INTERIOR-POINT METHODS FOR NONCONVEX NONLINEAR PROGRAMMING: JAMMING AND COMPARATIVE NUMERICAL TESTING

HANDE Y. BENSON, DAVID F. SHANNO, AND ROBERT J. VANDERBEI

Operations Research and Financial Engineering
Princeton University
ORFE-00-02

Revised August 28, 2000

ABSTRACT. The paper considers a current example of Wächter and Biegler which is shown not to converge for the standard primal-dual interior-point method for nonlinear programming. Three possible solutions to the problem are derived and discussed, including shifting slack variables, a trust region method, and a modified barrier method. The paper then compares LOQO, a line-search interior-point code, with SNOPT, a sequential-quadratic-programming code, and NITRO, a trust-region interior-point code on a large test set of nonlinear programming problems. Specific types of problems which can cause LOQO to fail are identified.

1. INTRODUCTION

In two recent papers, [18],[14], a primal-dual interior-point algorithm for solving nonconvex nonlinear programming problems is described, together with a limited amount of computational experience with LOQO, the computer implementation of the algorithm. This algorithm is primarily a line search algorithm, although it does occasionally perturb the diagonal of the Hessian matrix to ensure that the search direction is a descent direction for the merit function, a property employed by many pure trust region methods. In the preliminary testing reported in [18],[14], the algorithm showed promise of being both an efficient and robust code for general nonconvex nonlinear programming.

Date: August 28, 2000.

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

Research of the first and third authors supported by NSF grant DMS-9870317, ONR grant N00014-98-1-0036. Research of the second author supported by NSF grant DMS-9805495.

Subsequent to this work, Wächter and Biegler [19] provided an example for which primal-dual interior-point methods fail to converge, and the iterates are bounded away from the optimum. In [11], Nocedal and Marazzi further examined this nonconvergence issue and proposed a general trust region method as a resolution. In this paper, we show that classical jamming is why the Wächter and Biegler example fails to converge and show three possible resolutions of this issue, including the one currently implemented in LOQO.

The second, and major, purpose of this paper is to test LOQO on a much larger suite of problems, including problems with many thousands of variables and constraints, in order to ascertain how efficient and robust the algorithm is. For comparison purposes, the algorithm is tested against NITRO [5], a trust-region interior-point code, and SNOPT [9], an active-set, quasi-Newton-based, sequential-quadratic-programming code. The results of the comparative testing are documented in a subsequent section, and show that LOQO is a very robust and efficient code.

As with all nonlinear programming algorithms, however, LOQO does not solve all of the problems in the test set. We have analyzed those problems which LOQO fails to solve and have determined six distinct types of problems that can cause trouble for LOQO. We include examples of each type from the test set in the interest of encouraging research in how best to detect and deal with problems of these types when using LOQO or a similar algorithm. In addition, to successfully solve some of the problems in the test set, a few LOQO parameters must be changed from the default values. This information is also included, for the difficulty being addressed is in general applicable to a wide class of algorithms.

2. An Infeasible Interior-Point Method

We begin with a basic description of the LOQO algorithm, since it is required to understand the analysis of the following sections, as well as the discussion of numerical results. The basic problem we consider is

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

where $f(x)$ and the $h_i(x)$'s are assumed to be twice continuously differentiable and x is an n-vector of free (i.e., unbounded) variables. The problem is *convex* if f is convex and each of the h_i's is concave.

Adding slacks, (1) becomes

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

where $h(x)$ and w are vectors representing the $h_i(x)$'s and w_i's, respectively. The inequality constraints are then eliminated by incorporating them in a logarithmic barrier term in the objective function, transforming (2) to

$$\text{minimize } f(x) - \mu \sum_{i=1}^{m} \log(w_i) \tag{3}$$

$$\text{subject to } h(x) - w = 0.$$

Denoting the Lagrange multipliers for the system (3) by y, the first order conditions for the problem are

$$\begin{aligned} \nabla f(x) - \nabla h(x)^T y &= 0 \\ -\mu W^{-1} e + y &= 0 \\ h(x) - w &= 0 \end{aligned} \tag{4}$$

where W is the diagonal matrix with the w_i's as diagonal elements, e is the vector of all ones, and $\nabla h(x)$ is the Jacobian matrix of the vector $h(x)$. The primal-dual system is obtained from (4) by multiplying the second equation by W, giving the system

$$\begin{aligned} \nabla f(x) - \nabla h(x)^T y &= 0 \\ -\mu e + WYe &= 0 \\ h(x) - w &= 0 \end{aligned} \tag{5}$$

where again Y is the diagonal matrix with the y_i's on the diagonal.

Newton's method is employed to iterate to a triple (x, w, y) satisfying (5). Denoting

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

and

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

the Newton system for (5) is then

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

This system is not symmetric, but can be symmetrized by negating the first equation and multiplying the second by $-W^{-1}$, giving the system

$$\begin{bmatrix} -H(x,y) & 0 & A(x)^T \\ 0 & -W^{-1}Y & -I \\ A(x) & -I & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta w \\ \Delta y \end{bmatrix} = \begin{bmatrix} \sigma \\ -\gamma \\ \rho \end{bmatrix}, \tag{6}$$

where

$$\begin{aligned} \sigma &= \nabla f - A(x)^T y, \\ \gamma &= \mu W^{-1} e - y, \\ \rho &= w - h(x). \end{aligned}$$

Solving the second equation of (6) for Δw gives

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

which when substituted into the third equation gives the *reduced KKT system*:

$$\begin{bmatrix} -H(x,y) & A(x)^T \\ A(x)^T & WY^{-1} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta w \end{bmatrix} = \begin{bmatrix} \sigma \\ \rho + WY^{-1}\gamma \end{bmatrix}. \tag{7}$$

Denoting $H(x,y)$ and $A(x)$ by H and A, respectively, LOQO generally solves a modification of (7) in which λI is added to H to ensure that $H + A^T W^{-1} T A + \lambda I$ is positive definite (see [14]). With this change, we get that

$$\begin{aligned} \Delta x &= (N + \lambda I)^{-1}(A^T(W^{-1}Y\rho + \gamma) - \sigma), \\ \Delta y &= W^{-1}Y\rho + \gamma - W^{-1}Y\Delta x, \end{aligned} \tag{8}$$

where

$$N = H + A^T W^{-1} Y A$$

is called the *dual normal matrix*.

LOQO then proceeds to a new estimate of the optimum by

$$\begin{aligned} x^{(k+1)} &= x^{(k)} + \alpha^{(k)}\Delta x^{(k)}, \\ w^{(k+1)} &= w^{(k)} + \alpha^{(k)}\Delta w^{(k)}, \\ y^{(k+1)} &= y^{(k)} + \alpha^{(k)}\Delta y^{(k)}, \end{aligned} \tag{9}$$

where $\alpha^{(k)}$ is chosen to ensure both that $w^{(k+1)} > 0$ and that the merit function

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

is sufficiently reduced.

