LOQO: AN INTERIOR POINT CODE FOR QUADRATIC PROGRAMMING

ROBERT J. VANDERBEI

Statistics and Operations Research
Princeton University
SOR-94-15

Revised: November 30, 1998

ABSTRACT. This paper describes a software package, called LOQO, which implements a primal-dual interior-point method for general nonlinear programming. We focus in this paper mainly on the algorithm as it applies to linear and quadratic programming with only brief mention of the extensions to convex and general nonlinear programming, since a detailed paper describing these extensions were published recently elsewhere. In particular, we emphasize the importance of establishing and maintaining symmetric quasidefiniteness of the reduced KKT system. We show that the industry standard MPS format can be nicely formulated in such a way to provide quasidefiniteness. Computational results are included for a variety of linear and quadratic programming problems.

1. INTRODUCTION

LOQO is a software package for solving general (smooth) nonlinear optimization problems. It implements an infeasible-primal-dual path-following method. For linear programming, such methods were first proposed independently by Lustig [12] and Tanabe [26]. The method as applied to LP was subsequently studied empirically by Lustig et.al. [13]. Global convergence was proved by Kojima et.al [11] while Zhang [39] established polynomiality of a long-step infeasible path-following method. Finally, superlinearly convergent variants were given by Potra [23] and Wright [37]. Detailed discussion of the computational aspects for infeasible-interior-point methods for linear programming were given by Lustig et.al. [15] and by Andersen et.al. [1]. Comprehensive modern treatments of interior-point methods for linear programming can be found in [38] and [33].

The first extension of primal-dual path-following methods to convex optimization was given by Monteiro and Adler [20] with superlinear convergence being established somewhat later by Monteiro

Research supported by AFOSR through grant AFOSR-91-0359, the NSF through grant CCR-9403789, and the ONR through grant N00014-98-1-0036.

and Zhang [21]. Preliminary computational studies were done by Breitfeld and Shanno [3] and Shanno and Simantiraki [25].

The software package LOQO has undergone many developmental phases since the first version was released in 1991. From the start, we recognized that quasidefinite matrices (introduced and studied in [32]) are important for obtaining efficient algorithms for convex quadratic programming. The original incarnation of LOQO (version 1.xx) is described in [34]. In these early versions, quasidefiniteness was not present in the reduced KKT system itself but rather was introduced during the process of matrix factorization by an appropriate choice of initial pivots. In 1994, we discovered techniques guaranteeing quasidefiniteness in the reduced KKT system itself. This greatly freed up the choice of pivot elements in the factorization process. The technical report [31] documenting these changes was never published. This paper is derived directly from that technical report. In preparing this paper, we updated the text to reflect developments taking place since 1994 and we completely redid all computational experiments using the most current version of LOQO (3.10).

The most significant development between 1994 and the present is that LOQO can now solve general nonlinear optimization problems (having smoothly defined objective and constraints). The changes to the algorithm that makes this possible are described in detail in [36]. In this paper, we describe them only briefly.

Before proceeding with a detailed treatment, we present in the following subsection on overview of the main feature that distinguishes LOQO for other LP and QP solvers, namely, having a quasidefinite reduced KKT matrix regardless of problem formulation.

1.1. Quasidefinite Reduced KKT Systems. Most software packages for solving linear programming problems using interior point technology take as input a linear program in the industry standard MPS format. This is a flexible format that allows variables to be free or bounded (on one side or both sides) and allows constraints to be either inequalities or equalities with the inequalities even allowed to be two-sided. Yet, these packages invariably map the given problem to a *standard form* in which all constraints are equalities and all variables are either free, nonnegative, or bounded between zero and a finite upper bound. The algorithm then involves solving a system of equations involving the following reduced KKT matrix:

$$\begin{bmatrix} -D & A^T \\ A & 0 \end{bmatrix}.$$

Here, A is the constraint matrix for the problem in standard form and D is a diagonal matrix having nonnegative values. The diagonal elements of D are positive finite numbers with the following exceptions:

- Free variables produce zero diagonal elements.
- Variables having zero upper bound (i.e. fixed variables in the original MPS file) produce infinitely large diagonal elements.

Hence, free variables and fixed variables raise specific issues that we shall address.

It is easy to remedy the situation regarding fixed variables. The infinite diagonal element is a consequence of initializing the primal variable x_j and its slack t_j to the upper bound u_j so that the

slack's definitional constraint is satisfied:

$$x_j + t_j = u_j.$$

When $u_j = 0$, this constraint forces both x_j and t_j to vanish, which is how an infinity ends up appearing in the corresponding location on D. However, Lustig [12] (and independently Tanabe [26]) showed that interior-point methods could and should be viewed as infeasible methods and that the obvious algorithm will work toward feasibility at the same time that it works toward optimality. Once, infeasible interior-point methods became popular with the appearance of Lustig, et.al.'s, computationally oriented paper [13], it became natural to relax the above constraint and initialize x_j and t_j to a positive value even when the upper bound is zero. This remedy for the fixed variable problem was discovered independently by Lustig et.al. [12] and by Vanderbei [28]. Of course, as the algorithm progresses, the fixed variables x_j and the corresponding t_j will get small and this will force the corresponding diagonal element of D to grow very large. But it is well-known that even without fixed variables, every element of D approaches either infinity or zero. Hence, this remedy for fixed variables makes them behave no worse than any other variables.

Regarding free variables, one possibility is to replace each one with a difference of two nonnegative numbers. There are two disadvantages to this approach. First, it involves a model reformulation and, as our ultimate goal is to create a code for general nonlinear programming, it is desirable to design an algorithm that does not include any problem reformulations in the reduced KKT system since in nonlinear programming these reformulations will need addressing at every iteration rather than just at the beginning. Secondly and more importanly, previous work (see, e.g., [15]) has identified numerical instabilities caused by pairs of split variables becoming large even as their difference remains bounded. This tendency has caused numerical troubles in the past and specific techniques have been proposed to remedy this situation such as simultaneously shifting them at every iteration so that the smaller one is some specified value. In this paper, we revisit the issue of free variable splitting and argue that the agressive reduction of the barrier parameter found in LOQO (and other codes) prevents the split variables from getting large. In LOQO, for each free variable a new constraint is added (implicitly) that asserts that the free variable is the difference of two nonnegative numbers. The free variable is not then eliminated. Instead, the step directions associated with the nonnegative variables are eliminated in the reduction from the KKT system to the reduced KKT system. That elimination then produces a positive diagonal element in the reduced KKT system.

Of course, the zero matrix in the lower right-hand block of the reduced KKT matrix is also undesirable. Because of this zero matrix, most existing implementations first choose their pivots from the upper left block and thereby reduce the system to the so-called normal equations which involves a matrix of the following form:

$$AD^{-1}A^T.$$

Many papers have discussed the pros and cons of working with the normal equations. The obvious advantage is that the matrix is positive semidefinite. But a disadvantage appears when A has dense columns. In this case, the positive semidefinite matrix is dense even if A is mostly very sparse. A related disadvantage appears when one wishes to generalize the algorithm from solving linear programs to one that can solve quadratic programming problems.

The zero on the lower right-hand block appears because of the conversion of every problem to equality form. If one had derived the algorithm to act on problems in inequality form (even allowing two sided inequalities), then this lower right-hand block becomes a diagonal matrix with all strictly negative elements. But then one must address how to handle equality constraints, if in fact the original problem has them (as they almost always do). The obvious approach is to represent each equality constraint as a two sided inequality with the two sides agreeing:

$$b \le a^T x \le b.$$

If one does this and relaxes the appropriate slack definitional constraints, one obtains a method for treating equality constraints that produces strictly negative entries in the lower right-hand block. Note that this technique is quite analogous to the technique we described above for handling fixed variables.

Employing the techniques described above, the reduced KKT matrix becomes a symmetric quasi-definite matrix:

$$\begin{bmatrix} -D & A^T \\ A & E \end{bmatrix}.$$

Here, D and E are positive *definite* diagonal matrices. We showed in [32] that one can factor any symmetric permutation of a symmetric quasi-definite matrix. Hence, using this approach it is quite natural to solve the reduced KKT system directly instead of reducing it to the system of normal equations.

1.2. Outline. In this paper, we describe in detail the algorithm as outlined above. We also describe a software package, called LOQO, which implements this method. In Section 2, we describe one iteration of the symmetrically formulated primal-dual path-following method. Section 3, describes the starting point selection and termination criteria as implemented in LOQO. Then, in Section 4, some more specific implementation details are presented that further define the algorithm. Section 5 discusses the extension of the algorithm to handle pure inequalities (i.e. infinite ranges) and infinite upper and/or lower bounds on the variables. Finally, computational results can be found in Section 6.

2. ALGORITHM

The algorithm implemented in LOQO is a infeasible-primal-dual path-following method. As implemented, it operates directly on QP problems presented in the following general form, which for LP's

corresponds directly to the industry standard MPS format:

$$\begin{array}{ll} \text{minimize} & f + c^T x + \frac{1}{2} x^T H x \\ \text{subject to} & b \le Ax \le b + r \\ & l \le x \le u. \end{array} \tag{2.1}$$

Here, A is the $m \times n$ matrix of coefficients, b is called the *right-hand side* (even though it appears on the left), r is the vector of *ranges* on the constraints, and u and l are the vectors of *upper bounds* and *lower bounds*, respectively, on the variables. Each element b_i of b is assumed to be a finite real number. However, the elements of the other vectors are permitted to take values in the extended reals subject to the following limitations:

$$\begin{aligned} 0 \le r_i &\le \infty \\ -\infty \le l_j &< \infty \\ -\infty < u_j &\le \infty. \end{aligned}$$

Infinities require special treatment. They shall be discussed in Section 5.

The objective function, $f + c^T x + \frac{1}{2} x^T H x$, is a quadratic function. For historical reasons, the constant term f is called the *fixed adjustment*. The matrix H appearing in the quadratic term is assumed to be positive semidefinite so that the objective function is convex.

Derivations of the infeasible-primal-dual path-following method for problems presented in simpler forms can be found, for example, in [5, 13, 14, 34, 30, 35]. Hence, we proceed directly to the derivation in the present general context.

