems. The reason is that the algorithm eventually took small steps and therefore failed to make sufficient progress before the iteration limit was reached. No merit reductions to the steplength were incurred and

n	m	p	s	pf	MINOS	LANCELOT	LOQO
100	20	200	.1	.1	7	29	1
100	20	2000	.1	.01	15	36	2
100	20	200	1.	.1	34	28	4
250	50	500	.1	.1	380	109	13
250	50	500	.5	.1	693	295	27
250	50	500	.1	.3	293	98	14
500	100	1000	.1	.1	4304	811	85
500	0	1000	.1	.1	5850	1506	86
500	100	0	.1	.1	2345	273	49
1000	200	500	.02	.1	5247	1434	147

TABLE 5. Mittelmann's results for convex quadratic programming. Numbers shown under the solvers are solution times in seconds. For LOQO, the following parameter settings were used: bndpush=100 honor_bnds=0 pred_corr=1 mufactor=0. (These are the LP defaults. In the next release of LOQO, these values will be the defaults for QPs too.)

n	m	p	s	pf	MINOS	LANCELOT	LOQO
100	20	200	.1	.1	1	*	1
100	20	2000	.1	.01	5	63	9
100	20	200	1.	.1	8	*	17
250	50	500	.1	.1	41	*	*
250	50	500	.5	.1	160	*	56
250	50	500	.1	.3	37	*	19
500	100	1000	.1	.1	542	*	172
500	0	1000	.1	.1	602	*	*
500	200	1000	.1	.1	462	1389	*
1000	200	2000	.02	.1	7106	*	1311

TABLE 6. Mittelmann's results for nonconvex quadratic programming. Numbers shown under the solvers are solution times in seconds. An asterisk indicates that the solution obtained was either infeasible or suboptimal. For LOQO, the following parameter settings were used: bndpush=100 honor_bnds=0 pred_corr=1 mufactor=0.

so the short steps must have arisen from some variables being very close to their bounds. Future research will address identifying and remedying this phenomenon.

7. CONCLUSIONS

The aim of our work has been to take an existing interior–point code for quadratic programming, namely LOQO, and modify it as little as possible to develop an efficient code for nonconvex nonlinear programming. We found that essentially only two changes were needed:

(1) A *merit function* to ensure proper steplength control.
(2) *Diagonal pertubation* to the Hessian matrix to ensure that the search directions are descent directions when the problem is not convex.

As noted throughout the paper, the algorithm documented herein represents our first attempt at an interior-point code for nonconvex nonlinear programming problems. We believe that significant improvements can and will be made over time. However, we personally were both gratified and somewhat surprised at how robust and how efficient this initial code has proved to be.

8. ACKNOWLEDGEMENTS

We'd like to thank David Gay for doing much of the hard work by providing the automatic differentiation algorithms in AMPL to compute gradients and Hessians.

9. TRIBUTE

The second author of this paper has known Olvi Mangasarian for approximately thirty years. I believe we first met at the first of the four Madison conferences on mathematical programming held in May, 1970. All four conferences, organized first by Olvi, Ben Rosen and Klaus Ritter and later by Olvi, Bob Meyer and Steve Robinson, in my opinion contributed greatly to the development of mathematical programming during a period of considerable excitement and great progress. The excitement was perhaps greater than Olvi might have wished at the first conference, where tear gas pervaded the campus, the National Guard was omnipresent, and we had to move the sessions off campus to avoid the antiwar protests. Since these early days, Olvi and I have remained close friends, meeting regularly at conferences, where our wives have shared many happy days visiting the art museums while we attended sessions (and often wished we were with our wives). I have been around long enough to have taught from Olvi's book when it was new, and have followed all of his seminal work from the theorems of the alternative through the work of his student Shi Ping Han on SQP to his work on proximal point algorithms, complementarity, and most recently on detecting breast cancer, to name just a few of his accomplishments. Through all of this he has been a great friend, congenial host, and valued companion. The first author is young, and new to nonlinear programming. This is a wonderful way for him to make the acquaintance of one of the icons of our field.

REFERENCES

[1] A.R.Conn, N. Gould, and Ph.L. Toint. A globally convergent Lagrangian barrier algorithm for optimization with general inequality constraints and simple bounds. *Math. of Computation*, 66:261–288, 1997.

[2] A.R.Conn, N. Gould, and Ph.L. Toint. A primal–dual algorithm for minimizing a non–convex function subject to bound and linear equality constraints. Technical Report Report 96/9, Dept of Mathematics, FUNDP, Namur (B), 1997.

