INTERIOR-POINT METHODS FOR NONCONVEX NONLINEAR PROGRAMMING: REGULARIZATION AND WARMSTARTS

HANDE Y. BENSON AND DAVID F. SHANNO

ABSTRACT. In this paper, we investigate the use of an exact primal-dual penalty approach within the framework of an interior-point method for nonconvex nonlinear programming. This approach provides regularization and relaxation, which can aid in solving ill-behaved problems and in warmstarting the algorithm. We present details of our implementation within the LOQO algorithm and provide extensive numerical results on the CUTEr test set and on warmstarting in the context of nonlinear, mixed integer nonlinear, and goal programming.

1. INTRODUCTION

A nonlinear programming problem (NLP), in its most general form, can be expressed as

$$\begin{array}{lll} \text{minimize} & f(x) & \\ \text{subject to} & 0 \leq h(x) \leq r & (1) \\ & l \leq x \leq u, & \end{array}$$

where $x \in \mathbb{R}^n$ are the decision variables, $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function, $h : \mathbb{R}^n \to \mathbb{R}^m$ are the constraint functions, $r \in \mathbb{R}^m$ is a nonnegative (possibly infinite) parameter, and $l \in \mathbb{R}^n$ and $u \in \mathbb{R}^n$ are lower and upper bounds on the decision variables, one or both of which can be infinite.

The algorithm we will use to solve such problems is the infeasible primal-dual interior-point method implemented in LOQO [22]. As such, we will assume that all objective and constraint functions are twice continuously differentiable. If f is convex, h_i are concave for $r_i = \infty$ and linear otherwise, then the problem is convex and the algorithm, if it terminates successfully, will locate a global optimum. If the problem is not convex, the algorithm will be searching for a local optimum.

Interior-point codes, such as LOQO [22], IPOPT [24] and KNITRO [19], have been shown in numerical studies such as [5] and [18] to be more efficient than other codes using sequential quadratic programming (SQP) or reduced gradient approaches, especially on large and sparse problems. Despite this computational advantage, however, ill-behaved problems and problems without Karush-Kuhn-Tucker (KKT) solutions have posed serious difficulties for claiming reliability of these codes. In [23], Wächter and Biegler showed that a class of interior-point methods become stuck at a nonoptimal, and in fact infeasible, point on a particular problem. In

Date: November 4, 2005.

Key words and phrases. interior-point methods, nonlinear programming, warmstarting, penalty methods.

Research of the first author is sponsored by ONR grant N00014-04-1-0145. Research of the second author is supported by NSF grant DMS-0107450.

[7], this phenomenon was called "jamming." A large set of problems known as mathematical programs with equilibrium constraints (MPECs) have been shown in [1] and others to have an unbounded set of optimal Lagrange multipliers, and in [2] it is shown that a primal-dual interior-point code such as LOQO will have convergence problems while looking for the analytic center of the face of optimal dual solutions. Additionally, proper stopping procedures, similar to the elastic mode of the SQP code SNOPT [12], need to be investigated for detecting lack of KKT solutions, whether it be from problems that are primal and/or dual infeasible or where the duality gap at the optimal solution is nonzero.

There have been several recent studies into using a penalty method approach for resolving some of these issues. As discussed in [2], an ℓ_1 penalty approach can be implemented within the framework of an interior-point code to resolve the issues of jamming, infeasibility identification, and the location of nonKKT solutions. This recent interest in penalty methods arose in the context of solving MPECs, and [1], [2], and [16] have proposed the use of these methods to resolve the issue of unbounded Lagrange multipliers. Numerical results presented in [2] and [16] show improved numerical performance on MPECs. Additionally, in [13], an interior-point method using an ℓ_1 penalty approach was proposed in the context of general NLP.

In the recent paper [4], the authors have shown that an exact primal-dual penalty approach can also be useful for warmstarting an interior-point method for linear programming. *Warmstarting* is the use of information obtained during the solution of an initial problem to solve subsequent, closely-related problems. When no such information is used, the new problem is solved from a *coldstart*. As demonstrated in [4], an interior-point method can become stuck at the initial problem's solution, thereby requiring more iterations than a coldstart to solve a problem. The relaxation aspect of a penalty approach is ideally suited to making iterates "unstuck," and the authors in [4] proposed a new primal-dual penalty framework so that the relaxation/bounding scheme of a penalty method could be applied on both the primal and the dual sides to accommodate various types of perturbations to the problem data.

In this paper, we will extend the exact primal-dual penalty approach of [4] to NLP and show that not only will this approach be useful for warmstarting NLPs, but that it will retain the theoretical advantages of the purely primal penalty method for solving ill-behaved problems. We have implemented the primal-dual penalty in LOQO, and we will present extensive numerical results on the CUTEr [9] test set, as well as on additional warmstarting examples. As shown in [7], LOQO is a very robust code, and, therefore, we will use the algorithm without modification as much as possible and employ the penalty approach only as necessary. Therefore, we start by presenting the interior-point algorithm currently implemented in LOQO.