This completes our description of the algorithm implemented in LOQO. This algorithm is called an *infeasible interior-point method*. The modifier "infeasible" refers to the fact that primal feasibility is not required by the algorithm at the beginning (and then enforced throughout). Instead, it is only achieved as one approaches an optimal solution. The modifier "interior-point" refers to the fact that the slack variables are required to be strictly positive at the beginning (and then throughout). As has been documented many places (see e.g. [10]), infeasible interior-point methods represent the state-of-the-art in linear programming.

There are two salient points concerning LOQO that we feel may not be well understood, and both are important in the context of this paper. The first is that the decision variables x are always free variables—only slack variables have bounds. The second deals with how more general constraints than simple inequalities are treated. For example, to see how equality constraints are dealt with in LOQO, we first consider a range constraint of the form

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

This is easily converted to a system of constraints

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

It is shown in [17] how to solve the resulting system without increasing the size of the matrix factored in solving (8). LOQO treats all equality constraints as range constraints with $r_i = 0$. Treating equality constraints as inequalities in this way and decision variables as free variables gives a natural symmetry to the algorithm which will prove important in analyzing the Wächter-Biegler example.

We end this section by noting that explicit formulae can be given for the step directions Δw and Δy. Since we shall need the explicit formula for Δw in the next section, we record it here:

$$\begin{aligned} \Delta w &= -A\tilde{N}^{-1}\nabla f(x) \\ &\quad - \left(I - A\tilde{N}^{-1}A^T W^{-1} Y\right)\rho \\ &\quad + \mu A\tilde{N}^{-1}A^T W^{-1} e, \end{aligned}$$

where

$$\tilde{N} = N + \lambda I$$

denotes the *perturbed dual normal matrix*.

3. An Anti-Jamming Theorem for Nondegenerate Problems

Before introducing the Wächter-Biegler example, we present in this section a theoretical result that gives conditions under which jamming will not occur. An infeasible interior-point algorithm may jam on a certain problem instance if there exists a point on the boundary of the nonnegative orthant (in slack variable space) at which the search direction is well-defined and does not point back into the interior of the nonnegative orthant. In this section, we show that except for some minor nondegeneracy conditions the interior-point method presented in the previous section does indeed point inwards.

To state our result, we need to introduce some notation. In particular, it is convenient to let

$$\tilde{H} = H + \lambda I.$$

We consider a point $(\bar{x}, \bar{w}, \bar{y})$ satisfying the following conditions:

(1) Nonnegativity: $\bar{w} \geq 0$, $\bar{y} \geq 0$.
(2) Strict complementary: $\bar{w} + \bar{y} > 0$.

Of course, we are interested in a point where some of the $\bar{w}_i$'s vanish. Let $\mathcal{U}$ denote the set of constraint indices for which $\bar{w}_i$ vanishes and let $\mathcal{B}$ denote those for which it doesn't.

Write A (and other matrices and vectors) in block form according to this partition

$$A = \begin{bmatrix} B \\ U \end{bmatrix}.$$

Matrix A and all other quantities are functions of the current point. We use the same letter with a bar over it to denote the value of these objects at the point $(\bar{x}, \bar{w}, \bar{y})$.

Theorem 1. *If the point $(\bar{x}, \bar{w}, \bar{y})$ satisfies conditions* (1) *and* (2)*, and $\bar{U}$ has full row rank, then Δw has a continuous extension to this point and $\Delta w_{\mathcal{U}} = \mu Y_{\mathcal{U}}^{-1} e_{\mathcal{U}} > 0$ there.*

Proof. From the formula for Δw given in the previous section, we see that

$$\begin{aligned} \Delta w_{\mathcal{U}} + \rho_{\mathcal{U}} &= -U\tilde{N}^{-1}\nabla f(x) \\ &\quad + U\tilde{N}^{-1}A^T W^{-1}\left(Y\rho + \mu e\right). \end{aligned} \tag{10}$$

To prove the theorem, we must analyze the limiting behavior of $U\tilde{N}^{-1}$ and $U\tilde{N}^{-1}A^TW^{-1}$.

Let

$$K = \tilde{N} - U^T W_{\mathcal{U}}^{-1} Y_{\mathcal{U}} U = \tilde{H} + B^T W_{\mathcal{B}}^{-1} Y_{\mathcal{B}} B.$$

Applying the Sherman–Morrison–Woodbury formula, we get

$$\begin{aligned} \tilde{N}^{-1} &= (U^T W_{\mathcal{U}}^{-1} Y_{\mathcal{U}} U + K)^{-1} \\ &= K^{-1} - K^{-1}U^T\left(UK^{-1}U^T + W_{\mathcal{U}}Y_{\mathcal{U}}^{-1}\right)^{-1} UK^{-1}. \end{aligned}$$

Hence,

$$\begin{aligned} U\tilde{N}^{-1} &= UK^{-1} - UK^{-1}U^T\left(UK^{-1}U^T + W_{\mathcal{U}}Y_{\mathcal{U}}^{-1}\right)^{-1} UK^{-1} \\ &= W_{\mathcal{U}}Y_{\mathcal{U}}^{-1}\left(UK^{-1}U^T + W_{\mathcal{U}}Y_{\mathcal{U}}^{-1}\right)^{-1} UK^{-1}. \end{aligned} \tag{11}$$

From the definition of $\mathcal{U}$, we have that $\bar{W}_{\mathcal{U}} = 0$. In addition assumption (2) implies that $\bar{Y}_{\mathcal{U}} > 0$, which then implies that $Y_{\mathcal{U}}^{-1}$ remains bounded in a neighborhood of $(\bar{x}, \bar{w}, \bar{y})$. Also, $\bar{K}$ is positive definite since λ is chosen to make $\tilde{H}$ positive definite. Hence, $\bar{K}$ is nonsingular. The assumption that $\bar{U}$ has full row rank therefore implies that the following limit exists:

$$\lim\left(UK^{-1}U^T + W_{\mathcal{U}}Y_{\mathcal{U}}^{-1}\right)^{-1} UK^{-1} = \left(\bar{U}\bar{K}^{-1}\bar{U}^T\right)^{-1}\bar{U}\bar{K}^{-1}.$$