[3] M.P. Bendsøe, A. Ben-Tal, and J. Zowe. Optimization methods for truss geometry and topology design. *Structural Optimization*, 7:141–159, 1994.

[4] A. Brooke, D. Kendrick, and A. Meeraus. *GAMS: A User's Guide*. Scientific Press, 1988.

[5] R.H. Byrd, M.E. Hribar, and J. Nocedal. An interior point algorithm for large scale nonlinear programming. Technical Report OTC 97/05, Optimization Technology Center, Northwestern University, 1997.

[6] A.R. Conn, N.I.M. Gould, and Ph.L. Toint. *LANCELOT: a Fortran Package for Large-Scale Nonlinear Optimization (Release A)*. Springer Verlag, Heidelberg, New York, 1992.

[7] A. El-Bakry, R. Tapia, T. Tsuchiya, and Y. Zhang. On the formulation and theory of the Newton interior-point method for nonlinear programming. *J. of Optimization Theory and Appl.*, 89:507–541, 1996.

[8] A.V. Fiacco and G.P. McCormick. *Nonlinear Programming: Sequential Unconstrainted Minimization Techniques*. Research Analysis Corporation, McLean Virginia, 1968. Republished in 1990 by SIAM, Philadelphia.

[9] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. Technical Report NA/171, University of Dundee, Dept. of Mathematics, Dundee, Scotland, 1997.

[10] A. Forsgren and P.E. Gill. Primal-dual interior methods for nonconvex nonlinear programming. Technical Report NA 96-3, Department of Mathematics, University of California, San Diego, 1996.
[11] R. Fourer, D.M. Gay, and B.W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Scientific Press, 1993.
[12] R.S. Gajulapalli and L.S. Lasdon. Computational experience with a safeguarded barrier algorithm for sparse nonlinear programming. Technical report, University Texas at Austin, 1997.
[13] D.M. Gay, M. L. Overton, and M.H. Wright. A primal-dual interior method for nonconvex nonlinear programming. In *Proceedings of the 1996 International Conference on Nonlinear Programming*. Kluwer, 1998. To appear.
[14] P.E. Gill, W. Murray, D.B. Ponceleón, and M.A. Saunders. Solving reduced KKT systems in barrier methods for linear and quadratic programming. Technical Report SOL 91-7, Systems Optimization Laboratory, Stanford University, Stanford, CA, July 1991.
[15] R.L. Graham. The largest small hexagon. *Journal of Combinatorial Theory*, 18:165–170, 1975.
[16] W. Hock and K. Schittkowski. *Test examples for nonlinear programming codes*. Lecture Notes in Economics and Mathematical Systems 187. Springer Verlag, Heidelberg, 1981.
[17] H. Lebret and S. Boyd. Antenna array pattern synthesis via convex optimization. *IEEE Transactions on Signal Processing*, 45:526–532, 1997.
[18] I.J. Lustig, R.E. Marsten, and D.F. Shanno. Interior point methods for linear programming: computational state of the art. *ORSA J. on Computing*, 6:1–14, 1994.
[19] H. Mittelmann. Benchmarks for optimization software. http://plato.la.asu.edu/bench.html.
[20] B.A. Murtagh and M.A. Saunders. MINOS 5.4 user's guide. Technical Report SOL 83-20R, Systems Optimization Laboratory, Stanford University, 1983. Revised February 1995.
[21] D.F. Shanno and E.M. Simantiraki. Interior-point methods for linear and nonlinear programming. In I.S. Duff and G.A. Watson, editors, *The State of the Art in Numerical Analysis*, pages 339–362. Oxford University Press, New York, 1997.
[22] R.J. Vanderbei. Large-scale nonlinear AMPL models. http://www.sor.princeton.edu/~rvdb/ampl/nlmodels/.
[23] R.J. Vanderbei. LOQO: An interior point code for quadratic programming. Technical Report SOR 94-15, Princeton University, 1994.
[24] R.J. Vanderbei. Symmetric quasi-definite matrices. *SIAM Journal on Optimization*, 5(1):100–113, 1995.
[25] R.J. Vanderbei. *Linear Programming: Foundations and Extensions*. Kluwer Academic Publishers, 1996.

ROBERT J. VANDERBEI, PRINCETON UNIVERSITY, PRINCETON, NJ

DAVID F. SHANNO, RUTGERS UNIVERSITY, NEW BRUNSWICK, NJ