2. LOQO: An Interior-Point Code for Nonlinear Programming

We begin with an overview of the LOQO algorithm. A more detailed explanation can be found in [22].

We concentrate here on the inequality constrained problem

$$\begin{array}{lll} \text{minimize} & f(x) & \\ \text{subject to} & h(x) & \geq \quad 0. \end{array} \tag{2}$$

This simplification is made for purely pedagogical reasons, and a description of how to handle more general problems will follow.

First, slack variables, w, are added to each of the constraints to convert them to equalities, and the nonnegativity constraints on the slack variables are eliminated by placing them in a barrier objective function, giving the Fiacco and McCormick [10] logarithmic barrier problem:

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

where $w \in \mathbb{R}^m$. The scalar μ is called the *barrier parameter*. Now that we have an optimization problem with no inequalities, we form the Lagrangian

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

where $y \in \mathbb{R}^m$ are called the *Lagrange multipliers* or the *dual variables*.

In order to achieve a stationary point of the Lagrangian function, we solve the primal-dual system arising from the first-order optimality conditions:

$$\begin{aligned} \nabla f(x) - A(x)^T y &= 0 \\ -\mu e + WYe &= 0 \\ h(x) - w &= 0 \end{aligned} \tag{3}$$

In order to solve this system, we use Newton's Method. Doing so gives the following system to solve:

$$\begin{bmatrix} -H(x,y) & A(x)^T \\ A(x) & D \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \sigma \\ \rho + D\gamma \end{bmatrix} \tag{4}$$

$$\Delta w = WY^{-1}(\gamma - \Delta y), \tag{5}$$

where

$$\begin{aligned} H(x,y) &= \nabla^2 f(x) - \textstyle\sum_{i=1}^{m} y_i \nabla^2 h(x) \\ D &= WY^{-1} \\ \sigma &= \nabla f(x) - A(x)^T y \\ \gamma &= -\mu W^{-1} e + y \\ \rho &= -h(x) + w. \end{aligned}$$

This system is solved using LDL^T *factorization*, which is a modified version of Cholesky factorization, and then performing a backsolve to obtain the step directions. If the entries of the diagonal matrix D have the wrong sign, the problem is nonconvex, and $H(x, y)$ is replaced in the current iteration by $H(x, y) + \lambda I$, where I is the $n \times n$ identity matrix and λ is a sufficiently large perturbation parameter.

The algorithm starts at an initial solution $(x^{(0)}, w^{(0)}, y^{(0)})$ and proceeds iteratively toward the solution through a sequence of points which are determined by the search directions obtained from the reduced KKT system as follows:

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

where $0 < \alpha \leq 1$ is the *steplength* and the superscripts denote the iteration number. $\alpha^{(k)}$ is chosen to ensure that $w^{(k+1)} > 0$, $y^{(k+1)} > 0$, and either the barrier function or the primal infeasibility is reduced. Further details of steplength control can be found in [5].

LOQO concludes that it has reached an optimal solution when the first order conditions (3) with $\mu = 0$ are satisfied within a tolerance level, which is set to 10^{-7} by default.

2.1. **Problems in General Form.** Even though we described the algorithm for the inequality constrained problem (2), LOQO can handle problems involving variable bounds, as well as range and equality constraints. A variable with the bounds

$$l \leq x \leq u,$$

is converted to

$$\begin{array}{lcl} x - g & = & l \\ x + t & = & u \\ g, t & \geq & 0. \end{array} \tag{6}$$

A range constraint of the form

$$0 \leq h(x) \leq r$$

is converted to

$$\begin{array}{lcl} h(x) - w & = & 0 \\ w + p & = & r \\ w, p & \geq & 0. \end{array} \tag{7}$$

An equality constraint is treated as a range constraint as well, with $r = 0$. Therefore, each equality constraint has two slack variables, w, and p, associated with it. This has important implications for the Cholesky factorization, as LOQO performs a *dual ordering*, where the initial pivots are picked from the lower right block of the matrix in (4). According to the computational experience reported in [20], a dual ordering can serve to decrease computation time per iteration and provide a more numerically stable ordering. Including slack variables for equality constraints (rather than not slacking them at all) ensures that there will always be strictly positive entries on the diagonal of the lower right block, regardless of the type of constraint, so that a dual ordering can always be performed.

2.2. **The Dual Problem.** Although dual feasibility is included among the first-order conditions (3), the dual objective function, and thereby the form of the dual problem for an NLP is not generally of concern to solvers (unlike in linear programming). Having its roots in linear and quadratic programming, LOQO works with the dual problem

$$\begin{array}{lll} \text{maximize} & f(x) - \nabla f(x)^T x - (h(x) - A(x))^T y & \\ \text{subject to} & \nabla f(x) - A(x)^T y & = \ 0 \\ & y & \geq \ 0 \end{array} \tag{8}$$

in the context of NLP. In fact, it can be seen that given a primal and dual feasible solution, the duality gap reduces to the complementarity condition. In the past, the duality gap was a measure of optimality in LOQO for NLPs, as well, but in the current implementation, it has been replaced by the complementarity condition, as described above. We have chosen to include the dual problem here in order to illustrate the symmetry of the primal-dual penalty approach in the next section.