2.1. Add Slacks. The first step in the derivation is to introduce slack variables as appropriate to replace all inequality constraints with simple nonnegativity constraints. Hence, we rewrite the primal problem (2.1) as follows:

$$\begin{array}{lrcl} \text{minimize} & f + c^T x + \frac{1}{2} x^T H x & & \\ \text{subject to} & Ax - w &=& b \\ & x - g &=& l \\ & x + t &=& u \\ & w + p &=& r \\ & g, w, t, p &\ge& 0 \\ & x \text{ free.} & & \end{array} \tag{2.2}$$

The dual of (2.2) is:

$$
\begin{array}{lll}
\text{maximize} & f + b^T y - \frac{1}{2} x^T H x + l^T z - u^T s - r^T q & \\
\text{subject to} & A^T y + z - s - Hx &= c \\
& y + q - v &= 0 \\
& z, v, s, q &\geq 0 \\
& y \text{ free.} &
\end{array}
\tag{2.3}
$$

2.2. Central Path. The next step in the derivation is to introduce the primal-dual *central path* (introduced by Megiddo [17]—see also [33]). We parametrize this path by a positive real parameter μ. Indeed, for each $\mu > 0$, we define the associated central-path point in primal-dual space as the unique point that simultaneously satisfies the conditions of primal feasibility, dual feasibility, and μ-complementarity. Ignoring nonnegativity (which is enforced separately), these conditions are:

$$
\begin{aligned}
Ax - w &= b \\
x - g &= l \\
x + t &= u \\
w + p &= r \\
\\
A^T y + z - s - Hx &= c \\
y + q - v &= 0 \\
\\
GZe &= \mu e \\
VWe &= \mu e \\
STe &= \mu e \\
PQe &= \mu e.
\end{aligned}
\tag{2.4}
$$

The last four equations are the μ-complementarity conditions. As usual, each upper case letter that appears on the left in these equations denotes the diagonal matrix having the components of the corresponding lower-case vector on its diagonal. This is a nonlinear system of $5n+5m$ equations in $5n+5m$ unknowns. It has a unique solution in the strict interior of the appropriate orthant in primal-dual space:

$$
\{(x, g, w, t, p, y, z, v, s, q) : g, w, t, p, z, v, s, q \geq 0\}. \tag{2.5}
$$

This fact can be seen by noting that these equations are the first order optimality conditions for an associated strictly convex barrier problem (see, e.g. [35]).

As μ tends to zero, the central path converges to an optimal solution to both the primal and dual

problems. A *primal-dual path-following algorithm* is defined as any iterative process that starts from a point in the strict interior of (2.5) and at each iteration estimates a value of μ representing a point on the central path that is in some sense closer to the optimal solution than the current point and then attempts to step toward this central-path point making sure that the new point remains in the strict interior of the appropriate orthant.

Suppose for the moment that we have already decided on the target value for μ. Let $(x, \ldots, q)$ denote the current point in the orthant and let $(x + \Delta x, \ldots, q + \Delta q)$ denote the point on the central path corresponding to the target value of μ. The defining equations for the point on the central path can be written as

$$
\begin{array}{rcll}
A\Delta x - \Delta w &=& b - Ax + w &=: \rho \\
\Delta x - \Delta g &=& l - x + g &=: \nu \\
\Delta x + \Delta t &=& u - x - t &=: \tau \\
\Delta w + \Delta p &=& r - w - p &=: \alpha \\
A^T \Delta y + \Delta z - \Delta s - H\Delta x &=& c - A^T y - z + s + Hx &=: \sigma \\
-\Delta y - \Delta q + \Delta v &=& y + q - v &=: \beta \\
G^{-1} Z \Delta g + \Delta z &=& \mu G^{-1} e - z - G^{-1}\Delta G \Delta z &=: \gamma_z \\
V^{-1} W \Delta v + \Delta w &=& \mu V^{-1} e - w - V^{-1}\Delta V \Delta w &=: \gamma_w \\
S T^{-1} \Delta t + \Delta s &=& \mu T^{-1} e - s - T^{-1}\Delta T \Delta s &=: \gamma_s \\
P^{-1} Q \Delta p + \Delta q &=& \mu P^{-1} e - q - P^{-1}\Delta P \Delta q &=: \gamma_q,
\end{array}
\tag{2.6}
$$

where we have introduced notations $\rho, \ldots, \gamma_q$ as shorthands for the right-hand side expressions. This is almost a linear system for the direction vectors $(\Delta x, \ldots, \Delta q)$. The only nonlinearities appear on the right-hand sides of the complementarity equations (i.e., in $\gamma_z, \ldots, \gamma_q$).

2.3. Predictor-Corrector. LOQO implements a *predictor-corrector* [18] approach to finding a good approximate solution to equations (2.6). The *predictor step* consists of dropping both the μ terms and the "delta" terms that appear on the right-hand side (i.e., $\gamma_z = -z$, etc.) and solving the resulting linear system for the "delta" variables. Then an estimate of an appropriate target value for μ is made and the μ and "delta" terms are reinstated on the right-hand side using the current estimates and the resulting system is again solved for the "delta" variables. This second calculation is refered to as the *corrector step* and the resulting step directions are used to move to a new point in primal-dual space.

2.4. Solving the Indefinite System. Clearly the main computational burden is to solve system (2.6) twice in each iteration. It is important to note that this is a large, sparse, indefinite, linear system. It is,

however, symmetric if one negates certain rows and rearranges rows and columns appropriately:

$$\left[\begin{array}{cccc|cccccc}
-ST^{-1} & & & & & & & -I & & \\
& -G^{-1}Z & & & & & -I & & & \\
& & -P^{-1}Q & & & -I & & & & \\
& & & & -I & -I & & & I & \\
& & & -H & A^T & & I & -I & & \\
\hline
& & -I & A & & & & & & \\
& -I & -I & & & & & & & \\
-I & & & I & & & & & & \\
& & & -I & & & & & & \\
& & I & & & & & & V^{-1}W &
\end{array}\right]
\left[\begin{array}{c} \Delta t \\ \Delta g \\ \Delta p \\ \Delta w \\ \Delta x \\ \hline \Delta y \\ \Delta q \\ \Delta z \\ \Delta s \\ \Delta v \end{array}\right]
=
\left[\begin{array}{c} -\gamma_s \\ -\gamma_z \\ -\gamma_q \\ \beta \\ \sigma \\ \hline \rho \\ -\alpha \\ \nu \\ -\tau \\ \gamma_w \end{array}\right].$$

As we shall see shortly, the symmetry of this system suggests a systematic process of elimination. In other works (such as [13, 14]) the analogous system is not written in a symmetric form and the elimination process, in the early stages, is often performed in *ad hoc* ways until ultimately arriving at a symmetric positive semidefinite system (involving a matrix with the symbolic structure of either AA^T or A^TA and just one of the delta variables – either Δy or Δx). These other works then assert that this new system can be solved robustly and hence that the reduction yields a more numerically stable procedure. However, this argument ignores the fact that, once the reduced system has been solved, its solution must then be used to "back substitute" to get all of the other delta variables that were eliminated and that these extra calculations represent predetermined pivots in the original system that might reintroduce the numerical instability that one had hoped to avoid. Hence, in the end, we are still faced with solving a large, indefinite, linear system and it is this system that we should study carefully. By employing a systematic elimination process from the beginning, one hopes to obtain a more robust procedure for the original large system.

The most numerically robust technique for solving large, perhaps nonsymmetric, systems is to employ a full-pivoting solver. Unfortunately, such an approach yields a code that is significantly slower than ones based on a predetermined reduction. Also, experience shows that these predetermined reductions (which amount to pivoting without regard for the size of the pivot element) can yield remarkably robust codes (this phenomenon is only partially understood – see, e.g., [32]). Hence, to be competitive, LOQO uses a predetermined reduction, which we now describe.

Now we describe the systematic elimination process. First, we use the pivot elements $-ST^{-1}$, $-G^{-1}Z$, $-P^{-1}Q$, and $V^{-1}W$ to solve for Δt, Δg, Δp, and Δv, respectively,

$$\begin{aligned}
\Delta t &= S^{-1}T(\gamma_s - \Delta s) \\
\Delta g &= GZ^{-1}(\gamma_z - \Delta z) \\
\Delta p &= PQ^{-1}(\gamma_q - \Delta q) \\
\Delta v &= VW^{-1}(\gamma_w - \Delta w),
\end{aligned}$$

which we then eliminate from the remaining equations to obtain

$$\left[\begin{array}{cc|ccccc} -VW^{-1} & & -I & -I & & & \\ & -H & A^T & & I & -I & \\ \hline -I & A & & & & & \\ -I & & & PQ^{-1} & & & \\ & I & & & GZ^{-1} & & \\ & -I & & & & S^{-1}T & \end{array}\right]\left[\begin{array}{c} \Delta w \\ \Delta x \\ \hline \Delta y \\ \Delta q \\ \Delta z \\ \Delta s \end{array}\right] = \left[\begin{array}{rcl} \beta - VW^{-1}\gamma_w & =: & \hat{\beta} \\ \sigma & & \\ \hline \rho & & \\ -\alpha + PQ^{-1}\gamma_q & =: & -\hat{\alpha} \\ \nu + GZ^{-1}\gamma_z & =: & \hat{\nu} \\ -\tau + S^{-1}T\gamma_s & =: & -\hat{\tau} \end{array}\right],$$

where once again we have introduced abbreviated notations for some of the right-hand side expressions. Next, we use the pivot elements PQ^{-1}, GZ^{-1}, and $S^{-1}T$ to solve for Δq, Δz, and Δs, respectively,

$$\begin{aligned} \Delta q &= P^{-1}Q(\Delta w - \hat{\alpha}) \\ \Delta z &= G^{-1}Z(\hat{\nu} - \Delta x) \\ \Delta s &= ST^{-1}(\Delta x - \hat{\tau}). \end{aligned}$$

Eliminating these variables from the remaining equations, we get

$$\left[\begin{array}{cc|c} -E^{-1} & & -I \\ & -(H+D) & A^T \\ \hline -I & A & \end{array}\right]\left[\begin{array}{c} \Delta w \\ \Delta x \\ \hline \Delta y \end{array}\right] = \left[\begin{array}{c} \hat{\beta} - P^{-1}Q\hat{\alpha} \\ \sigma - G^{-1}Z\hat{\nu} - ST^{-1}\hat{\tau} \\ \hline \rho \end{array}\right],$$

where

$$E = \left(VW^{-1} + P^{-1}Q\right)^{-1}$$

and

$$D = G^{-1}Z + ST^{-1} \tag{2.7}$$

are positive-definite diagonal matrices. Finally, we use the pivot element $-E^{-1}$ to solve for Δw,

$$\Delta w = -E(\hat{\beta} - P^{-1}Q\hat{\alpha} + \Delta y),$$

which brings us to the so-called *reduced KKT* equations:

$$\left[\begin{array}{c|c} -(H+D) & A^T \\ \hline A & E \end{array}\right]\left[\begin{array}{c} \Delta x \\ \hline \Delta y \end{array}\right] = \left[\begin{array}{c} \sigma - G^{-1}Z\hat{\nu} - ST^{-1}\hat{\tau} \\ \hline \rho - E(\hat{\beta} - P^{-1}Q\hat{\alpha}) \end{array}\right]. \tag{2.8}$$