Here and throughout this proof, all limits are understood to be taken as (x, w, y) approaches $(\bar{x}, \bar{w}, \bar{y})$. From the previous limit, we see that

$$\lim U\tilde{N}^{-1} = 0.$$

It is easy to check that assumptions (1)–(2) imply that the terms multiplied by $U\tilde{N}^{-1}$ on the first line of (10) remain bounded in the limit and therefore

$$\lim -U\tilde{N}^{-1}\nabla f(x) = 0. \tag{12}$$

Now, consider $U\tilde{N}^{-1}A^TW^{-1}$. Writing A and W in block form and using (11), we get

$$U\tilde{N}^{-1}A^TW^{-1} = W_{\mathcal{U}}Y_{\mathcal{U}}^{-1}\left(UK^{-1}U^T + W_{\mathcal{U}}Y_{\mathcal{U}}^{-1}\right)^{-1} UK^{-1}\begin{bmatrix} B^TW_{\mathcal{B}}^{-1} & U^TW_{\mathcal{U}}^{-1} \end{bmatrix}.$$

The analysis of $U\tilde{N}^{-1}$ in the previous paragraph applies to the first block of this block matrix and shows that it vanishes in the limit. The analysis of the second block is more delicate because of the $W_{\mathcal{U}}^{-1}$ factor:

$$\begin{aligned} W_{\mathcal{U}}Y_{\mathcal{U}}^{-1}\left(UK^{-1}U^T+\ W_{\mathcal{U}}Y_{\mathcal{U}}^{-1}\right)^{-1}UK^{-1}U^TW_{\mathcal{U}}^{-1} & \\ &= W_{\mathcal{U}}Y_{\mathcal{U}}^{-1}\left(I-\left(UK^{-1}U^T+W_{\mathcal{U}}Y_{\mathcal{U}}^{-1}\right)^{-1}W_{\mathcal{U}}Y_{\mathcal{U}}^{-1}\right)W_{\mathcal{U}}^{-1} \\ &= \left(I-W_{\mathcal{U}}Y_{\mathcal{U}}^{-1}\left(UK^{-1}U^T+W_{\mathcal{U}}Y_{\mathcal{U}}^{-1}\right)^{-1}\right)Y_{\mathcal{U}}^{-1} \\ &= UK^{-1}U^T\left(UK^{-1}U^T+W_{\mathcal{U}}Y_{\mathcal{U}}^{-1}\right)^{-1}Y_{\mathcal{U}}^{-1} \end{aligned}$$

From this last expression, we see that the limiting value for the second block is just $Y_{\mathcal{U}}^{-1}$. Putting the two blocks together, we get that

$$\lim U\tilde{N}^{-1}A^TW^{-1} = \left[\begin{array}{cc} 0 & \bar{Y}_{\mathcal{U}}^{-1} \end{array}\right]$$

and hence that

$$\lim U\tilde{N}^{-1}A^TW^{-1}\left(Y\rho+\mu e\right) = \bar{\rho}_{\mathcal{U}}+\mu\bar{Y}_{\mathcal{U}}^{-1}e_{\mathcal{U}}. \tag{13}$$

Combining (12) and (13), we see that

$$\lim \Delta w_{\mathcal{U}} = \mu\bar{Y}_{\mathcal{U}}^{-1}e_{\mathcal{U}}.$$

□

It follows from Theorem 1 that the interior-point algorithm should not jam provided that U has full rank and the sequence stays away from boundary points where complimentary pairs of variables both vanish. For linear and quadratic programming, the sequence of iterates does indeed obey this strict complementarity property (see, e.g., [1]) so one would expect it to hold for NLP too, although we don't prove it here.

In the next section, we give an example where jamming does indeed occur. We analyze this example in the context of the theorem just presented.

4. THE WÄCHTER-BIEGLER EXAMPLE

Wächter and Biegler considered the problem

$$\text{minimize } x_1 \tag{14}$$

$$\begin{aligned} \text{subject to } x_1^2-x_2+a &= 0, \\ x_1-x_3-b &= 0, \\ x_2,x_3 &\geq 0. \end{aligned}$$

They show that if $b \geq 0$, $a-mb \leq \min(0,\frac{-a}{2})$, and $x_1^{(0)} < 0$, $x_2^{(0)}, x_3^{(0)} > 0$, where

$$m = \frac{(x_1^{(0)})^2-x_2^{(0)}+a}{\left|x_1^{(0)}-x_3^{(0)}-b\right|},$$

then the sequence of points generated by certain interior-point algorithms remain bounded away from the optimum value. As stated, the problem doesn't fit the paradigm studied in the previous section. However, it is easy to recognize variables x_2 and x_3 as slack variables and hence the problem, in its simplest form, is:

$$\begin{aligned} &\text{minimize } x_1 \\ &\text{subject to } x_1^2 \geq -a, \\ &\qquad\qquad x_1 \geq b. \end{aligned} \tag{15}$$

The basic interior-point method of the previous section applied to this problem gets stuck at a point where both slack variables vanish. Hence, the matrix U of the previous section is a 2×1 matrix and cannot have full row rank. We continue this section with an analysis of this example in its original form.

The original form of the Wächter and Biegler example involves a mix of free and nonnegative variables. Let $I \subset \{1, 2, \ldots, n\}$ denote the set of indices corresponding to the nonnegative variables. In what follows, we will denote by x_I the subvector of x consisting of the nonnegative variables. Similar notations will be employed where needed. With these notations, the general problem form considered by Wächter and Biegler is

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } h(x) = 0, \\ &\qquad\qquad x_I \geq 0. \end{aligned} \tag{16}$$

The logarithmic barrier formulation of (16) is

$$\begin{aligned} &\text{minimize } f(x) - \mu \sum_{i \in I} \log(x_i) \\ &\text{subject to } h(x) = 0. \end{aligned} \tag{17}$$

The first-order conditions for (17) are

$$\begin{aligned} \nabla f(x) - \nabla h(x)^T y - \begin{bmatrix} v_I \\ 0 \end{bmatrix} &= 0, \\ \mu X_I^{-1} e - v_I &= 0, \\ -h(x) &= 0, \end{aligned} \tag{18}$$

where X_I is the $|I| \times |I|$ diagonal matrix containing the vector x_I, and v_I is a vector of dual slacks associated with x_I. The primal-dual form of (18) is

$$\begin{aligned} \nabla f(x) - \nabla h(x)^T y - \begin{bmatrix} v_I \\ 0 \end{bmatrix} &= 0, \\ X_I V_I e - \mu e &= 0, \\ -h(x) &= 0. \end{aligned} \tag{19}$$

The Newton system for (19) is

$$\text{(20)}\quad \begin{bmatrix} H(x,y) & -\begin{bmatrix} I \\ 0 \end{bmatrix} & -A(x)^T \\ \begin{bmatrix} V_I & 0 \end{bmatrix} & X_I & 0 \\ -A(x) & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta v_I \\ \Delta y \end{bmatrix} = \begin{bmatrix} -\nabla f(x) + A(x)^T y + \begin{bmatrix} v_I \\ 0 \end{bmatrix} \\ \mu e - X_I V_I e \\ h(x) \end{bmatrix}.$$