3. An Exact Primal-Dual Penalty Approach to Regularization and Relaxation

In this paper, we will use the primal-dual penalty approach introduced in [4] for warmstarting interior-point methods for linear programming. The classical penalty method as described in [11] is purely primal, and we will need the dual aspect mainly for detecting dual infeasibility and for warmstarting.

The primal penalty problem corresponding to (2) has the form

$$\begin{array}{ll} \text{minimize} & f(x) + c^T\xi \\ \text{subject to} & h(x) + w = 0 \\ & -\xi \le w \le b \\ & \xi \ge 0, \end{array} \tag{9}$$

where $\xi \in \mathbb{R}^m$ are the new decision variables corresponding to the relaxation of the constraints and $c \in \mathbb{R}^m$ are their penalty parameters. The new upper bounds, b, on the slack variables are the dual penalty parameters, and the dual penalty problem, then, has the form

$$\begin{array}{ll} \text{maximize} & f(x) - \nabla f(x)^T x - (h(x) - A(x))^T y - b^T\psi \\ \text{subject to} & \nabla f(x) - A(x)^T y = 0 \\ & -\psi \le y \le c - \psi \\ & \psi \ge 0, \end{array} \tag{10}$$

where $\psi \in \mathbb{R}^m$ are the dual relaxation variables. The relaxation of the primal slack variables and the penalty term in the primal objective function result in the bounding of the Lagrange multipliers, as in the purely primal approach. The bounding of the primal slack variables has the reverse effect on the dual and results in the relaxation of the Lagrange multipliers from below by ψ and introduces a dual penalty term. The upper bound on the Lagrange multipliers, $c - \psi$ differs from the purely primal approach, which would have c as the upper bound, but as ψ is driven to 0, the upper bound can be kept sufficiently large. The primal-dual penalty approach (9)-(10) is *exact*, that is, given an optimal primal-dual pair (x^*, y^*) to (2)-(8), the pair (9)-(10) also has a solution at (x^*, y^*) for sufficiently large but finite b and c. Note that we have chosen to allow the slack variables and the Lagrange multipliers to go negative rather than directly relax the constraints, as discussed in [11], [13], and [2]. This will allow for better warmstarting by changing the terms in the barrier objective of our infeasible interior-point method.

Solving (9) instead of (2) has several benefits, including the relaxation of the problem to achieve feasibility and the bounding of the primal and dual variables. Many of these benefits were illustrated and/or proved in [2] for the purely primal penalty, and we will provide a quick overview here. However, we will first discuss the application of the algorithm of Section 2 to (9).

3.1. The Penalty Approach within the IPM Framework.

We follow the development of Section 2 to present the algorithm to solve (9). The logarithmic barrier problem associated with (9) is

$$\begin{array}{ll} \text{minimize} & f(x) + c^T\xi - \mu\sum_{i=1}^m \log(\xi_i) - \mu\sum_{i=1}^m \log(w_i + \xi_i) - \mu\sum_{i=1}^m \log(b_i - w_i) \\ \text{subject to} & h(x) - w = 0, \end{array}$$

where $w \in \mathbb{R}^m$ are the slack variables and μ is the (nonnegative) barrier parameter. The first-order conditions for the barrier problem are as follows:

$$\nabla f(x) - A(x)y = 0 \tag{11}$$
$$h(x) - w = 0 \tag{12}$$
$$\mu e - (W + \Xi)(Y + \Psi)e = 0 \tag{13}$$
$$\mu e - \Xi(C - Y - \Psi)e = 0 \tag{14}$$
$$\mu e - \Psi(B - W)e = 0 \tag{15}$$

where C, B, Ξ and Ψ are the diagonal matrices with the entries of c, b, ξ, and ψ, respectively. (12) and (13) correspond to the dual and primal feasibility conditions of the original problem. The new complementarity conditions, (14)-(15), perturb the original complementarity condition and bound the primal slacks and the Lagrange multipliers above by the primal and the dual penalty parameters.

Applying Newton's Method to (12)-(15) and eliminating Δw, $\Delta\xi$, and $\Delta\psi$, the the reduced KKT system for the primal-dual penalty problem has the same form as (4) with

$$\begin{aligned} D &= \left(\left((Y + \Psi)^{-1}(W + \Xi) + \Xi(C - Y - \Psi)^{-1} \right)^{-1} + \Psi(B - W)^{-1} \right)^{-1} \\ \gamma &= \left((Y + \Psi)^{-1}(W + \Xi) + \Xi(C - Y - \Psi)^{-1} \right)^{-1} \\ &\quad \left(\mu(Y + \Psi)^{-1}e - \mu(C - Y - \Psi)^{-1}e - w \right) - \left(\mu(B - W)^{-1}e - \psi \right) \end{aligned} \tag{16}$$