2.5. Solving the Symmetric Quasidefinite System. Up to this point, none of the eliminations have produced off-diagonal fill-in in the remaining system. However, proceeding further will definitely introduce such fill-in. For example, if we were to use the first equation in (2.8) to solve for Δx and then eliminate it from the second equation, the resulting system for Δy would involve the matrix

$$A(D+H)^{-1}A^T$$

(this is the *normal-equations* approach used in OB1 [13, 14], CPLEX-barrier, and other codes). Similarly, if we were to solve the second equation for Δy and then eliminate this variable from the first equation, the resulting system for Δx would involve the matrix

$$A^T E^{-1} A$$

(this is the approach advocated by the optimization group at the National Institute of Standards and Technology [2, 24]). Both of these will generally entail fill-in, which in some cases can be considerable. For example, if the Hessian H is not a diagonal matrix or if the matrix A has a dense column, the first form can suffer "catastrophic fill-in" (see [19] for analysis of non-diagonal H). On the other hand, if A has a dense row, the second form will be very bad. If A has both dense columns and dense rows, then both of these forms will suffer unnecessarily large amounts of fill-in and one would prefer, in that case, to work with the larger system to find a pivot order that does not generate so much fill-in. This idea was first suggested by Turner [27] and has been adopted by Saunders et.al. [10, 9, 8] and Mehrotra [5]. All three use a Bunch-Parlet factorization of the indefinite system. Solving the larger system is also the approach adopted in LOQO, but LOQO does not employ a Bunch-Parlet factorization. Instead, LOQO uses a modified Cholesky factorization code that has been altered to solve *symmetric quasidefinite systems* (see [32]). Equation (2.8) is an example of such a system. The matrices defining these systems share with symmetric semidefinite matrices the nice property that the (diagonal) pivots can be selected based only on a fill-in minimizing heuristic; i.e., without regard for the numerical values, which may only be known later.

There are two types of fill-in minimizing heuristics: myopic heuristics, such as *minimum-degree* [7], which sequentially minimize the fill-in produced in each subsequent stage of elimination and global heuristics, such as *nested-dissection* [22], which analyze the overall structure to find good orderings. In the symmetric quasidefinite system (2.8), there is an obvious global structure that one can exploit. For example, the lower-right block is a diagonal matrix (as is the upper-left block whenever H is diagonal or absent). Hence, initial pivots selected from the lower-right block can only produce fill-in in the upper-left block. It is potentially advantageous to exploit this structure. To test this hypothesis, we compared two ordering schemes. The first is just a straight minimum-degree heuristic implemented as described in [29]. The second can be described as a priority minimum-degree method. Each diagonal element is assigned a small integer representing an elimination priority. At first, pivots are selected only from the elements assigned priority zero. Within this priority class, pivots are selected according to the usual minimum degree heuristic. Only after all priority zero pivots have been eliminated do we proceed to the priority one elements. Again, within this priority class, the elimination order is determined using the minimum-degree heuristic. This process is continued through each priority class until all elements are eliminated.

The priority classes are determined as follows. First a simple estimate is made of the number of nonzeros in AA^T and of the number of nonzeros in A^TA. If these estimates indicate that AA^T will have less fill-in than A^TA, then the pivot elements from the upper-left block are assigned priority zero and the lower-right block elements are given priority one. Otherwise, the reverse priority is assigned. Next, the priority zero pivots are scanned to see if any of them have high degree (relative to the other elements). Such elements are then reassigned to priority one so that these elements (which correspond to dense columns/rows of A) are eliminated last.

Let K denote the symmetric quasidefinite matrix in (2.8). The ordering heuristic computes a permutation matrix P which is applied to both the rows and the columns of K and the resulting matrix is factored into the product of a unit lower triangular matrix L times a diagonal matrix D times the

transpose of the lower triangular matrix:

$$PKP^T = LDL^T.$$

While we often refer to fill-in produced by some ordering heuristic, a better gauge of the quality of the ordering is to count the number of arithmetic operations, narth, required to compute the factorization. There is a simple formula for this:

$$\text{narth} = \sum_{j=0}^{k} \text{nonz}(L_j)^2 + 3\ \text{nonz}(L) + k.$$

Here, k denotes the number of rows/columns of K, nonz$(\cdot)$ denotes the number of nonzeros and L_j denotes the j-th column of L.

Table 1 shows the results of our comparison. It is clear that there can be a substantial difference between the two methods. However, on the average they are about the same. To justify this claim, let R_i denote the ratio of the number of arithmetic operations using the ordinary minimum-degree method to the number obtained using the priority method for the i-th problem. Since a ratio of 2 and a ratio of $1/2$ should be counted equally, we use the geometric mean to summarize the overall average behavior:

$$\bar{R} = \exp(\frac{1}{N}\sum_{i=1}^{N} \log R_i).$$

For the data in Table 1, $\bar{R} = 0.964$, which indicates that the ordinary minimum-degree heuristic was on the average 3.6% better. But, at the 95% confidence level, we must reject the hypothesis that the ordinary minimum-degree heuristic is better since the standard deviation of the logarithm of the R_i's divided by the square root of one less than the sample size is 0.037 indicating that the 3.6% deviation is only about one standard deviation away from the mean. Hence, we have no evidence to say that one method produces a better ordering than the other. However, there is a significant difference from the point of view of numerical stability. The priority-minimum-degree method tends to arrange the computations so that all additions (to positive diagonal elements) are taken first and the subtractions come last. This enhances the numerical stability greatly. Indeed, using the priority method, LOQO solves essentially all of the NETLIB problems listed in Table 1 (except **dfl001**) whereas the ordinary minimum-degree heuristic has trouble on 31 out of the 92 problems. For this reason, LOQO uses the priority method.

3. Openings and Endgames.

To start the algorithm we need to provide initial values for all the variables. Variables x and y are unrestricted whereas all the others must be positive. A simple heuristic would be to set the unrestricted variables to zero and the positive variables to one. However, it is better to try to arrange things to more closely satisfy at least some of the equations. LOQO initializes the variables as follows. First, x and y are found as solutions to the following system:

$$\begin{bmatrix} -(H+I) & A^T \\ A & I \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} c \\ b \end{bmatrix}.$$

Problem	Arithmetic Operations			Problem	Arithmetic Operations		
	Priority				Priority		
	Min.	Degree	Min. Degree		Min.	Degree	Min. Degree
25fv47	2539174		3565306	nesm	1748429		1431903
80bau3b	2748509		2748509	perold	2191541		2876285
adlittle	7133		5663	pilot4	1010846		555014
afiro	1061		845	pilot87	207214909		228516729
agg	196979		196979	pilotja	7592614		6558376
agg2	657690		1396978	pilotnov	5674027		6147413
agg3	657758		1666364	pilots	51224067		39909677
bandm	128171		143375	pilotwe	2391227		1107375
beaconfd	153089		163825	recipe	19867		31217
blend	22927		15907	sc105	5612		5758
bnl1	532244		579218	sc205	12704		11202
bnl2	14193875		14745741	sc50a	2162		1974
boeing1	210267		218625	sc50b	2430		1634
boeing2	76349		51049	scagr25	36423		21369
bore3d	91596		58960	scagr7	9189		5395
brandy	144779		210973	scfxm1	132963		127613
capri	258242		138934	scfxm2	276270		276642
cycle	6698386		5167144	scfxm3	420143		421377
czprob	181634		180058	scorpion	34786		22472
d2q06c	31944530		35309876	scrs8	203087		127471
d6cube	10731175		10821267	scsd1	46301		46301
degen2	1105538		1404352	scsd6	79279		79279
degen3	16180237		18245339	scsd8	154037		154037
dfl001	1528765265		1407015715	sctap1	47050		40934
e226	127213		208535	sctap2	638534		606684
etamacro	1187084		1219956	sctap3	685536		655744
fffff800	948102		1410548	seba	59443		59443
finnis	148455		122893	share1b	35160		30858
fit1d	224438		224438	share2b	22051		21849
fit1p	147998		147998	shell	89763		100237
fit2d	2031079		2031079	ship04l	112860		112048
fit2p	649721		649721	ship04s	80664		78364
forplan	200002		173956	ship08l	229591		229319
ganges	1267748		650708	ship08s	133199		128809
gfrdpnc	23646		23260	ship12l	285088		282560
greenbea	5753313		15706417	ship12s	148952		139800
greenbeb	5628409		15706417	sierra	362479		377549
grow15	235265		318707	stair	1399843		456177
grow22	347986		461336	standata	77818		67076
grow7	106441		160009	standmps	143212		120888
israel	110400		110400	stocfor1	14206		12014
kb2	10854		7928	stocfor2	693398		437664
lotfi	42453		31707	tuff	430930		388346
maros-r7	768194856		752106384	vtpbase	64143		29011
maros	1717571		2604431	wood1p	3738076		5835322
modszk1	117585		100731	woodw	3415693		3532285

TABLE 1. Number of arithmetic operations required to factor the symmetric indefinite system using (a) the minimum-degree heuristic and (b) the priority minimum-degree heuristic.

Then the other variables are set as follows:

$$\begin{aligned} g &= \max(\text{abs}(x - l), 100) \\ z &= \max(\text{abs}(x), 100) \\ t &= \max(\text{abs}(u - x), 100) \\ s &= \max(\text{abs}(x), 100) \\ v &= \max(\text{abs}(y), 100) \\ w &= \max(\text{abs}(y), 100) \\ p &= \max(\text{abs}(r - w), 100) \\ q &= \max(\text{abs}(y), 100), \end{aligned}$$

where max() and abs() denote componentwise maximum and absolute value, respectively.

The default stopping rule in LOQO is to stop when the primal and dual are both feasible and their objective functions agree to eight significant figures. The level of primal infeasibility is measured by computing

$$\text{primal\_infeasibility} = \frac{\sqrt{\|\rho\|^2 + \|\tau\|^2 + \|\alpha\|^2 + \|\nu\|^2}}{\|b\| + 1}.$$

Similarly, dual infeasibility is measured using

$$\text{dual\_infeasibility} = \frac{\sqrt{\|\sigma\|^2 + \|\beta\|^2}}{\|c\| + 1}.$$

By default, a solution is primal/dual feasible if these measures are less than 10^{-6}. The significant figures of agreement between the primal objective function value and the dual is computed as follows:

$$\text{sigfig} = \max\left(-\log_{10}\frac{|\text{primal\_obj} - \text{dual\_obj}|}{|\text{primal\_obj}| + 1}, \quad 0\right).$$

In addition to identifying optimal solutions, it is also important to identify infeasible and unbounded problems. This identification is based on the observation (see, e.g., [35]) that the primal infeasibility and the dual infeasibility decrease monotonically. If the primal problem is infeasible, then the primal infeasibility will stall (and in fact start to increase) before getting close to zero. If the primal problem is unbounded, then the dual infeasibility's reduction will stall. LOQO tests for stalling and, if detected, terminates with an appropriate message.