Using the middle equation to solve for Δv_I,

$$\Delta v_I = X_I^{-1}\left(\mu e - X_I V_I e - V_I \Delta x_I\right),$$

and eliminating it from the block system, we arrive at the *reduced KKT system*:

$$\begin{bmatrix} H(x,y) + \begin{bmatrix} V_I X_I^{-1} & 0 \end{bmatrix} & -A(x)^T \\ -A(x) & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} -\nabla f(x) + A(x)^T y + \begin{bmatrix} \mu X_I^{-1} e \\ 0 \end{bmatrix} \\ h(x) \end{bmatrix}.$$

Using these search directions, the algorithm can be summarized as follows:

$$\text{(21)}\quad \begin{aligned} x^{(k+1)} &= x^{(k)} + \alpha^{(k)} \Delta x^{(k)}, \\ v_I^{(k+1)} &= v_I^{(k)} + \alpha^{(k)} \Delta v_I^{(k)}, \\ y^{(k+1)} &= y^{(k)} + \alpha^{(k)} \Delta y^{(k)}. \end{aligned}$$

El Bakry et. al. [6] have studied the algorithm defined by (20) and (21) and show it converges to a local minimizer under certain conditions. One of the conditions is that the coefficient matrix of the system of equations appearing in (20) remain nonsingular. This is precisely the condition violated by this example. Examination of the coefficient matrix of the system (20) shows that a sufficient condition for its nonsingularity is that $H(x,y)$ be positive definite, $A(x)$ have full row rank, and that X_I be strictly positive. We ran the example in the form (15) using LOQO with $a = -1$, $b = 1$, and $x^{(0)} = -2$. This problem has the solution 1. After 10 iterations, a previous version of LOQO jams at the point $x = -1.127$ with the two slacks both less than 10^{-9}. Here, the matrix is essentially singular. To see precisely what happens, after standard algebraic manipulation, the system

(5) applied to problem (14) reduces to the system

$$\left(\frac{2x_1^2}{y_1} - \frac{x_2}{v_2}\right)\Delta y_1 + \frac{x_1}{y_1}\Delta y_2 = r_1, \tag{22}$$

$$\frac{x_1}{y_1}\Delta y_1 + \left(\frac{1}{2y_1} - \frac{x_3}{v_3}\right)\Delta y_2 = r_2, \tag{23}$$

with

$$\begin{aligned} r_1 &= -x_1^2 - x_2 - 1 - \frac{x_1 y_2}{y_1} - \frac{x_1}{y_1} - \frac{\mu}{v_2} + \frac{x_2}{v_2}y_1, \\ r_2 &= -x_3 + 1 - \frac{y_2}{2y_1} + \frac{1}{2y_1} - \frac{\mu}{v_3} + \frac{x_3}{v_3}y_2. \end{aligned}$$

Substituting $x_2 = 0$, $x_3 = 0$ into (22) and (23), we obtain a coefficient matrix that is singular. However, r_1 and r_2 are nonzero. What often happens is that the nearly singular matrix obtained with very small, but nonzero values for x_2 and x_3 lead to larger potential step sizes in all the variables, which forces α to become tiny to ensure that the nonnegative variables remain positive. Thus, the algorithm becomes stuck. This phenomenon is a type of *jamming*. It has been previously noted in Simantiraki and Shanno [15].

5. Remedies for Jamming.

We have identified three possible remedies for jamming. To implement the first of these remedies, however, it is necessary to restate the problem in a form closer to the one used by LOQO. For LOQO, the problem (16) would be restated as

$$\text{minimize } f(x)$$

$$\begin{aligned} \text{subject to } h(x) - w &= 0, \\ w + p &= 0, \\ x_I - g_I &= 0, \\ w, p, g_I &\geq 0. \end{aligned}$$

The Lagrangian for this problem is

$$\begin{aligned} L(x, w, p, g, y, q, z) = {} & f(x) - y^T(h(x) - w) + q^T(w + p) - z_I^T(x_I - g_I) \\ & -\mu \sum_{i=1}^{m} \log(w_i) - \mu \sum_{i=1}^{m} \log(p_i) - \mu \sum_{i \in I} \log(g_i) \end{aligned}$$

The first order conditions in primal-dual form are

$$\begin{aligned} \nabla f(x) - \nabla h(x)^T y - \begin{bmatrix} z_I \\ 0 \end{bmatrix} &= 0, \\ WYe + WQe - \mu e &= 0, \\ PQe - \mu e &= 0, \\ G_I Z_I e - \mu e &= 0, \\ -h(x) + w &= 0, \\ w + p &= 0, \\ -x_I + g_I &= 0. \end{aligned} \tag{24}$$

If we now define

$$\begin{aligned} \Phi(\Delta x, \Delta w, \Delta p, \Delta g_I) &= \frac{1}{2}\Delta x^T H(x,y)\Delta x + \frac{1}{2}\Delta w^T W^{-1}(Y+Q)\Delta w \\ &+ \frac{1}{2}\Delta p^T P^{-1} Q \Delta p + \frac{1}{2}\Delta g_I^T G_I^{-1} Z_I \Delta g_I \\ &+ \Delta x^T \left(\nabla f - A(x)^T y - \begin{bmatrix} z_I \\ 0 \end{bmatrix} \right) \\ &+ \Delta w^T (y + q - \mu W^{-1} e) \\ &+ \Delta p^T (q - \mu P^{-1} e) + \Delta g_I^T (z_I - \mu G_I^{-1} e) \end{aligned}$$

then with some simple algebraic manipulation it can be shown that the Newton direction for (24) is the solution to

$$\text{minimize } \Phi(\Delta x, \Delta w, \Delta p, \Delta g_I)$$

$$\begin{aligned} \text{subject to } A(x)\Delta x - \Delta w &= -h(x) + w, \\ \Delta w + \Delta p &= 0, \\ \Delta x_I - \Delta g_I &= 0, \end{aligned} \tag{25}$$

where the Lagrange multiplier updates Δy, Δq, and Δz_I are the multipliers associated with the constraints of (25). This is the standard sequential quadratic programming formulation for the search directions. If, as previously, we solve explicitly for the search directions associated with each of the slack variables, then the resulting reduced KKT matrix is given by

$$\begin{bmatrix} -H(x,y) - \begin{bmatrix} G_I^{-1} Z_I & \\ & 0 \end{bmatrix} & A(x)^T \\ A(x) & (P^{-1}Q + W^{-1}Y + W^{-1}Q) \end{bmatrix}, \tag{26}$$

Here, as above, analysis shows that if sufficiently many of the w_i's, p_i's, and g_i's become zero, the coefficient matrix again becomes singular. One way to resolve this issue, perhaps the simplest and the one used in the current LOQO, is just to shift small values of w_i, p_i,

and/or g_i to larger values, thus restoring nonsingularity of the reduced KKT matrix. Note that this is impossible using the formulation of (20), for, in that case, the small variables are decision variables x_i and arbitrarily shifting them will completely invalidate the algorithm. Here, however, the w_i's, p_i's, and g_i's are slack variables and interior-point methods can shift slacks without seriously impacting the progress of the algorithm. Indeed, this is common practice in interior-point codes for linear programming [10].