and

$$\begin{aligned} \Delta w &= -(W + \Xi)(Y + \Psi)^{-1}(\gamma_w + \Delta y + \Delta\psi) - \Delta\xi, \\ \Delta\xi &= (C - Y - \Psi)^{-1}\Xi(\Delta y + \Delta\psi) - \gamma_\xi, \\ \Delta\psi &= -\left((Y + \Psi)^{-1}(W + \Xi) + (B - W)\Psi^{-1} + (C - Y - \Psi)^{-1}\Xi \right)^{-1} \\ &\quad \left(b - \mu\Psi^{-1}e - \mu(Y + \Psi)^{-1}e + \mu(C - Y - \Psi)^{-1}e \right) \\ &\quad - \left((Y + \Psi)^{-1}(W + \Xi) + (B - W)\Psi^{-1} + (C - Y - \Psi)^{-1}\Xi \right)^{-1} \\ &\quad \left((Y + \Psi)^{-1}(W + \Xi) + (C - Y - \Psi)^{-1}\Xi \right) \Delta y \end{aligned}$$

There are several things to note about this system. First, it has the same sparsity structure as (4), the reduced KKT system of the original problem. Therefore, there would be no significant computational effort to switch back and forth between the solutions of these two related systems. Second, by adding the additional terms to the lower right diagonal of the matrix, the penalty method serves to regularize the system. This is achieved using the scheme of relaxing/penalizing via a variable and as the variable goes to 0, so does the regularization. Third, the original term in the lower right diagonal, WY^{-1}, has been changed to D. Doing so will allow for positive entries if the slack variables prematurely go to 0 and aid in warmstarting by eliminating 0 diagonal entries. This is essential for our implementation in the solver LOQO as it uses a dual ordering in its Cholesky factorization routine.

The steplength, $\alpha^{(k)}$, at each iteration k is chosen to ensure that

$$\begin{aligned} w^{(k+1)} + \xi^{(k+1)} &> 0, \\ y^{(k+1)} + \psi^{(k+1)} &> 0, \\ \xi^{(k+1)} &> 0, \\ \psi^{(k+1)} &> 0, \\ b - w^{(k+1)} &> 0, \\ c - y^{(k+1)} - \psi^{(k+1)} &> 0, \end{aligned}$$

and either the barrier function or the infeasibility is reduced. The algorithm concludes that it has reached an optimum when the optimality conditions for the original problem are satisfied.

3.2. **Problems in General Form.** The penalty problem, (9), is obtained from (2) by relaxing the nonnegativities of the slack variables. This approach will be used for problems in general form. Slack variables g and t on the variable bounds as well as w and p on the range and equality constraints are all allowed to take on negative values, at a penalty cost.

3.3. **Setting and Updating the Penalty Parameters.** The most important aspect of setting the initial values of the penalty parameters is to ensure that they are sufficiently large to allow for the current iterate. Therefore, the penalty parameters are set as follows:

$$\begin{aligned} b &= 10(w+\kappa) \\ c &= 10(y+\psi+\kappa), \end{aligned}$$

where κ is a constant with a default value of 1. The relaxation variables are initialized as

$$\begin{aligned} \xi &= \max(h(x)-w,0)+\tau \\ \psi &= \tau, \end{aligned}$$

where τ is a small parameter, currently set to $10^{-5}M$, where M is the greater of 1 and the largest primal or dual slack value. These initializations are generally sufficient for switching to the penalty method when we are far from the optimum, and close to the solution, especially after a warmstart, they allow starting the penalty method without moving the iterates too far from the current point. Note that the relaxation is performed using a variable, so if a larger relaxation is needed, the variables will move as necessary.

Since these initial values of the penalty parameters may not be large enough to admit the optimal solution, we also need an updating scheme for these parameters. Given the relaxations, a pair of optimal solutions can always be found for (9)-(10), and one possible "static" updating scheme is to solve a problem to optimality and to increase the penalty parameters if their corresponding relaxation variables are not sufficiently close to zero. However, this may require multiple solves of a problem and substantially increase the number of iterations necessary to find the optimal solution. We have instead implemented a "dynamic" updating scheme in LOQO, where the penalty parameters are checked at the end of each iteration and updated as follows:

$$\begin{array}{llll} \text{If} & w_i^{(k)} > 0.9b_i^{(k)}, & \text{then} & b_i^{(k+1)} = 10b_i^{(k)}, \qquad i=1,\ldots,m. \\ \text{If} & y_i^{(k)} + \psi_i^{(k)} > 0.9c_i^{(k)}, & \text{then} & c_i^{(k+1)} = 10c_i^{(k)}, \qquad i=1,\ldots,m. \end{array}$$

3.4. **Benefits of a Penalty Approach.** Many of the benefits of using an ℓ_1 penalty approach have been discussed in literature. Although we are relaxing the nonnegativities on the slack variables, rather than the constraints themselves directly, and we are using a penalty approach on the dual side, the same theoretical properties follow. We will now summarize them here.