4. MIDGAME STRATEGIES.

Two midgame strategies need to be mentioned. The first is the computation of step lengths. At the end of each iteration, the current solution is updated to a new solution according to the following

formulas:

$$\begin{aligned}
x &\leftarrow x + \Delta x/\alpha_p \\
g &\leftarrow g + \Delta g/\alpha_p \\
w &\leftarrow w + \Delta w/\alpha_p \\
t &\leftarrow t + \Delta t/\alpha_p \\
p &\leftarrow p + \Delta p/\alpha_p \\
\\
y &\leftarrow y + \Delta y/\alpha_d \\
z &\leftarrow z + \Delta z/\alpha_d \\
v &\leftarrow v + \Delta v/\alpha_d \\
s &\leftarrow s + \Delta s/\alpha_d \\
q &\leftarrow q + \Delta q/\alpha_d .
\end{aligned}$$

The normalizations α_p and α_d should be one (since the step directions were derived based on this assumption), but they may need to be shortened to maintain strict positivity of the nonnegative variables. Hence they are calculated as follows:

$$\begin{aligned}
\alpha_p &= \max\left(\max\left(-\frac{\Delta g_j}{g_j}, -\frac{\Delta w_i}{w_i}, -\frac{\Delta t_j}{t_j}, -\frac{\Delta p_i}{p_i}\right)/0.95, \quad 1\right) \\
\alpha_d &= \max\left(\max\left(-\frac{\Delta z_j}{z_j}, -\frac{\Delta v_i}{v_i}, -\frac{\Delta s_j}{s_j}, -\frac{\Delta q_i}{q_i}\right)/0.95, \quad 1\right).
\end{aligned}$$

Here, the inner maximizations are over all $j = 1, 2, \ldots, n$ or all $i = 1, 2, \ldots, m$ as appropriate for each of the four terms in the argument list. If the problem is a pure linear programming problem, then the step lengths are as given. If it is a quadratic program, then a common normalization is used:

$$\begin{aligned}
\alpha_p &\leftarrow \max(\alpha_p, \alpha_d) \\
\alpha_d &\leftarrow \max(\alpha_p, \alpha_d).
\end{aligned}$$

The second midgame strategy is the computation of the parameter μ. To understand the heuristic used to come up with a value for μ, first consider a point on the central path. From (2.4), we see that

$$z^T g = \mu n, \quad v^T w = \mu m, \quad s^T t = \mu n, \quad \text{and} \quad p^T q = \mu m.$$

Hence, one way to recover μ from a given point on the central path is to compute

$$\mu = \frac{z^T g + v^T w + s^T t + p^T q}{2m + 2n}.$$

We use this expression to assign a *par value* for μ even when the current point is not on the central path. Since we wish to step from the current point toward a point on the central path that is "closer" to the optimal solution it is desirable to use a value of μ that is some fraction of the par value. Simply setting this fraction to one tenth works most of the time, but it turns out to be better to assign it dynamically based on how much one must shorten the predictor step direction to preserve strict positivity of the nonnegative variables. Hence, LOQO calculates values for α_p and α_d as described above using the

predictor directions and then multiplies the par value of μ by the following fraction:

$$\left(\frac{\alpha_{pd} - 1}{\alpha_{pd} + 10}\right)^2,$$

where

$$\alpha_{pd} = \max(\alpha_p, \alpha_d).$$

This fraction tends to be close to zero when α_{pd} is close to 1 but gets close to one as α_{pd} becomes large.

5. HANDLING INFINITIES.

In this section, we discuss the absense of bounds on variables and ranges on constraints.

Our basic problem formulation (2.1) shows variables having two-sided bounds (above and below). But real-world problems contain a mix of bound types: some variables may have two-sided bounds, while others have only one-sided bounds (either upper or lower) and yet others have no bounds whatsoever (so-called *free variables*). Missing bounds are easy to handle. We simply omit the slack variable associated with the missing bound. For example, if x_j is a variable having no upper bound, then the primal problem (2.2) will have no upper-bound slack variable t_j and the associated dual (2.3) will be missing the corresponding variable s_j. Similarly, if a lower bound is missing, then a g_j and z_j are dropped. This causes no difficulties except for the fact that each free variable will produce a zero on the diagonal matrix D given by (2.7). Hence, free variables must be handled separately in the factorization of the symmetric quasidefinite matrix appearing in (2.8). In LOQO versions 2.16 and earlier, free variables are treated exactly this way. The diagonal entries of the symmetric quasidefinite matrix associated with these free variables are assigned to priority class two thereby ensuring that they are eliminated last (by which time the diagonals should be nonzero). In LOQO versions 2.17 and later, free variables are handled in a different manner – one in which the none of the diagonals are ever zero. This produces a more robust code. The details are discussed below.

If the i-th constraint has infinite range, i.e. a pure inequality constraint, then we leave out the slack variable p_i associated with this constraint and also the corresponding dual variable q_i.

5.1. Splitting Free Variables—An Old Idea Revisited. As mentioned in the introduction, an old technique for handling free variables, borrowed from the simplex method, is to replace each one with the difference of two nonnegative variables. It was noted early on that simple implementations of this technique have the property that the split variables tend to infinity even as the difference they represent remains bounded. To review why this happens, consider the following trivial problem:

$$\begin{aligned} &\text{minimize} && 0 \\ &\text{subject to} && x \geq 0. \end{aligned}$$

The dual for this problem is

$$\begin{array}{ll} \text{maximize} & 0 \\ \text{subject to} & z = 0 \\ & z \geq 0. \end{array}$$

(We are thinking of $x \geq 0$ as a bound constraint on x so that the constraint matrix in the primal is a 0×1 matrix. Therefore, the dual has no variables except for a slack variable, which we denote as usual by z.) The cental path equations for this problem are

$$\begin{aligned} z &= 0 \\ xz &= \mu. \end{aligned}$$

Of course, the central path itself does not exist and these equations have no solution (or, viewed another way, the solution is $x = \infty$, $z = 0$) but nonetheless they can be used to derive formulas for step directions:

$$\begin{aligned} \Delta z &= -z \\ z\Delta x + x\Delta z &= \mu - xz. \end{aligned}$$

For this problem, the step length turns out to be determined solely by z and is always as large as possible: $\alpha_{pd} = 1/0.95$. Consequently, the μ value used by LOQO on this problem is γxz where $\gamma = 0.000023$. It is easy to see then that after the k-th iteration

$$\begin{aligned} x^{(k)} &= x^{(0)}(1 + 0.95 * \gamma)^k \\ z^{(k)} &= z^{(0)}(1 - 0.95)^k. \end{aligned}$$

Of course, $x^{(k)}$ tends to infinity as k gets large. But, how fast? Consider the situation after 100 iterations. Starting from $x^{(0)} = z^{(0)} = 1$, we have $x^{(100)} \approx 1.002$ and $z^{(100)} \approx 10^{-130}$. So, z tends to zero (and thereby triggers the stopping rule) much faster than x tends to infinity. Of course, this is only a small example but it illustrates the fact that small values of μ can make the trend to infinity very slow. This is born out by extensive testing on hundreds of problems many of which are formulated entirely with free variables. Finally, we note that by changing the heuristic for calculating μ so that it involves a higher power of the par value, one can guarantee that even in the limit x remains finite. Such a statement is true for our small example and can probably be established in general but since LOQO does not use such a heuristic, we don't pursue it further.

Since splitting free variables has historically involved their elimination as well whereas we leave them in the model, we present here a detailed derivation of the resulting reduced KKT system. Indeed, consider a modification of our original problem (2.1) in which all variables are free:

$$\begin{array}{ll} \text{minimize} & f + c^T x + \frac{1}{2} x^T H x \\ \text{subject to} & b \leq Ax \leq b + r. \end{array} \tag{5.1}$$

Introducing new variables g and t and the constraint that $x = g - t$, we get the following primal

problem:

$$
\begin{array}{lrcl}
\text{minimize} & f + c^T x + \frac{1}{2} x^T H x & & \\
\text{subject to} & Ax - w & = & b \\
& x - g + t & = & 0 \\
& w + p & = & r \\
& g, w, t, p & \geq & 0 \\
& x \text{ free,} & &
\end{array}
\tag{5.2}
$$

which is clearly equivalent to (5.1). The dual of (5.2) is:

$$
\begin{array}{lrcl}
\text{maximize} & f + b^T y - \frac{1}{2} x^T H x - r^T q & & \\
\text{subject to} & A^T y + z - Hx & = & c \\
& z + s & = & 0 \\
& y + q - v & = & 0 \\
& z, v, s, q & \geq & 0 \\
& y \text{ free.} & &
\end{array}
\tag{5.3}
$$

(Note the high degree of symmetry between this primal/dual pair.) The equations defining the central path for this pair of problems closely parallels those we had before in (2.4):

$$
\begin{array}{rcl}
Ax - w & = & b \\
x - g + t & = & 0 \\
w + p & = & r \\
\\
A^T y + z - Hx & = & c \\
z + s & = & 0 \\
y + q - v & = & 0 \\
\\
GZe & = & \mu e \\
VWe & = & \mu e \\
STe & = & \mu e \\
PQe & = & \mu e.
\end{array}
\tag{5.4}
$$

Continuing with the usual algorithm derivation process, the equations for the step directions now

become:

$$
\begin{array}{rcll}
A\Delta x - \Delta w &=& b - Ax + w &=: \rho \\
\Delta x - \Delta g + \Delta t &=& -x + g - t &=: \nu \\
\Delta w + \Delta p &=& r - w - p &=: \alpha \\
A^T \Delta y + \Delta z - H\Delta x &=& c - A^T y - z + Hx &=: \sigma \\
\Delta z + \Delta s &=& -z - s &=: \tau \\
-\Delta y - \Delta q + \Delta v &=& y + q - v &=: \beta \\
G^{-1}Z\Delta g + \Delta z &=& \mu G^{-1}e - z - G^{-1}\Delta G\Delta z &=: \gamma_z \\
V^{-1}W\Delta v + \Delta w &=& \mu V^{-1}e - w - V^{-1}\Delta V\Delta w &=: \gamma_w \\
ST^{-1}\Delta t + \Delta s &=& \mu T^{-1}e - s - T^{-1}\Delta T\Delta s &=: \gamma_s \\
P^{-1}Q\Delta p + \Delta q &=& \mu P^{-1}e - q - P^{-1}\Delta P\Delta q &=: \gamma_q,
\end{array}
\tag{5.5}
$$