In the current LOQO, a slack variable, say w_i, is shifted whenever it is smaller than 10^{-4} and $\Delta w_i/w_i > 10^4$. All such slack variables are shifted by the same amount s which is given by the harmonic average of the values of the variables that are not being shifted:

$$s = \frac{k}{\sum_i \frac{1}{w_i} + \sum_i \frac{1}{p_i} + \sum_i \frac{1}{g_i}},$$

where k is the total number of terms contained in the sum. With this shift, LOQO solves the Wächter-Biegler problem (15) in 21 iterations.

The above approach has the virtue of simplicity and succeeds in solving the Wächter-Biegler problem. There is also a simple geometric explanation of how it works. Figure 1 shows the feasible set and the iterations both with and without shifting. The horizontal axis is the decision variable x and the vertical axis is the slack variable for the constraint $x^2 - 1 \geq 0$. The feasible region therefore is that portion of the parabola lying to the right of $x = 1$. Without shifting the iterates jam at approximately $(-1, 0)$. The new algorithm shifts just once, by a large amount on the seventh iteration, and then moves quickly to the optimal solution. The effect of shifting is to translate the linearization of the $x^2 - 1 \geq 0$ constraint so that a much larger step can be taken in the next iteration without the slack becoming negative.

We also describe two other potential resolutions to the jamming phenomenon. The first is a *trust-region method*. Here the problem (25) is modified to

$$\text{minimize } \Phi(\Delta x, \Delta w, \Delta p, \Delta g_I)$$

$$\begin{aligned}
\text{subject to } A(x)\Delta x - \Delta w &= -h(x) + w, \\
\Delta w + \Delta p &= 0, \\
\Delta x_I - \Delta g_I &= 0, \\
\|W^{-1}\Delta w\|_2^2 &\leq r_1, \\
\|P^{-1}\Delta p\|_2^2 &\leq r_2, \\
\|G_I^{-1}\Delta g_I\|_2^2 &\leq r_3.
\end{aligned}$$

Unfortunately, the trust-region constraints may make this subproblem infeasible. When the subproblem is feasible, its solution agrees with the solution to a diagonal perturbation of the reduced KKT system (26). However, the perturbation it produces might not yield the important property of making the upper left block in (26) into a negative definite matrix. In Byrd, et.al., [5], this issue is dealt with by including a trust-region constraint on the vector x of decision variables. Rather than attempting to implement these trust-region ideas, we

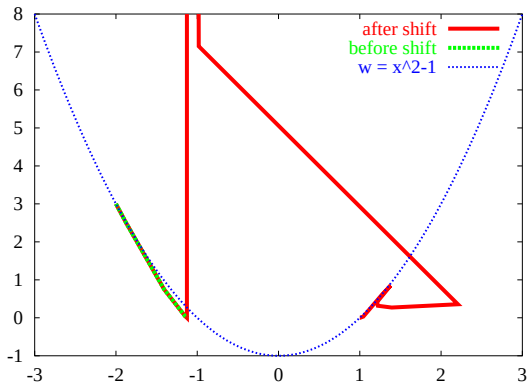

FIGURE 1. The Wachter example. The horizontal axis represents the variable x and the vertical axis represents the slack variable in the constraint $x^2 - 1 \geq 0$. The feasible region is that portion of the parabola lying to the right of $x = 1$. The starting point for the algorithm is $(-2, 3)$. Without shifting the algorithm jams near $(-1, 0)$.

chose the simple diagonal-perturbation method described in a previous section and we compare the resulting algorithm (LOQO) to a true trust-region algorithm as represented by NITRO.

A third way of dealing with the jamming issue is to use a modified-barrier method (Polyak [12]). This is most easily explained considering the problem in the form (16). Here the barrier problem (17) is modified to

$$\text{minimize } f(x) - \mu \sum_{i \in I} \log(1 + \frac{x_i}{\mu})$$

$$\text{subject to } h(x) = 0.$$

The first-order conditions, in primal-dual form, are

$$\begin{aligned} \nabla f(x) - \nabla h(x)^T y - \begin{bmatrix} v_I \\ 0 \end{bmatrix} &= 0, \\ V_I(X_I + \mu I)e - \mu e &= 0, \\ h(x) &= 0. \end{aligned} \tag{27}$$

The Newton system for (27) is

$$\begin{bmatrix} H(x,y) & -A(x)^T & -\begin{bmatrix} I \\ 0 \end{bmatrix} \\ A(x) & 0 & 0 \\ \begin{bmatrix} V_I & 0 \end{bmatrix} & 0 & X_I + \mu I \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta v \end{bmatrix} = \begin{bmatrix} -\nabla f(x) + A(x)^T y + \begin{bmatrix} v_I \\ 0 \end{bmatrix} \\ -h(x) \\ \mu e - V_I(X_I + \mu I)e \end{bmatrix}.$$

Here, if $H(x,y)$ is positive definite and $A(x)$ has full row rank, the system is nonsingular for all nonnegative x as long as $\mu > 0$. In fact, this actually allows for potentially jammed x_i's to temporarily become negative to alleviate jamming. Polyak analysed an algorithm based on these step directions. In his analysis, it is essential to assume that μ decreases monotonically. Because LOQO does not monotonically reduce μ, we have not tried this approach, but it may prove interesting as an alternative to the first two approaches discussed in this section, and causes none of the complications of the trust-region method.

6. NUMERICAL RESULTS

Interior-point methods were originally developed for nonlinear programming problems, [7], but quickly fell into disfavor due to numerous numerical difficulties. The more recent success of interior-point methods for linear and quadratic programming led to renewed interest in interior-point methods for nonconvex nonlinear programming, and several codes have been under extensive development for the last several years. While the successful interior-point codes for linear programming were all line-search codes, as is LOQO, significant effort has gone into trust-region codes for nonconvex nonlinear programming. One such code is NUOPT by Yamashita, Tanabe and Yabe [20], and in their report they cite impressive numerical performance. Another such code is Byrd, Hribar, and Nocedal's NITRO [5] alluded to previously. One goal of this work was to try to determine the relative efficiency and robustness of line-search versus trust-region methods on a large enough test set to be able to draw some reasonable conclusions.