3.4.1. *Regularization and Jamming.* As shown in [23], an interior-point method can become stuck at a nonoptimal point as the slack variables prematurely go to 0. This phenomenon is called *jamming* in [7], where the authors also show that it can be avoided if the Jacobian of the active constraints in nonsingular. The nonsingularity ensures that the reduced KKT matrix remains nonsingular, even as the entries in the lower diagonal approach zero.

A penalty approach can have a similar effect by regularizing the reduced KKT matrix, as in (16), even if the Jacobian is nonsingular. It is proved in [2] that the step direction for the relaxation variable will always be strictly positive as the slack variable approaches 0. Additionally, we update the penalty parameter whenever the relaxed slacks approach their upper bounds. This ensures that the diagonal terms will not prematurely go to 0. In fact, while the algorithm described in Section 2 jams at an infeasible point on the so-called Wächter-Biegler problem of [23], the penalty approach solves it easily to optimality.

3.4.2. *Relaxation and Infeasibility Detection.* Another benefit of a penalty approach is due to the relaxation of the slack variables. The primal-dual pair (9)-(10) always has a feasible solution pair, and therefore, optimal primal-dual solutions to these problems always exist. Therefore, a provably convergent NLP algorithm is guaranteed to find a local optimal solution to (9). If this solution has the property that for at least one $i = 1, \ldots, m$, $\xi_i \to a$ for some scalar $a > 0$ as $c_i \to \infty$, then the original problem is primal infeasible. Similarly, if $\psi_i \to a$ as $b_i \to \infty$ for any $i = 1, \ldots, m$ for some strictly positive scalar a, then the original problem is dual infeasible.

It is impractical to allow a penalty parameter to become infinite. Therefore, we have implemented both a primal and a dual feasibility restoration phase. The primal feasibility restoration phase is similar to Phase I of SNOPT [12], in that the problem

$$\begin{array}{ll} \text{minimize} & c^T\xi \\ \text{subject to} & h(x) + w = 0 \\ & -\xi \leq w \leq b \\ & \xi \geq 0 \end{array} \tag{17}$$

is solved in order to minimize feasibility. It differs from SNOPT's version in that the primal slack variables are still bounded above by the dual penalty parameter. Since these parameters get updated whenever necessary, we can always find a feasible solution to (17). If the optimal objective function value is nonzero (numerically, greater than the infeasibility tolerance), a certificate of primal infeasibility is issued.

Similarly, the dual feasibility restoration phase solves the dual feasibility problem:

$$\begin{array}{ll} \text{maximize} & f(x) - \nabla f(x)^T x - b^T\psi \\ \text{subject to} & \nabla f(x) - A(x)^T y = 0 \\ & -\psi \leq y \leq c - \psi \\ & \psi \geq 0. \end{array} \tag{18}$$

If the penalty objective is nonzero at the optimal solution to (18), then the problem (2) is determined to be dual infeasible.

3.4.3. *Warmstarting.* *Warmstarting* is the use of information obtained during the solution of a problem to solve the subsequent, closely-related problems. Interior-point methods have long been perceived as being deficient in terms of warmstarting. In [4], the authors highlight three sources of this deficiency:

(1) The diagonal entries of the matrix in (4) are either 0 or ∞ (numerically, a very large number) at the optimal solution of the original problem. A perturbation to the problem can affect the Hessian, Jacobian, and right-hand side of (4), but the entries on the lower right diagonal of (4) will remain the same as before. Any change to the row of an inactive constraint, which has an infinite diagonal entry, can be absorbed by a very small change to its Lagrange multiplier. If the inactive constraint is to be active for the new problem, however, the Lagrange multiplier may need to be moved significantly away from 0.
(2) If an active constraint is perturbed, then the step direction for its slack variable, which is currently at 0, may be negative. The barrier then forces the steplength to be cut until it is very small. The algorithm stalls at a nonoptimal point, thus exhibiting the same behavior as jamming.
(3) If an active constraint is to become inactive, its Lagrange multiplier may move to 0 even though the slack variable cannot make progress. Similarly, if an inactive constraint is to become active, its slack variable may move to 0 even though its Lagrange multiplier cannot make progress. In both of these cases, degeneracy arises, and the Cholesky factorization and the subsequent backsolve start producing numerical errors, and iterative refinement measures are needed. However, such measures provide step directions that move the iterates away from the warmstart solution, effectively producing a coldstart.

A small numerical example illustrating these points is provided in [4]. The remedy proposed in that paper is the use of an exact primal-dual penalty approach in the case of linear programming, and we are expanding the same argument to the case of NLP in this paper.

3.4.4. *Finding nonKKT Solutions.* Unlike linear programming, the duality gap at the optimal solution of an NLP does not have to be 0. As in the case of problem 13 from the Hock and Schittkowski test set [14], the unique primal optimal solution may not have a corresponding dual solution. Thus, the optimum is not a KKT point, and no other KKT points exist.