By appropriately arranging the variables and equations, this system can be written in a symmetric form:

$$
\left[\begin{array}{cccc|cccccc}
 & & & & & & I & I & & \\
 & -G^{-1}Z & & & & & -I & & & \\
 & & -P^{-1}Q & & & -I & & & & \\
 & & & & -I & -I & & & I & \\
 & & & -H & A^T & & I & & & \\
\hline
 & & & -I & A & & & & & \\
 & & -I & -I & & & & & & \\
I & -I & & & I & & & & & \\
I & & & & & & & S^{-1}T & & \\
 & & & I & & & & & V^{-1}W &
\end{array}\right]
\left[\begin{array}{c} \Delta t \\ \Delta g \\ \Delta p \\ \Delta w \\ \Delta x \\ \hline \Delta y \\ \Delta q \\ \Delta z \\ \Delta s \\ \Delta v \end{array}\right]
=
\left[\begin{array}{c} \tau \\ -\gamma_z \\ -\gamma_q \\ \beta \\ \sigma \\ \hline \rho \\ -\alpha \\ \nu \\ S^{-1}T\gamma_s \\ \gamma_w \end{array}\right].
$$

Using the pivot elements $-G^{-1}Z$, $-P^{-1}Q$, $S^{-1}T$, and $V^{-1}W$ to solve for Δg, Δp, Δs, and Δv, respectively, we get

$$
\begin{aligned}
\Delta g &= GZ^{-1}(\gamma_z - \Delta z) \\
\Delta p &= PQ^{-1}(\gamma_q - \Delta q) \\
\Delta s &= \gamma_s - ST^{-1}\Delta t \\
\Delta v &= VW^{-1}(\gamma_w - \Delta w),
\end{aligned}
$$

which we then eliminate from the remaining equations to obtain

$$
\left[\begin{array}{ccc|ccc}
-ST^{-1} & & & & & I \\
 & -VW^{-1} & & -I & -I & \\
 & & -H & A^T & & I \\
\hline
 & -I & A & & & \\
 & -I & & & PQ^{-1} & \\
I & & I & & & GZ^{-1}
\end{array}\right]
\left[\begin{array}{c} \Delta t \\ \Delta w \\ \Delta x \\ \hline \Delta y \\ \Delta q \\ \Delta z \end{array}\right]
=
\left[\begin{array}{c} \tau - \gamma_s =: \hat{\tau} \\ \beta - VW^{-1}\gamma_w =: \hat{\beta} \\ \sigma \\ \hline \rho \\ -\alpha + PQ^{-1}\gamma_q =: -\hat{\alpha} \\ \nu + GZ^{-1}\gamma_z =: \hat{\nu} \end{array}\right].
$$

Next, we use the pivot elements PQ^{-1} and GZ^{-1} to solve for Δq and Δz, respectively, to get

$$\begin{aligned}\Delta q &= P^{-1}Q(\Delta w - \hat{\alpha})\\ \Delta z &= G^{-1}Z(\hat{\nu} - \Delta x - \Delta t).\end{aligned}$$

Eliminating these variables from the remaining equations, we get

(5.6)
$$\left[\begin{array}{cccc|c} -(ST^{-1}+G^{-1}Z) & & -G^{-1}Z & & \\ & -(VW^{-1}+P^{-1}Q) & & & -I \\ -G^{-1}Z & & -(H+G^{-1}Z) & & A^T \\ \hline & -I & & A & \end{array}\right]\left[\begin{array}{c}\Delta t\\ \Delta w\\ \Delta x\\ \hline \Delta y\end{array}\right] = \left[\begin{array}{c}\hat{\tau} - G^{-1}Z\hat{\nu}\\ \hat{\beta} - P^{-1}Q\hat{\alpha}\\ \sigma - G^{-1}Z\hat{\nu}\\ \hline \rho\end{array}\right].$$

Finally, we use the pivot elements $-(ST^{-1}+G^{-1}Z)$ and $-(VW^{-1}+P^{-1}Q)$ to solve for Δt and Δw:

$$\begin{aligned}\Delta t &= -DS^{-1}T\left(GZ^{-1}\hat{\tau} - \hat{\nu} + \Delta x\right)\\ \Delta w &= -E\left(\hat{\beta} - P^{-1}Q\hat{\alpha} + \Delta y\right),\end{aligned}$$

where

$$E = \left(VW^{-1} + P^{-1}Q\right)^{-1}$$

and

(5.7)
$$D = \left(S^{-1}T + GZ^{-1}\right)^{-1}.$$

Eliminating Δt and Δw from (5.6), we arrive at last at the reduced KKT system:

(5.8)
$$\left[\begin{array}{c|c} -(H+D) & A^T \\ \hline A & E\end{array}\right]\left[\begin{array}{c}\Delta x\\ \hline \Delta y\end{array}\right] = \left[\begin{array}{c}\sigma - D(\hat{\nu} + S^{-1}T\hat{\tau})\\ \hline \rho - E(\hat{\beta} - P^{-1}Q\hat{\alpha})\end{array}\right].$$

Note that, since D and E are both *positive definite*, this reduced KKT system is quasidefinite. Also, the lower-left block is precisely the original constraint matrix, which as we've said before is a desirable property given our intention to extend LOQO to the realm of general nonlinear programming.

6. Computational Results.

In this section, we attempt to characterize the performance of LOQO. Our experiments were performed on a Silicon Graphics Indigo R4600 workstation having a 133 MHz clock, 160 MBytes of real memory, and running the IRIX 5.3 operating system.

We tested version 3.10 of LOQO which is written in ANSI C and was compiled using the **-O** optimization flag. Version 3.10 is the first version of LOQO that incorporates special techniques for general nonlinear programming. Namely, it uses a merit function to help choose an appropriate step length and it uses diagonal perturbation of the Hessian to guarantee a descent direction when problems are nonconvex. Associated with these new features are new parameters and new defaults for some old parameters. It turns out that some of these parameters need to be set differently if the problem is linear or convex quadratic as opposed to if it is a general nonlinear problem. An attempt was made to detect if a problem is linear and to set the defaults correctly in that case. It is a little harder to detect that a

problem is convex quadratic and so in this version of LOQO these problems get the same default values as general nonlinear optimization problems. Consequently, when a problem is known to be convex quadratic, some parameters need to be changed from their defaults. The specific parameters and their values are given below. Future releases of LOQO will automatically set the affected parameters correctly.

6.1. Linear Programming Problems. For linear programming problems, most parameters get the correct default values automatically but we forgot to disable diagonal perturbation for these problems. When the solution gets nearly optimal numerical issues often arise that makes the problem look slightly nonconvex and LOQO then tries diagonal perturbation. This behaviour needs to be disabled. It is done so by including the assertive parameter **`convex`** in the list of parameters. In a UNIX csh environment, this can be accomplished at the shell prompt by writing

```
setenv loqo_options "convex"
```

Our first set of test problems is taken from the standard NETLIB [6] test suite of linear programming problems. There are currently two collections of NETLIB problems. One contains problems that have optimal solutions and the other contains problems that are infeasible. Tables 2 and 3 show results for the collection having optimal solutions. All the problems except for **`dfl001`** and **`forplan`** were solved to an acceptable level of infeasibility and with eight figures of agreement between the primal and dual solution.

`Dfl001` is a badly scaled problem: the coefficients in the objective function range from about **`0.1`** to about **`1.0e+8`**. The preprocessing phase of LOQO does not attempt to scale the problem, hence there is no correction for this gross scale discrepancy. As a result, LOQO has numerical trouble with **`dfl001`** from the start. However, simply dividing the objective function by **`1.0e+6`** corrects this defect and LOQO is then able to solve the problem in 44 iterations. It took 5201.02 seconds to get this solution.

On iteration 30, **`forplan`** had 8 digits of agreement between primal and dual objective functions, relative primal infeasibility of **`1.89e-11`**, and dual infeasibility of **`9.17e-04`** before numerical troubles set in which on the 36th iteration led LOQO to conclude that the problem is dual infeasible. It should be emphasized that infeasibilities are measured in a relative sense and sometimes the relativizer might itself be of an inappropriate magnitude. This might have been the case for both **`forplan`** since the initial relative infeasibility was greater that **`1.00e+03`**, which is rather large.

Table 4 shows the number of iterations needed to detect infeasibility for the collection of infeasible problems. There are two ways in which infeasibility and unboundedness are detected. The first, as mentioned earlier, is if the primal or dual infeasibility increases from one iteration to the next. The other is if some iteration produces a step direction that points in an unbounded direction away from a feasible solution for either the primal or the dual. For several problems, infeasibility was detected in preprocessing. Two problems, **`cplex2`** and **`gosh`** ran 200 iterations and stopped at the iteration limit. Problem **`cplex2`** is interesting. It is so close to being feasible that it satisfies LOQO's test for feasibility (in both the primal and the dual), but runs into numerical trouble after achieving only five significant figures.