As any code using Newton's method requires second partial derivatives, we have chosen to formulate the models in AMPL [8], a modelling language that provides analytic first and second partial derivatives. In order to construct a meaningful test suite, we have been engaged in reformulating from *standard input format* (SIF) to AMPL all models in the CUTE [4] (constrained and unconstrained testing environment) test suite. To date, we have converted and validated 699 models. For those problems with variable size, we have used the largest reported number of variables and constraints, except in the case of the `ncvxqp` family of problems and `fminsurf`, where the largest possible size were beyond the capabilities of the solvers. In addition, we have expressed the entire Schittkowski [13] test suite in AMPL. Together, this comprises a test suite of 889 AMPL models, which form

the test set for this paper. These models vary greatly in size and difficulty and have proved useful in drawing meaningful conclusions. All of the AMPL models used in our testing are available at [16].

The CUTE suite contains some problems that are excluded from our set. We have not yet converted to AMPL any models requiring special functions as wellas some of the more complex models. We will continue to convert the remainder of the suite to AMPL models as time allows, but believe that the results of this section show that the current test suite is sufficiently rich to provide meaningful information.

As we know of no AMPL interface to the NUOPT code of [20], we have chosen to compare LOQO (5.04, 20000713) to NITRO (20000727) to attempt to assess the relative merits of the line-search and trust-region variants of an interior-point code. In addition, to compare the relative merits of interior-point codes with a more traditional sequential-quadratic-programming code, we have also included a comparison with SNOPT (5.3-5(3) Jul 2000), a state-of-the-art code due to Gill, Murray, and Saunders [9]. All three codes were run from AMPL version 20000814.

SNOPT is different from the tested interior-point codes in many important ways. It uses only first derivatives, estimating the Hessian of the Lagrangian using a limited-memory quasi-Newton method. It also is an active set method, using at each iteration an estimate for the set of constraints active at the optimum point rather than all constaints, active or inactive, as interior-point methods do. Thus the method provides a true contrast to interior-point methods.

Beyond simply testing comparatively, we are also interested in identifying those models where interior-point methods fail and for those models we attempt to determine characteristics that lead to failure. We have identified a number of these characteristics, which we will discuss in detail subsequently.

Also, most codes for nonlinear programming have user defined parameters which can be used to tune the algorithm to obtain a successful result on an otherwise difficult problem. This is certainly true of LOQO. As we felt unqualified to tune either SNOPT or NITRO, we chose to run and compare all three codes on the default settings for the user-defined parameters with the small exceptions:

- In LOQO, `iterlim` was increased from 200 to 500 to make it more comparable with the defaults in SNOPT and NITRO.
- In SNOPT, `superbasics_limit` was increased from 50 to 50000.
- All three codes were run with `timlim` set to 60 minutes.

We then report separately the results on the additional models that we were able to solve with LOQO after tuning. For each of these models, we indicate which parameters are adjusted to a nondefault value, which we feel is instructive in using an interior-point code. Summary results appear in the column labeled LOQO-tuned in Table 1.

All 889 models were run on a SUN SPARC Station running SunOS 5.8 with 4GB of main memory and a 400MHz clock speed. As the detailed results are too voluminous to include here, we have chosen to aggregate them in Table 1 and to display them graphically.

Size	Probs		SNOPT	NITRO	LOQO	LOQO tuned
Small	585	Solved	558	516	555	17
n+m $<$ 100		Total Time	30.11	192.76	72.59	2.61
Medium	101	Solved	79	73	81	10
n+m $<$ 1000		Total Time	741.57	277.09	562.04	214.00
Large	109	Solved	73	98	96	8
n+m $<$ 10000		Total Time	37593.77	7160.60	8451.12	129.56
Very Large	94	Solved	30	18	56	30
10000 $\leq$ n+m		Total Time	13753.40	4245.10	5284.50	7853.76

TABLE 1. Number of problems solved and total running times by solver

	small	medium	large	very large
LOQO vs SNOPT	1.58	1.03	0.14	0.17
LOQO vs NITRO	0.66	0.71	0.91	0.91

TABLE 2. Pairwise comparisons. Numbers represent the geometric average of the ratios of the LOQO solution time to the other solver solution time averaged over those problems that are solved by both LOQO and the other solver.

Full tabular results are available at [2]. In the table, we have aggregated the problems based on problem size. We note here that NITRO has a limit of 10,000 variables, so that of the 94 problems categorized as very large, NITRO was able to attempt only 30.

For each of SNOPT and NITRO, we give in Table 2 the geometric average of the ratios of the LOQO solution time to the other solver solution time averaged over those problems that are solved by both LOQO and the other solver. We use the geometric mean so that a ratio of two and a ratio of one-half are treated symmetrically. To be clear, a number less than one indicates that LOQO outperformed the other solver on average.

For the small problem column, we have eliminated problems where one of the solvers had a solution time reported as 0 seconds. If a timer with a precision better than the hundredths of a second were available, the results would favor SNOPT even more on the small problems.

Furthermore, the ratios reflect solution time but not the accuracy of the solution. In almost all cases, LOQO and SNOPT obtain comparably accurate solutions whereas NITRO finds slightly less accurate ones.

Examination of the results in the table on small problems shows little difference between LOQO and SNOPT, in terms of robustness, and SNOPT is more efficient than LOQO on these problems, as expected. NITRO, however, is significantly slower and far less robust. The reasons for this remain to be determined, especially in view of the apparently efficient