As shown in [2], the ℓ_1 penalty approach produces a KKT point close to the unique optimum. In fact, the KKT point approaches the optimum as the penalty parameters go to ∞. Therefore, the exact penalty method behaves as an inexact method in the absence of KKT solutions. In [2], this behavior is proved for the general NLP case.

3.4.5. *Unboundedness of Optimal Sets of Primal and Dual Variables.* Optimality condition (15) ensures that the Lagrange multipliers are bounded above by the penalty parameter. For problems such as mathematical programs with equilibrium constraints (MPECs), this feature is particularly attractive as their optimal Lagrange multiplier sets are unbounded. Similarly, an optimal set of primal variables

can also be unbounded, for example

$$\begin{array}{lrcl} \text{minimize} & x_1^2 - x_2^2 & & \\ \text{subject to} & x_1 - x_2 & \geq & 0 \\ & x_1, x_2 & \geq & 0. \end{array}$$

For a primal-dual interior-point method, such as LOQO, such unboundedness will cause numerical difficulties as the algorithm will try to find the analytic center of the face of optimal solutions, and this center will be at infinity. Therefore, employing a penalty approach with sufficiently large penalty parameters to admit at least one set of optimal primal-dual solutions will allow the algorithm to find the center at a finite solution.

3.5. **Switching in and out of a Penalty Approach.** As numerical studies such as [6] and [18] show, LOQO (and other interior-point NLP solvers) are fairly robust and efficient just as they are. Bringing relaxation variables and penalty parameters into the model may require additional iterations to be performed, especially if the solver will need to first increase the penalty parameters to admit an optimal solution. In fact, in the next section, we will show that using a primal-dual penalty approach to solve our problems from start to finish significantly increases the total number of iterations and the total runtime on a standard set of problems.

Therefore, given the benefits outlined above, we have decided to use the penalty approach only as necessary. That is, the problem will be solved using the algorithm described in Section 2, without the penalty method, unless one of the following occurs:

(1) A warmstart is performed: In order to avoid getting stuck at the previous problem's optimal solution, the regularization and the relaxation properties of a penalty approach are needed. Therefore, when performing a warmstart, the primal-dual penalty method is used from the start of the algorithm.
(2) Numerical problems arise: If any of the diagonal entries in the lower right corner of the matrix in (4) go to 0 prematurely, as in the Wächter-Biegler problem, or if the reduced KKT system (4) becomes inconsistent and requires iterative refinement, the regularization aspect of a penalty method would help to remedy the numerical problems that arise in the solution of the reduced KKT system. The numerical problems may also arise due to functions that become nondifferentiable at certain points along the algorithm's trajectory toward a solution. Restarting the algorithm, this time in penalty mode, would change this trajectory, and unless the point of nondifferentiability is at the optimal solution, it can be avoided.
(3) There are too many step cuts: The length of a step taken along the computed step direction may have to be substantially cut before the step can sufficiently reduce the barrier objective or the primal infeasibility. In this case, the algorithm will not be able to make good progress from the current point. If the steplength is halved at least 10 times, we will switch over to penalty mode.
(4) Sufficient progress toward optimality cannot be made: The Armijo condition employed in the steplength control determines measures of sufficient progress, based on the computed step direction and the gradient of the Lagrangian at the current solution. If the norm of either of these vectors is too small or if the Armijo condition would admit an increase in either

	Problems Solved	Total Iterations	Total Runtime
Default	986	50520	3029.73
Penalty	977	60820	7096.78

TABLE 1. Comparison of the performance of the default and the penalty codes on the CUTEr test suite. The total iterations and runtime reported are only for commonly solved problems.

the barrier objective or the primal infeasibility, then the Armijo criteria are no longer reliable, and we cannot be expected to make sufficient progress toward optimality. In this case, we will switch over to penalty mode.

If any of these problems arise in penalty mode, and primal or dual feasibility have not been attained, LOQO will switch into feasibility restoration mode on either the primal side or the dual side, as necessary. As noted before, feasible solutions always exist for the penalty problems, and the algorithm proceeds in penalty mode with the original objective functions after a feasible point has been attained. If the problems above are encountered again, the algorithm enters primal or dual feasibility restoration mode, this time solving the feasibility problem to optimality. If the relaxation on either the primal or the dual side cannot be driven to 0, we have a certificate of primal or dual infeasibility. Otherwise, the algorithm will leave penalty mode and proceed from a primal-dual feasible pair of solutions.

4. Numerical Results

The interior-point method outlined in Section 2 is implemented in LOQO, Version 6.06. We will refer to this code as the "default" in our numerical results. We also implemented the primal-dual penalty approach in LOQO. We have a version that uses the penalty method from start to finish, without the switching conditions outlined in the previous section, and we will call this version the "penalty" code. Most of our numerical analysis will instead focus on the "hybrid" code, which implements the primal-dual penalty, but only invokes it in accordance with the switching conditions. We have performed several tests on the CUTEr test set [9], the Hock and Schittkowski test set [14], and mixed-integer nonlinear programming (MINLP) problems from the MINLPLib [8] test set, among others.