Problem	Solution Statistics				Performance Stats	
	Primal Objective	Dual Objective	Primal Infeas.	Dual Infeas.	Iterations	Time
25fv47	5.5018459e+03	5.5018459e+03	3.27e-13	2.53e-11	28	20.71
80bau3b	9.8722419e+05	9.8722419e+05	3.66e-14	3.58e-12	33	43.14
adlittle	2.2549496e+05	2.2549496e+05	5.71e-11	2.42e-16	16	0.18
afiro	-4.6475314e+02	-4.6475314e+02	4.25e-13	8.01e-11	13	0.06
agg	-3.5991767e+07	-3.5991767e+07	3.10e-12	6.93e-11	23	1.08
agg2	-2.0239252e+07	-2.0239252e+07	8.32e-15	1.72e-07	33	8.81
agg3	1.0312116e+07	1.0312116e+07	1.13e-15	7.89e-07	37	9.96
bandm	-1.5862802e+02	-1.5862802e+02	5.57e-12	2.52e-11	20	1.52
beaconfd	3.3592486e+04	3.3592485e+04	4.00e-11	8.61e-09	15	0.68
blend	-3.0812150e+01	-3.0812150e+01	7.17e-13	3.65e-12	17	0.29
bnl1	1.9776296e+03	1.9776295e+03	2.59e-11	6.22e-12	28	6.40
bnl2	1.8112365e+03	1.8112365e+03	1.69e-12	4.60e-13	35	83.66
boeing1	-3.3521357e+02	-3.3521357e+02	3.35e-12	4.99e-12	26	2.42
boeing2	-3.1501873e+02	-3.1501873e+02	1.65e-13	1.18e-11	19	0.55
bore3d	1.3730804e+03	1.3730804e+03	5.81e-11	4.25e-12	21	0.62
brandy	1.5185099e+03	1.5185099e+03	5.27e-11	1.93e-09	20	1.09
capri	2.6900129e+03	2.6900129e+03	6.50e-13	1.07e-10	27	1.91
cycle	-5.2263930e+00	-5.2263930e+00	8.26e-09	1.22e-12	31	38.24
czprob	2.1851967e+06	2.1851967e+06	8.14e-11	7.76e-14	35	10.63
d2q06c	1.2278421e+05	1.2278421e+05	1.91e-10	4.23e-10	34	166.05
d6cube	3.1549167e+02	3.1549167e+02	4.33e-13	3.96e-14	28	84.35
degen2	-1.4351780e+03	-1.4351780e+03	2.59e-10	3.88e-15	18	5.24
degen3	-9.8729398e+02	-9.8729400e+02	1.18e-08	1.90e-09	23	89.36
dfl001*	1.1266396e+01	1.1266396e+01	1.69e-10	2.25e-11	44	5201.02
e226	-1.8751929e+01	-1.8751929e+01	1.20e-14	2.52e-12	25	1.90
etamacro	-7.5571522e+02	-7.5571524e+02	5.12e-13	2.86e-14	29	4.84
fffff800	5.5567957e+05	5.5567956e+05	8.59e-15	2.22e-07	43	10.16
finnis	1.7279107e+05	1.7279106e+05	2.50e-11	4.17e-12	27	2.29
fit1d	-9.1463780e+03	-9.1463783e+03	6.91e-08	5.12e-13	18	4.99
fit1p	9.1463782e+03	9.1463781e+03	1.06e-11	4.05e-11	18	4.70
fit2d	-6.8464293e+04	-6.8464294e+04	1.71e-09	6.18e-14	25	185.01
fit2p	6.8464294e+04	6.8464292e+04	5.88e-12	1.62e-10	20	42.01
forplan*	-1.8207020e+02	-1.4821157e+10	3.14e-12	1.73e+06	39	4.63
ganges	-1.0958574e+05	-1.0958574e+05	1.17e-12	2.24e-11	34	17.94
gfrdpnc	6.9022360e+06	6.9022360e+06	8.88e-11	2.23e-13	33	4.00
greenbea	-7.2555247e+07	-7.2555248e+07	2.03e-07	2.72e-10	66	73.18
greenbeb	-4.3022602e+06	-4.3022603e+06	9.95e-11	9.18e-11	39	39.1
grow15	-1.0687094e+08	-1.0687094e+08	2.37e-08	1.41e-09	22	3.82
grow22	-1.6083434e+08	-1.6083434e+08	9.99e-07	1.86e-09	22	5.76
grow7	-4.7787812e+07	-4.7787812e+07	3.33e-07	1.32e-09	21	1.72
israel	-8.9664465e+05	-8.9664463e+05	2.02e-11	3.01e-10	33	1.54
kb2	-1.7499001e+03	-1.7499001e+03	2.52e-08	1.48e-10	18	0.16
lotfi	-2.5264702e+01	-2.5264703e+01	9.38e-13	7.03e-11	21	0.80
maros-r7	1.4971852e+06	1.4971852e+06	6.33e-14	3.90e-09	19	2370.61
maros	-5.8063744e+04	-5.8063744e+04	1.58e-10	3.05e-11	45	16.13
modszk1	3.2061973e+02	3.2061973e+02	1.36e-13	6.61e-15	36	6.88

TABLE 2. Solution and performance statistics for the NETLIB problems having optimal solutions (25fv47–modszk1).

Problem	Solution Statistics				Performance Stats	
	Primal Objective	Dual Objective	Primal Infeas.	Dual Infeas.	Iterations	Time
nesm	1.4076037e+07	1.4076036e+07	6.80e-11	2.95e-11	38	20.40
perold	-9.3807553e+03	-9.3807554e+03	1.38e-15	2.03e-10	49	26.15
pilot4	-2.5811392e+03	-2.5811393e+03	1.08e-12	6.71e-11	43	11.12
pilot87	3.0171035e+02	3.0171035e+02	2.29e-13	7.04e-12	47	1764.57
pilotja	-6.1131364e+03	-6.1131365e+03	4.50e-15	7.32e-11	52	65.06
pilotnov	-4.4972762e+03	-4.4972763e+03	1.41e-14	4.15e-10	29	38.99
pilots	-5.5748971e+02	-5.5748973e+02	1.57e-12	4.05e-10	40	313.60
pilotwe	-2.7201075e+06	-2.7201076e+06	1.26e-08	1.33e-13	47	19.03
recipe	-2.6661600e+02	-2.6661600e+02	4.45e-10	1.45e-10	14	0.32
sc105	-5.2202060e+01	-5.2202061e+01	1.95e-12	1.83e-11	15	0.21
sc205	-5.2202061e+01	-5.2202061e+01	7.53e-14	3.97e-11	17	0.51
sc50a	-6.4575077e+01	-6.4575077e+01	2.55e-13	9.35e-14	15	0.11
sc50b	-7.0000000e+01	-7.0000001e+01	3.13e-13	6.36e-14	13	0.10
scagr25	-1.4753433e+07	-1.4753433e+07	5.30e-12	5.46e-10	25	1.80
scagr7	-2.3313898e+06	-2.3313898e+06	2.25e-09	2.19e-09	18	0.34
scfxm1	1.8416759e+04	1.8416759e+04	1.64e-12	1.30e-09	26	2.19
scfxm2	3.6660262e+04	3.6660261e+04	2.32e-11	3.12e-10	26	4.48
scfxm3	5.4901255e+04	5.4901254e+04	2.27e-11	5.39e-10	26	6.95
scorpion	1.8781248e+03	1.8781248e+03	5.38e-10	1.90e-15	17	0.92
scrs8	9.0429696e+02	9.0429695e+02	3.64e-12	2.15e-15	23	3.20
scsd1	8.6666668e+00	8.6666667e+00	5.19e-12	3.08e-15	15	1.07
scsd6	5.0500000e+01	5.0500000e+01	4.36e-12	4.18e-15	17	2.17
scsd8	9.0500002e+02	9.0499999e+02	8.24e-11	4.48e-15	15	4.04
sctap1	1.4122500e+03	1.4122500e+03	3.60e-12	1.04e-15	22	1.39
sctap2	1.7248071e+03	1.7248071e+03	3.77e-12	1.25e-15	20	5.81
sctap3	1.4240000e+03	1.4240000e+03	5.74e-12	2.40e-12	21	8.48
seba	1.5711600e+04	1.5711600e+04	4.22e-11	1.50e-12	23	2.91
share1b	-7.6589318e+04	-7.6589319e+04	3.27e-11	9.12e-12	40	1.26
share2b	-4.1573224e+02	-4.1573224e+02	1.71e-11	2.03e-11	16	0.30
shell	1.2088253e+09	1.2088253e+09	1.41e-11	3.72e-10	37	5.71
ship04l	1.7933245e+06	1.7933245e+06	8.96e-09	9.01e-16	27	5.10
ship04s	1.7987147e+06	1.7987147e+06	6.21e-08	8.58e-16	26	3.28
ship08l	1.9090552e+06	1.9090552e+06	2.04e-09	1.29e-15	40	12.52
ship08s	1.9200982e+06	1.9200981e+06	3.31e-09	1.51e-15	33	5.25
ship12l	1.4701879e+06	1.4701879e+06	8.01e-10	1.46e-15	32	13.24
ship12s	1.4892361e+06	1.4892361e+06	3.08e-09	1.15e-15	29	5.64
sierra	1.5394356e+07	1.5394356e+07	6.21e-10	3.21e-10	26	9.20
stair	-2.5126695e+02	-2.5126695e+02	3.98e-12	9.24e-10	20	6.55
standata	1.2576995e+03	1.2576995e+03	1.12e-13	4.81e-12	19	2.17
standmps	1.4060175e+03	1.4060175e+03	3.51e-14	3.58e-12	28	3.94
stocfor1	-4.1131976e+04	-4.1131976e+04	1.03e-09	3.73e-12	19	0.32
stocfor2	-3.9024408e+04	-3.9024409e+04	5.65e-10	1.21e-10	27	12.59
tuff	2.9214778e-01	2.9214776e-01	5.41e-13	1.61e-12	24	3.75
vtpbase	1.2983146e+05	1.2983146e+05	1.77e-11	4.32e-09	24	0.35
wood1p	1.4429025e+00	1.4429024e+00	1.63e-11	8.84e-13	25	29.09
woodw	1.3044764e+00	1.3044763e+00	4.37e-11	4.29e-10	31	35.08

TABLE 3. Solution and performance statistics for the NETLIB problems having optimal solutions (nesm–woodw).

Problem	Iterations	Problem	Iterations	Problem	Iterations
bgdbg1	0	ex73a	89	klein3	50
bgetam	52	forest6	15	mondou2	40
bgindy	83	galenet	0	pang	46
bgprtr	10	gosh	200	pilot4i	0
box1	64	gran	122	qual	93
ceria3d	0	greenbea	0	reactor	0
chemcom	10	itest2	0	refinery	70
cplex1	60	itest6	0	vol1	79
cplex2	200	klein1	48	woodinfe	0
ex72a	85	klein2	40		

TABLE 4. Iterations to detect infeasibility for the infeasible NETLIB problems. Note, 0 indicates that infeasibility was detected in preprocessing, whereas 200 indicates that the iteration limit was reached without detecting infeasibility.

6.2. Quadratic Programming Problems. As mentioned at the beginning of this section, in LOQO version 3.10, parameter values have different defaults depending on whether a problem is linear or nonlinear. Consequently, quadratic programming problems are treated as general nonlinear problems even though the appropriate default values for linear programming work much better on these problems. Therefore, the runs documented in this section were all performed with the following nondefault parameter settings:

- **convex**: ensures that none of the special code for nonconvex nonlinear programming is called.
- **bndpush=100**: ensures that initial values are sufficiently far removed from their bounds.
- **honor_bnds=0**: allows variables to violate their bounds initially.
- **pred_corr=1**: enables the predictor-corrector method.
- **mufactor=0**: sets the predictor direction to the primal-dual affine-scaling direction.

Future releases of LOQO will ensure that these are the defaults for convex quadratic programming problems (as was the case in earlier releases).

We used the repository of convex quadratic programming problems collected by Maros and Mészáros [16]. It should be noted that the problems stored in this repository are stored in an extended form, called QPS, of the industry standard MPS format. In MPS and QPS format, a constant term for the objective function is stored in the right-hand side section of the file. Unfortunately, LOQO does not pick this constant out from this section and so the objective function values reported below don't match those given in [16]. Spot checks verified that this is the sole source of discrepancy between our values and those reported in [16].