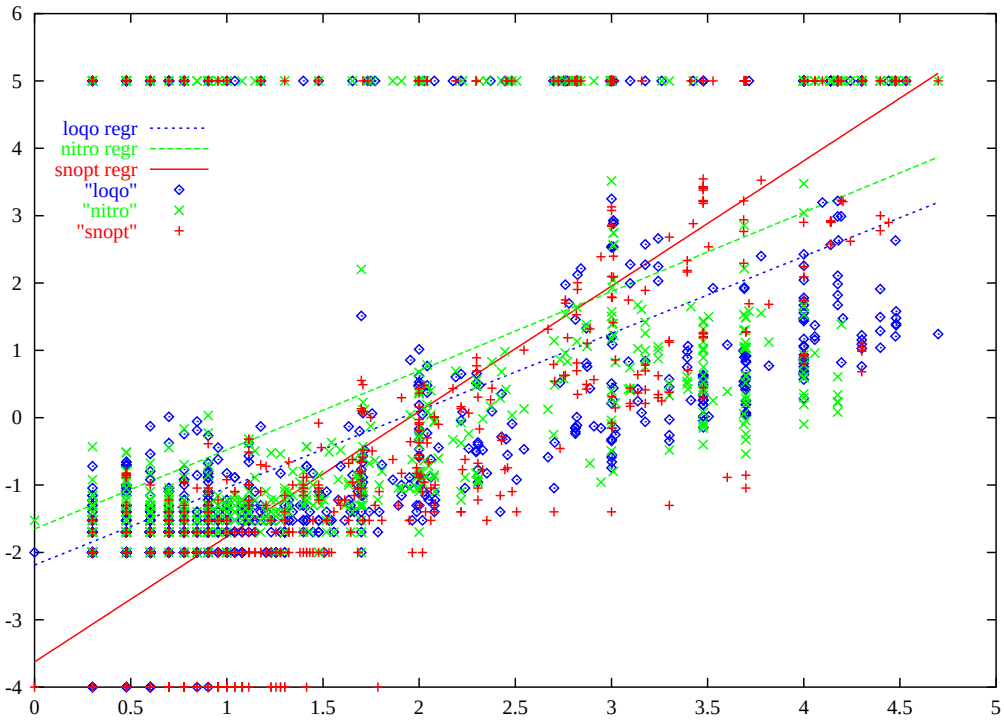

FIGURE 2. A log-log plot of solution times vs. number of variables. The logs are base 10. A solution time of 10^5 is used for problems that failed to solve. Failures are included in the L^2-regression computations, which are also shown.

and robust performance of NUOPT [20]. LOQO and SNOPT are equally robust and efficient on the medium sized problems. NITRO is less efficient and less robust on the medium sized problems. On the large problems, NITRO and LOQO are equally robust and efficient, but SNOPT is much slower and less robust. This appears to be due almost entirely to the use of a limited-memory quasi-Newton method by SNOPT, which is less efficient than a true Newton method which exploits sparsity in the Hessian. This difference, especially in terms of robustness, with SNOPT becomes dramatically greater as problem size increases. Here, most of the failures of SNOPT are due to iteration limit or time limit violations. LOQO is also far more robust than NITRO on the very large problems.

Two immediate conclusions can be drawn from the tables. The first is that a quasi-Newton-based sequential-quadratic-programming (SQP) method is very satisfactory for small to medium sized problems, but a sparsity exploiting method for estimating and factoring the Hessian is essential for large problems. Furthermore, while the SQP method is satisfactory for the stated classes, it has little apparent advantage over LOQO. The second conclusion is that a line search interior-point algorithm exhibits excellent robustness, as well as efficiency, compared to a trust-region method.

6.1. **Failure Modes.** As encouraging as the results are, it is apparent that LOQO does not solve all of the test problems. In view of this, we tried to determine problem characteristics that will cause LOQO trouble. We have identified six specific problems that will cause difficulties for any interior-point algorithm. These will now be discussed briefly.

(1) *Failure to satisfy a constraint qualification condition*. Newton-based interior-point methods search for a KKT point. If no such point exists at the optimum, the algorithm must fail. This was the case with the models `hs013` and `gausselm` from the CUTE test suite and `s221` from the Schittkowski suite.

(2) *An unbounded face of optimal solutions*. Interior-point methods go to the center of the face of optimal solutions. If this set is unbounded, the method may not converge. This was the case with the models `manne` from the CUTE suite and `s333` from the Schittkowski suite.

(3) *Infeasible or unbounded solution*. Until the homogeneous method [21] was developed for linear programming, determining infeasibility and unboundedness was very ad hoc for interior-point codes for LP. As we know of no analogue to the homogeneous method for nonconvex nonlinear programming, determination remains ad hoc for interior-point methods. The unbounded problems are `s340` from the Schittkowski suite and `fletcbv3`, `fletchbv`, `indef`, and `static3` from the CUTE suite. The problem `lewispol` from the CUTE suite is infeasible. LOQO sometimes concludes that a problem appears unbounded or dual unbounded, but often reaches the iteration limit without being able to state decisively that this is the case.

(4) *Nondifferentiability*. The theory of Newton-based interior-point methods depends on the assumption that the objective function and all constraints are twice continuously differentiable. Not surprisingly, when this is not the case, the algorithm may fail. This occurs with the models `s332` and `s391` from the Schittkowski suite, and models `broydn7d` and `core2` from the CUTE suite. While nondifferentiability is sometimes built into models designed to break algorithms, in real models it is often the result of poor modelling, and can be eliminated with good technique. For example, in `s332`, the problem contains the expressions

$$\begin{aligned} y_i &= \arctan\left(\left|\frac{1/t_i - x_1}{\ln t_i + x_2}\right|\right), \\ p &= \max y_i, \\ p &\leq 30. \end{aligned}$$

Since $\arctan|x| = |\arctan x|$, the above system is equivalent to

$$-30 \leq \arctan\left(\frac{1/t_i - x_1}{\ln t_i + x_2}\right) \leq 30.$$

This latter constraint is clearly differentiable, while the original system is not. All three codes failed on the problem as initially posed, but when restated all solved the problem easily, SNOPT in 0.14 seconds and 8 iterations, NITRO in 0.16 seconds

brainpc*	bratu1d	eigenb2	himmelp5
huestis	loghairy	mdhole	ncvxbqp*
ncvxqp3	ncvxqp6	oet2	optprloc
orthrege	orthrgdm	qrtquad	s238
s347	s355	s361	s374
s376	s380	snake	steenbrb
steenbrg			

TABLE 3. Models that required a nondefault value for `bndpush`. Here an asterisk indicates a collection of similar problems. The last four models are from the Schittkowski set—the rest come from the CUTE set.

and 23 iterations, and LOQO in 0.11 seconds and 30 iterations. Thus, if the true reason for building a model is to obtain a solution, it is important to eliminate nondifferentiablity wherever possible.

(5) *Bad scaling*. This is the case with problem `launch` from the CUTE suite and `s210` from the Schittkowski suite.

(6) *Underdetermined nonlinear regression models.* All algorithms fail to solve `heart6`, `heartls`, `palmer5a`, and `palmer7a`. Each of these is a nonlinear least squares regression model. In each case, the data is insufficient to determine all of the regression coefficients.

In addition to these quantifiable errors, LOQO failed to solve some problems for which we currently have no explanation: `cresc50`, `cresc132`, `eigenbco`, `dixchlnv`, `genrose`, `minc44`, `qpnboei1`, `steenbrc`, and `steenbrf`.

6.2. **Tuning.** In addition to LOQO's failures, as previously mentioned, it had to be tuned on some models. There are four ways in which this tuning was done. We examine each briefly.

(1) *Setting the initial values of the slack variables.* As described in the first two sections, LOQO introduces slack variables into the models. While all test problems come with initial estimates of the decision variables, no information on setting of slacks is provided. The default version of LOQO for nonconvex nonlinear programming sets these initial values all to 1. A user defined parameter, `bndpush`, allows one to alter the default setting. These problems are shown in Table 3. For the exact settings of `bndpush`, see [3].