4.1. **CUTEr Set.** We have tested the primal-dual penalty approach, both the "penalty" and the "hybrid" codes, on 1073 problems from the CUTEr test suite [9] and compare our results to the nonpenalty version of LOQO (Version 6.06) as described in Section 2. Summary results are provided in this section. Full testing results are not included with this paper due to space limitations but are available online at [3].

Tables 1 and 2 present an overview performance for each code on the full set of problems. The first column is the number of problems solved by each code, and the second and third columns are the total number of iterations and the total runtime (in CPU seconds) over the commonly solved problems. Note that the total iterations and total runtimes for the default code differ between the two tables, and this is due to the fact that the sets of commonly solved problems are different for the two comparisons.

	Problems Solved	Total Iterations	Total Runtime
Default	986	57116	3927.50
Hybrid	1002	39777	4881.23

TABLE 2. Comparison of the performance of the default and the hybrid codes on the CUTEr test suite. The total iterations and runtime reported are only for commonly solved problems.

In Table 1, a comparison between the default and the penalty codes are provided. The penalty code, whose performance can be greatly affected by the choice of the penalty parameter, does suffer from the initializations described in the previous section. Since the initial estimates of the slack variables may be far from the optimal solution, setting the penalty parameters too small may need too many iterations to update them to the right values. On the other hand, setting them too large may seek to minimize the infeasibility first. For an infeasible interior-point NLP solver such as LOQO, focusing first on the attainment of feasibility may be detrimental to the algorithm's performance. We have verified that many of the small problems that the penalty code fails on can be solved by a careful calibration of the initial values of the slack variables. This shows promise for the hybrid code, which will start after several iterations of the algorithm or after a warmstart, with good estimates for the slack variable values and, therefore, for the penalty parameters. Additionally, using the penalty code from start to finish on the commonly solved problems takes almost 20% more iterations and twice as much time. The difference in the total runtime is mainly due to two problems, *porous1* and *msqrtb*, where the default code finds the solution in fewer iterations and each additional iteration is very expensive. These two problems combined contribute to about 2800 seconds of the difference between the total runtimes.

As shown in Table 2, the hybrid code retains the theoretical robustness of the penalty code and improves on its computational robustness by using better estimates of the penalty parameters. By switching over only as necessary, it is far more efficient than the penalty code. In fact, by providing regularization when necessary, it also improves on the efficiency of the default code. Therefore, the rest of the numerical results in this section will concentrate on the hybrid code. It is also important to make three more notes here: (1) the hybrid method switched into penalty mode on 253 problems, (2) the difference between the default and the hybrid performance in iterations is attributable in most part to five problems, *s380*, *s380a*, *palmer7a*, *rk23*, and *haifam*, which together account for a difference of about 14,000 iterations, and (3) the difference between the default and the hybrid performance in runtime is due entirely to one problem *reading7*, which accounts for a difference of approximately 1000 CPU seconds.

We now examine the problems documented in Table 3 that were solved by the hybrid code, but not by the default code. There are 28 such problems, and of these, one of the problems, *s220*, switched to the penalty after predicting that sufficient progress cannot be made using the step directions calculated without the penalty. The rest of the problems switch over after making too many steplength cuts in the filter. These cuts may be due to numerical issues, unboundedness of the optimal set, or any other reason that may affect the quality of the step direction.

artif	brainpc0	brainpc5	catenary
cresc100	cresc132	discs	eigmaxc
eqc	gausselm	hager4	kiwcresc
palmer5a	palmer7e	polak5	polak6
reading1	robotarm	s220	s355
s355a	sineali	snake	spiral
trainf	twirimd1	vanderm3	zamb2

TABLE 3. Problems solved by the hybrid code and not by the default.

Local infeasibility: *powellsq*
Numerical problems in penalty mode: *yao*
Less accuracy: *deconvc*, *dixchlnv*, *hvycrash*, *spanhyd*, *steenbre*, *steenbrg*, *vanderm1*
Iteration Limit: *lukvli10*, *methanol*, *reading4*

TABLE 4. Problems solved by the default code and not by the hybrid code.

There are also 12 problems that are solved to optimality by the default code but not by the hybrid code. These problems are categorized in Table 4. Some of this difference can be attributed to the different trajectories taken by the two approaches. For example, in solving problem *powellsq*, the hybrid code goes to a neighborhood in which feasibility cannot be attained. The six problems that end with less accuracy do so because they have reached the optimal objective function value on the primal side and are primal feasible, but the dual objective function and/or dual infeasibility have not yet reached their optimal values. In this case, the purely primal nature of the steplength control mechanism is the cause of the problem, as we check for predicted decrease in either the barrier objective or the primal infeasibility as the Armijo conditions. Since the primal solution does not need to move to satisfy the stopping criteria, the predicted decrease is too small. We conclude by deciding that the current solution cannot be improved. This is the correct conclusion for the primal solution, but the dual solution can still improve. One remedy for this situation is to consider including dual measures in the steplength control mechanism, as in [21].