Some of the problems in the repository are quadratic variants of problems from the NETLIB collection. Results for these problems are shown in Table 5. Of the 46 problems in this test set only two (**forplan** and **pivotnov**) failed to achieve the conditions for the stopping rule (which is relative primal and dual infeasibilities less that **1.0e-6** and 8 digits of agreement between primal and dual objective functions). On iteration 30, **forplan** had 8 digits of agreement between primal and dual objective functions, relative primal infeasibility of **1.89e-11**, and dual infeasibility of **9.17e-04** before numerical troubles set in which on the 36th iteration led LOQO to conclude that the problem is

dual infeasible. Similarly on iteration 48, **pilotnov** had 6 figures of agreement, primal infeasibility of **1.43e-08**, and dual infeasibility of **1.46e-02**, after which numerical difficulties set in and LOQO was never able to find a dual feasible solution. As we saw with the linear programming version of **forplan**, normalizer used in the relativization might be of the wrong magnitude. This might have been the case for both **forplan** and **pilotnov** since the initial relative infeasibility was in both cases greater that **1.00e+03**.

Another collection of problems in the repository come from the CUTE set of problems [4]. Results for these problems are shown in Tables 6 and 7. All 76 problems in this set solve to optimality.

Problem	Solution Statistics				Performance Stats	
	Primal Objective	Dual Objective	Primal Infeas.	Dual Infeas.	Iters.	Time
25fv47	1.3744448e+07	1.3744447e+07	3.35e-09	7.49e-08	42	166.03
adlittle	4.8031886e+05	4.8031886e+05	2.31e-10	4.14e-11	17	0.22
afiro	-1.5907818e+00	-1.5907818e+00	3.92e-15	1.20e-10	16	0.08
bandm	1.6352342e+04	1.6352342e+04	5.37e-10	7.53e-10	23	2.24
beaconfd	1.6471207e+05	1.6471207e+05	3.67e-09	6.17e-08	14	1.06
bore3d	3.1002009e+03	3.1002008e+03	3.02e-07	3.52e-10	23	1.21
brandy	2.8375115e+04	2.8375115e+04	1.62e-09	8.47e-08	20	1.47
capri	6.6793293e+07	6.6793293e+07	3.62e-12	2.36e-07	81	7.12
e226	2.0554043e+02	2.0554043e+02	2.89e-12	7.40e-12	23	2.22
etamacro	8.6760369e+04	8.6760369e+04	3.18e-12	3.00e-10	44	52.42
fffff800	8.7314749e+05	8.7314751e+05	7.32e-12	5.89e-07	39	17.89
forplan*	7.4934651e+09	3.2478707e+09	2.71e-12	4.92e+05	36	4.33
gfrd-pnc	1.0079058e+11	1.0079058e+11	3.88e-07	6.66e-09	38	4.78
grow15	-1.0169364e+08	-1.0169364e+08	2.37e-07	1.10e-09	26	4.44
grow22	-1.4962895e+08	-1.4962896e+08	4.09e-07	1.55e-09	29	7.69
grow7	-4.2798714e+07	-4.2798714e+07	2.14e-07	1.46e-09	26	2.13
israel	2.5347838e+07	2.5347838e+07	4.49e-13	1.86e-10	56	2.87
pilotnov*	4.7285893e+06	4.7285752e+06	2.48e-07	2.23e+03	200	294.00
recipe	-2.6661600e+02	-2.6661600e+02	4.99e-12	1.75e-11	17	0.46
sc205	-5.8139535e-03	-5.8139535e-03	5.87e-16	5.33e-07	56	1.82
scagr25	2.0173794e+08	2.0173794e+08	1.89e-10	3.49e-09	27	1.91
scagr7	2.6865949e+07	2.6865949e+07	2.24e-10	4.44e-09	24	0.43
scfxm1	1.6882691e+07	1.6882691e+07	5.66e-09	6.11e-07	36	3.27
scfxm2	2.7776161e+07	2.7776160e+07	6.53e-09	7.02e-07	41	7.60
scfxm3	3.0816353e+07	3.0816353e+07	8.42e-09	9.01e-07	42	11.81
scorpion	1.8805096e+03	1.8805095e+03	1.27e-09	2.76e-12	18	1.03
scrs8	9.0456001e+02	9.0456001e+02	4.09e-12	1.79e-13	25	3.54
scsd1	8.6666668e+00	8.6666667e+00	3.99e-12	8.93e-14	16	1.38
scsd6	5.0808214e+01	5.0808214e+01	2.48e-12	3.59e-13	19	3.07
scsd8	9.4076358e+02	9.4076357e+02	4.15e-11	3.59e-12	16	5.30
sctap1	1.4158611e+03	1.4158611e+03	3.77e-11	6.69e-12	21	1.33
sctap2	1.7350265e+03	1.7350265e+03	2.37e-11	2.33e-12	20	7.08
sctap3	1.4387547e+03	1.4387547e+03	2.66e-11	2.20e-12	21	10.35
seba	8.1481799e+07	8.1481800e+07	6.62e-08	2.04e-09	40	5.65
share1b	7.2007834e+05	7.2007833e+05	2.07e-10	3.54e-09	46	1.45
share2b	1.1703692e+04	1.1703692e+04	1.26e-09	1.13e-08	24	0.39
shell	1.5726368e+12	1.5726368e+12	9.26e-08	2.79e-07	57	90.00
ship04l	2.4200155e+06	2.4200155e+06	1.46e-08	6.26e-11	30	6.28
ship04s	2.4249936e+06	2.4249937e+06	1.87e-08	8.33e-11	29	4.32
ship08l	2.3760406e+06	2.3760406e+06	4.52e-09	1.26e-11	33	140.80
ship08s	2.3857286e+06	2.3857287e+06	1.38e-07	3.78e-10	31	21.60
ship12l	3.0188763e+06	3.0188764e+06	1.90e-07	4.70e-10	36	240.48
ship12s	3.0569622e+06	3.0569622e+06	2.13e-08	5.23e-11	35	25.56
sierra	2.3750458e+07	2.3750458e+07	5.54e-11	2.49e-11	33	11.77
stair	7.9854527e+06	7.9854528e+06	3.41e-09	1.45e-07	35	5.56
standata	6.4118384e+03	6.4118384e+03	2.23e-12	2.44e-11	21	2.63

TABLE 5. Performance statistics for quadratic variants of the feasible NETLIB problems.

Problem	Solution Statistics				Performance Stats	
	Primal Objective	Dual Objective	Primal Infeas.	Dual Infeas.	Iters.	Time
aug2d	1.6775117e+06	1.6775118e+06	1.72e-08	1.74e-08	10	57.29
aug2dc	1.8082680e+06	1.8082681e+06	1.81e-08	1.82e-08	10	57.13
aug2dcqp	6.4880347e+06	6.4880347e+06	1.61e-08	6.12e-09	16	87.40
aug2dqp	6.2271119e+06	6.2271119e+06	4.30e-08	1.65e-08	16	87.27
aug3d	-7.8243228e+02	-7.8243227e+02	1.87e-07	2.28e-07	8	7.93
aug3dc	-1.1652376e+03	-1.1652376e+03	1.87e-07	1.91e-07	8	7.93
aug3dcqp	-9.4313784e+02	-9.4313786e+02	1.02e-12	3.52e-13	17	15.61
aug3dqp	-6.6126233e+02	-6.6126233e+02	3.17e-15	1.79e-14	19	17.33
cvxqp1_l	1.0870480e+08	1.0870480e+08	5.09e-11	7.74e-07	63	30660.00
cvxqp1_m	1.0875115e+06	1.0875115e+06	1.15e-08	7.98e-09	32	64.64
cvxqp1_s	1.1590718e+04	1.1590718e+04	1.31e-09	6.08e-11	13	0.27
cvxqp2_l	8.1842458e+07	8.1842458e+07	4.12e-09	7.49e-07	25	7508.00
cvxqp2_m	8.2015543e+05	8.2015543e+05	3.31e-09	1.28e-09	16	20.79
cvxqp2_s	8.1209405e+03	8.1209403e+03	1.96e-09	1.25e-09	11	0.20
cvxqp3_l	1.1571110e+08	1.1571110e+08	1.15e-10	7.22e-07	47	26200.00
cvxqp3_m	1.3628287e+06	1.3628287e+06	3.22e-09	1.36e-07	48	111.60
cvxqp3_s	1.1943432e+04	1.1943432e+04	4.38e-10	5.82e-09	13	0.30
dtoc3	2.3526248e+02	2.3526248e+02	1.10e-08	1.36e-08	11	25.51
dual1	3.5012967e-02	3.5012965e-02	7.34e-14	2.19e-14	18	0.97
dual2	3.3733677e-02	3.3733672e-02	2.39e-13	3.10e-15	16	1.14
dual3	1.3575584e-01	1.3575583e-01	2.94e-13	6.07e-15	16	1.66
dual4	7.4609085e-01	7.4609082e-01	2.01e-13	6.02e-16	15	0.60
dualc1	6.1552508e+03	6.1552508e+03	3.06e-10	3.94e-15	36	0.94
dualc2	3.5513077e+03	3.5513077e+03	9.93e-11	3.31e-15	34	0.78
dualc5	4.2723233e+02	4.2723231e+02	1.40e-08	1.24e-11	15	0.49
dualc8	1.8309359e+04	1.8309359e+04	2.30e-09	1.51e-13	20	1.25
genhs28	9.2717369e-01	9.2717369e-01	1.29e-07	5.54e-07	8	0.02
gouldqp2	1.8427677e-04	1.8425871e-04	9.25e-13	5.28e-13	17	1.46
gouldqp3	-2.9647837e+04	-2.9647837e+04	5.72e-09	3.22e-13	12	1.27
hs118	6.6482045e+02	6.6482044e+02	3.79e-11	1.94e-10	15	0.05
hs21	4.0000001e-02	3.9999999e-02	2.66e-14	1.78e-13	13	0.03
hs268	-1.4463000e+04	-1.4463000e+04	5.71e-10	2.18e-12	10	0.03
hs35	-8.8888889e+00	-8.8888889e+00	1.04e-16	6.19e-17	12	0.03
hs35mod	-8.7499999e+00	-8.7500001e+00	5.01e-14	1.33e-14	14	0.03
hs51	-6.0000000e+00	-6.0000000e+00	6.11e-08	5.42e-08	8	0.03
hs52	-6.7335244e-01	-6.7335244e-01	1.80e-08	3.19e-09	9	0.03
hs53	-1.9069767e+00	-1.9069768e+00	6.77e-11	2.92e-14	11	0.03
hs76	-4.6818182e+00	-4.6818182e+00	1.05e-16	5.76e-17	13	0.03

TABLE 6. Performance statistics for quadratic CUTE problems (aug2d to hs76).