(2) *Setting convergence parameters.* The default convergence parameters for LOQO are relative primal and dual infeasibilities of 10^{-6} and eight digits of agreement between the primal and dual objective functions. For some nonlinear problems, accuracy at this level is numerically unattainable. The problems for which it is necessary to relax the convergence criterion are shown in Table 4. Again details are available at [3].

bloweya	bloweyb	bloweyc	dittert
brainpc2	brainpc3	brainpc5	brainpc6
brainpc7	brainpc9	hager3	haifam
kissing	liswet1	liswet2	liswet7
liswet8	liswet9	meyer3	orthrds2
palmer7e	probpenl	qrtquad	smmpsf
steenbrb	steenbrg	trainh	

TABLE 4. Models that required relaxation of the stopping criteria. All these models come from the CUTE set.

There is one more model, `bigbank`, which LOQO was unable to solve in its original formulation without setting the infeasibility tolerance to 10^{-4}. However, LOQO's difficulty with this problem was easily traced to the presence of upper and lower bounds of $\pm 10^9$ on some of the variables. These bounds are not tight, as determined by the other solvers, and therefore probably represent the modelers desire to express them as free variables. We took the liberty to remove these bounds from this model after which LOQO solved the problem easily. Wildly out-of-scale bounds will generally cause problems for interior-point methods as all of the data is used in every iteration.

(3) *Increasing the iteration limit.* Doing so allowed LOQO to solve `orthrds2`, `ncvxbqp1`, and `ncvxbqp2`.

(4) *Asserting problem convexity.* If a problem is declared to be convex, using the `convex` parameter, no diagonal perturbations are added to the Hessian, even if a diagonal element of the wrong sign appears. Asserting convexity is necessary to solve `hager1`, `hager2`, `hager4`, `reading1`, `s255 spiral`, `ubh1`, and `ubh5`. This is interesting, for it means that adding a diagonal perturbation parameter to the Hessian matrix can destabilize the algorithm under some circumstances, which may contribute, in some degree, to NITRO's instability.

6.3. **Final Remarks.** Of the 889 problems shifting takes place 19 times. Fourteen of these problems solved without shifting using default parameters. Of the remaining five, three can be solved without shifting but by changing the default starting value of the slack variables. This leaves two problems where jamming was an issue and shifting resolved it.

Comparative numerical testing on a set of problems this large is never easy. Occasionally, algorithms report incorrectly that an optimum has been found. This occurred when SNOPT decided that the starting point was optimal for several of the `liswet` problems. This is almost certainly due to the numerical difficulty of these problems.

Also, different algorithms find different local optima. For example, SNOPT found 49 local optima, NITRO 51, and LOQO 39, where a local optimum is defined to be a solution which is not as good as the best known. This is inevitable, and says little about the algorithms.

Despite these difficulties, we feel that the results of this testing strongly indicate that a Newton-based line-search interior-point algorithm can be a very stable and efficient method for nonconvex nonlinear programming, and at least to us, no other algorithm seems more attractive at this point in time.

Acknowledgement. We would like to thank Jorge Nocedal and Philip Gill for providing us with their NITRO and SNOPT codes, respectively, for use in the comparative testing.

REFERENCES

[1] I. Adler and R.D.C. Monteiro. Limiting behavior of the affine scaling continuous trajectories for linear programming. *Mathematical Programming*, 50:29–51, 1991. 7

[2] H.Y. Benson, D.F. Shanno, and R.J. Vanderbei. Benchmarks comparing LOQO with SNOPT and NITRO on most problems in the cute/schittkowski test set. http://www.princeton.edu/∼rvdb/cute_table.pdf. 16

[3] H.Y. Benson, D.F. Shanno, and R.J. Vanderbei. LOQO parameter tuning on problems in the cute/schittkowski test set. http://www.princeton.edu/∼rvdb/tuned_table.pdf. 19

[4] I. Bongartz, A.R. Conn, N. Gould, and Ph.L. Toint. Constrained and unconstrained testing environment. http://www.cse.clrc.ac.uk/Activity/CUTE+74. 14

[5] R.H. Byrd, M.E. Hribar, and J. Nocedal. An Interior Point Algorithm for Large Scale Nonlinear Programming. *SIAM J. Opt.*, 9(4):877–900, 1999. 2, 12, 14

[6] 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. 9

[7] 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. 14

[8] R. Fourer, D.M. Gay, and B.W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Scientific Press, 1993. 14

[9] P.E. Gill, W. Murray, and M.A. Saunders. User's guide for SNOPT 5.3: A Fortran package for large-scale nonlinear programming. Technical report, Systems Optimization Laboratory, Stanford University, Stanford, CA, 1997. 2, 15

[10] 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. 4, 12

[11] M. Marazzi and J. Nocedal. Feasibility control in nonlinear optimization. Technical Report OTC 2000/04, Optimization Technology Center, Northwestern University, March 2000. 2

[12] R.A. Polyak. Modified barrier functions (theory and methods). *Mathematical Programming*, 54:177–222, 1992. 13

[13] K. Schittkowski. *More Test Samples for Nonlinear Programming codes*. Springer Verlag, Berlin-Heidelberg-New York, 1987. 14

[14] D.F. Shanno and R.J. Vanderbei. Interior-Point Methods for Nonconvex Nonlinear Programming: Orderings and Higher-Order Methods. *Math. Prog.*, 87(2):303–316, 2000. 1, 4

[15] E.M. Simantiraki and D.F. Shanno. An infeasible-interior-point method for linear complimentarity problems. *SIAM J. Optimization*, 7:620–640, 1997. 10

[16] R.J. Vanderbei. AMPL models. http://www.sor.princeton.edu/˜rvdb/ampl/nlmodels. 15

[17] R.J. Vanderbei. LOQO: An interior point code for quadratic programming. *Optimization Methods and Software*, 12:451–484, 1999. 5

[18] R.J. Vanderbei and D.F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13:231–252, 1999. 1

[19] A. Wächter and L. Biegler. Failure of global convergence for a class of interior point methods for nonlinear programming. Technical Report CAPD Tech Rep B-99-07, Carnegie Mellon University, 1999. 2

[20] H. Yamashita, H. Yabe, and T. Tanabe. A globally and superlinearly convergent primal-dual interior point trust region method for large scale constrained optimization. Technical report, Mathematical Systems Inc., 1997, revised 1998. 14, 15, 17

[21] Y. Ye, M.J. Todd, and S. Mizuno. An $o(\sqrt{nl})$-iteration homogeneous and self-dual linear programming algorithm. *Mathematics of Operations Research*, 19:53–67, 1994. 18

HANDE Y. BENSON, PRINCETON UNIVERSITY, PRINCETON, NJ

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

ROBERT J. VANDERBEI, PRINCETON UNIVERSITY, PRINCETON, NJ