REFERENCES

[1] E.D. Andersen, J. Gondzio, Cs. Mészáros, and X. Xu. Implementation of interior-point methods for large scale linear programming. In T. Terlaky, editor, *Interior point methods of mathematical programming*, pages 189–252. Kluwer Academic Publishers, Dordrecht and Boston, 1996.

[2] P.T. Boggs, P.D. Domich, J.E. Rogers, and C. Witzgall. An interior-point method for linear and quadratic programming problems. Technical Report NISTIR-4556, National Institute of Standards and Technology, 1991.

Problem	Solution Statistics				Performance Stats	
	Primal Objective	Dual Objective	Primal Infeas.	Dual Infeas.	Iters.	Time
hues-mod	3.4824464e+07	3.4824464e+07	1.45e-12	3.22e-09	32	44.55
huestis	3.4824464e+11	3.4824464e+11	8.87e-11	2.22e-07	19	36.05
ksip	5.7579794e-01	5.7579794e-01	6.22e-14	4.18e-12	22	7.62
liswet1	-2.4892877e+03	-2.4892876e+03	1.23e-11	4.83e-11	46	75.94
liswet10	-2.4764642e+03	-2.4764642e+03	7.69e-11	4.17e-11	57	97.47
liswet11	-2.4762261e+03	-2.4762260e+03	1.87e-11	1.09e-11	44	71.08
liswet12	-7.8882259e+02	-7.8882257e+02	2.18e-10	2.45e-10	80	148.86
liswet2	-1.6669919e+03	-1.6669919e+03	6.92e-10	2.93e-11	14	19.77
liswet3	-1.0002588e+03	-1.0002588e+03	1.84e-10	9.76e-12	16	22.55
liswet4	-7.1451587e+02	-7.1451589e+02	3.62e-10	2.25e-11	16	22.52
liswet5	-1.5976166e+04	-1.5976166e+04	5.06e-09	6.94e-11	13	18.35
liswet6	-2.1622242e+03	-2.1622243e+03	1.35e-09	4.93e-11	14	19.77
liswet7	-2.0264192e+03	-2.0264191e+03	1.61e-11	3.38e-10	33	58.89
liswet8	-1.8107900e+03	-1.8107899e+03	3.29e-10	2.01e-10	69	131.86
liswet9	-5.6200874e+02	-5.6200874e+02	3.41e-10	2.06e-10	77	144.65
lotschd	2.3984159e+03	2.3984159e+03	1.97e-11	1.42e-09	11	0.03
mosarqp1	-9.5287544e+02	-9.5287545e+02	4.14e-16	2.40e-13	16	5.52
mosarqp2	-1.5974821e+03	-1.5974821e+03	4.38e-16	1.28e-12	14	3.80
powell20	5.2089583e+10	5.2089583e+10	3.22e-13	5.21e-07	92	155.34
primal1	-3.5012964e-02	-3.5012968e-02	1.50e-12	7.71e-16	21	3.02
primal2	-3.3733664e-02	-3.3733680e-02	1.48e-12	5.96e-15	19	4.11
primal3	-1.3575582e-01	-1.3575585e-01	1.08e-12	3.45e-15	19	9.66
primal4	-7.4609084e-01	-7.4609084e-01	1.71e-13	3.12e-15	17	6.37
primalc1	-6.1552508e+03	-6.1552508e+03	1.37e-15	3.61e-10	29	0.87
primalc2	-3.5513076e+03	-3.5513076e+03	3.64e-14	4.65e-09	27	0.68
primalc5	-4.2723233e+02	-4.2723233e+02	2.29e-13	2.71e-10	16	0.59
primalc8	-1.8309430e+04	-1.8309430e+04	3.49e-13	5.67e-09	19	1.39
qpcblend	-7.8425371e-03	-7.8425507e-03	3.29e-15	6.08e-14	21	0.33
qpcboei1	1.1503914e+07	1.1503914e+07	1.53e-09	2.20e-08	31	3.71
qpcboei2	8.1719622e+06	8.1719622e+06	7.29e-13	1.80e-07	40	1.58
qpcstair	6.2043874e+06	6.2043874e+06	1.49e-08	6.36e-07	38	6.37
s268	-1.4463000e+04	-1.4463000e+04	5.71e-10	2.18e-12	10	0.03
stcqp1	1.5514355e+05	1.5514355e+05	1.06e-10	2.92e-11	12	33.19
stcqp2	2.2327313e+04	2.2327313e+04	5.38e-12	9.73e-12	13	76.20
tame	0.0000000e+00	-2.0360249e-08	5.94e-11	6.37e-11	11	0.02
ubh1	1.1160008e+00	1.1160008e+00	4.78e-09	3.80e-12	33	85.48
yao	-7.5420744e+01	-7.5420744e+01	3.49e-12	2.95e-11	47	14.52
zecevic2	-4.1250000e+00	-4.1250001e+00	1.41e-11	9.99e-12	12	0.03

TABLE 7. Performance statistics for quadratic CUTE problems (hues-mod to zecevic2.

[3] M.G. Breitfeld and D.F. Shanno. Preliminary computational experience with modified log-barrier functions for large-scale nonlinear programming. In W.W. Hager, D.W. Hearn, and P.M. Pardalos, editors, *Large scale optimization: state of the art*, pages 45–66. Kluwer Academic Publishers, 1994.

[4] A.R. Conn, N. Gould, and Ph.L. Toint. Constrained and unconstrained testing environment. http://www.dci.clrc.ac.uk/Activity.asp?C

[5] R. Fourer and S. Mehrotra. Solving symmetric indefinite systems in an interior point method for linear programming. *Mathematical Programming*, 62:15–40, 1991.

[6] D.M. Gay. Electronic mail distribution of linear programming test problems. *Mathematical Programming Society COAL Newslettter*, 13:10–12, 1985.
[7] A. George and J.W.H. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Review*, 31:1–19, 1989.
[8] P.E. Gill, W. Murray, D.B. Ponceleón, and M.A. Saunders. Primal-dual methods in linear programming. Technical Report SOL 91-3, Systems Optimization Laboratory, Stanford University, Stanford, CA, April 1991.
[9] 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.
[10] P.E. Gill, W. Murray, D.B. Ponceleón, and M.A. Saunders. Preconditioners for indefinite systems arising in optimization. *SIAM Journal on Matrix Analysis and Applications*, 13(1):292–311, 1992.
[11] M. Kojima, N. Megiddo, and S. Mizuno. A primal-dual infeasible-interior-point algorithm for linear programming. *Mathematical Programming*, 61:263–400, 1993.
[12] I.J. Lustig. Feasibility issues in a primal-dual interior-point method for linear programming. *Mathematical Programming*, 49(2):145–162, 1990.
[13] I.J. Lustig, R.E. Marsten, and D.F. Shanno. Computational experience with a primal-dual interior point method for linear programming. *Lin. Alg. and Appl.*, 152:191–222, 1991.
[14] I.J. Lustig, R.E. Marsten, and D.F. Shanno. On implementing Mehrotra's predictor-corrector interior point method for linear programming. *SIAM Journal on Optimization*, 2:435–449, 1992.
[15] 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.
[16] I. Maros and Cs. Mészáros. A repository of convex quadratic programming problems. Technical Report DOC 97/6, Imperial College, London, UK, Department of Computing, Imperial College, London, UK, 1997.
[17] N. Megiddo. Pathways to the optimal set in linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming*, pages 131–158, New York, 1989. Springer-Verlag.
[18] S. Mehrotra. On the implementation of a (primal-dual) interior point method. *SIAM Journal on Optimization*, 2:575–601, 1992.
[19] Cs. Mészáros. On the sparsity issues of interior point methods for quadratic programming. Technical Report WP 98-4, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, 1998.
[20] R.D.C. Monteiro and I. Adler. Interior path following primal-dual algorithms. part ii: Convex quadratic programming. *Mathematical Programming*, 44:43–66, 1989.
[21] R.D.C. Monteiro and F. Zhou. On superlinear convergence of infeasible-interior-point algorithms for linearly constrained convex programs. *Computational Optimization and Applications*, 8:245–262, 1997.
[22] S. Pissanetzky. *Sparse Matrix Technology*. Academic Press, 1984.
[23] F.A. Potra. A quadratically convergent predictor-corrector method for solving linear programs from infeasible starting points. *Mathematical Programming*, 67:383–406, 1994.
[24] J.E. Rogers, P.T. Boggs, and P.D. Domich. A predictor-corrector-O3D formulation for quadratic programming. Technical report, National Institute of Standards and Technology, 1994.
[25] 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.
[26] K. Tanabe. Centered newton method for linear programming: exterior point method (in japanese). In *Proc. Inst. Stat. Mathematics*, volume 37, pages 146–148, 1989.
[27] K. Turner. Computing projections for the Karmarkar algorithm. *Linear Algebra and Its Applications*, 152:141–154, 1991.
[28] R.J. Vanderbei. Affine scaling and linear programs with free variables. *Mathematical Programming*, 39:31–44, 1989.

[29] R.J. Vanderbei. An implementation of the minimum-degree algorithm using simple data structures. Technical report, AT&T Bell Laboratories, 1990.
[30] R.J. Vanderbei. Interior-point methods: algorithms and formulations. *ORSA J. on Computing*, 6:32–34, 1994.
[31] R.J. Vanderbei. LOQO: An interior point code for quadratic programming. Technical Report SOR 94-15, Princeton University, 1994.
[32] R.J. Vanderbei. Symmetric quasi-definite matrices. *SIAM Journal on Optimization*, 5(1):100–113, 1995.
[33] R.J. Vanderbei. *Linear Programming: Foundations and Extensions*. Kluwer Academic Publishers, 1996.
[34] R.J. Vanderbei and T.J. Carpenter. Symmetric indefinite systems for interior-point methods. *Mathematical Programming*, 58:1–32, 1993.
[35] R.J. Vanderbei, A. Duarte, and B. Yang. An algorithmic and numerical comparison of several interior-point methods. Technical Report SOR 94-05, Princeton University, 1994.
[36] R.J. Vanderbei and D.F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. Technical Report SOR-97-21, Statistics and Operations Research, Princeton University, 1997.
[37] S.J. Wright. An infeasible-interior-point algorithm for linear complementarity problems. *Mathematical Programming*, 67:29–52, 1994.
[38] S.J. Wright. *Primal-Dual Interior-Point Methods*. SIAM, Philadelphia, USA, 1996.
[39] Y. Zhang. On the convergence of a class of infeasible-interior-point methods for the horizontal linear complementarity problem. *SIAM J. on Optimization*, 4:208–227, 1994.

ROBERT J. VANDERBEI, PROGRAM IN STATISTICS AND OPERATIONS RESEARCH, PRINCETON UNIVERSITY, PRINCETON, NJ 08544
E-mail address: **rvdb@princeton.edu**