Next, we examine the problems that remain unsolved by both the default and the hybrid codes. A distinct advantage of the hybrid code is that having tried the penalty code and, if necessary, the feasibility restoration approaches, we can make a determination as to whether the problem is primal or dual infeasible and whether we have made all the possible progress toward an optimal solution. To that end, the hybrid code has identified the problems in Table 5 as being primal infeasible.

These three problems are in addition to *powellsq* as discussed in Table 4. Since the algorithm is trying to attain a local optimum, the problems are classified as being locally infeasible. We will discuss results of dual infeasibility detection later in this section.

argauss
himmelbd
lewispol

TABLE 5. Primal infeasible problems identified by the hybrid code.

Problem Name	Complementarity	Primal Feasibility	Dual Feasibility
allinitc	1.00e-5	1.00E-008	5.00E-008
brainpc6	1.00e-8	8.00E-004	4.00E-008
brainpc9	1.00e-7	1.00E-005	2.00E-005
hs013	1.00e-3	8.00E-008	9.00E-007
s221	1.00e-4	5.00E-008	6.00E-007
steenbrb	1.00e-7	4.00E-017	9.00E-007
steenbrf	1.00e-3	4.00E-013	1.00E-003

TABLE 6. Problems not solved by the default code and where sufficient progress cannot be made, as identified by the hybrid code.

	Problems Solved	Total Iterations
Hybrid LOQO	1016	55819
IPOPT	1013	39061

TABLE 7. Overview of the performance of hybrid LOQO and IPOPT on the CUTEr test suite. Total iterations are reported only on the commonly solved problems.

The hybrid code stops the problems in Table 6 before reaching the desired levels of accuracy because further progress cannot be made, even after trying the penalty approach. In Table 6, we provide the scaled measures of optimality at the point the hybrid code stops working on the problem and signals the user that the "current point cannot be improved." Two of these problems, *hs013* and *s221*, have a unique optimum that is not a KKT point, and as discussed in the previous section, such a point can be approached as the penalty parameters increase to infinity. The hybrid code determines that it has increased the penalty too many times and stops. It is also important to note that the optimality measures reported in Table 6 are the KKT conditions of the original problem and not the relaxed penalty problem.

There are still 50 problems that remain unsolved by both the default and the hybrid codes. Some of these problems have points of nondifferentiability that cannot be avoided and some have scaling issues that cause numerical problems. While we cannot overcome the nondifferentiability, we are looking into rescaling approaches for future versions of LOQO.

Next, we compare the hybrid code to the interior-point solver IPOPT [24] (Version 3.0.0). It should be noted that the basic approach of each algorithm is as described in Section 2. They differ in details such as Cholesky factorization routines, steplength control, handling of equality constraints, and regularization. The two codes also differ in their stopping criteria. While IPOPT relaxes variable bounds and

uses the relaxed measures to stop, LOQO always computes the optimality measures of the original problem, even in penalty mode. This results in accuracy differences between the two codes. For example, on problem *hs013*, the solution reported by IPOPT as being optimal with less than 10^{-9} in all three unscaled measures of optimality is determined by LOQO to be dual infeasible and with only 10^{-4} as the complementarity.

We provide a performance comparison in Table 7 showing that the hybrid approach is competitive in terms of robustness. After accounting for 8 problems (*chain*, *huestis*, *qpnboei1*, *reading1*, *sawpath*, *trainf*, and *zamb2*) which together make up about 14000 of the difference in the numbers of iterations, we can see that the hybrid code is also competitive in terms of efficiency on the commonly solved problems. In order to deflect some of the accuracy level differences, we have included the problems attaining lower levels of accuracy than LOQO's default settings, as listed in Tables 4 and 6. Doing so ensures that problems such as *hs013* where LOQO stops within 3 digits of the true optimum and IPOPT stops within 2 digits are counted as solved by both codes. We have also included problems listed as "Solved to Acceptable Level" by IPOPT as solved in the above table. We have verified that for all of those problems both methods are approaching the same optimum.

4.2. **Dual Infeasibility Detection.** As discussed in the previous section, the primal-dual penalty approach can be used to detect dual infeasible problems. Given that a primal feasible solution exists, dual infeasibility indicates that a problem is primal unbounded. There are three such problems in the CUTEr test suite (*s255*, *s340*, and *s340a*). However, all three of these problems have free variables, and there are no corresponding slack variables to be relaxed to test our dual infeasibility detection approach. One possible remedy is to split a free variable, x, into

$$x - g + t = 0, \quad g \geq 0, \quad t \geq 0,$$

and relax and bound the primal slacks g and t and their corresponding dual slacks. Free variable splitting of this form has been removed from LOQO in its latest version due to efficiency issues, but it would have an advantage in terms of dual infeasibility detection.

Since a suitable test problem was not available in the CUTEr set, we formulated one of our own:

$$\begin{array}{lrcl} \text{minimize} & -x_1^4 - x_2^4 & & \\ \text{subject to} & x_2^2 - x_1^2 & \geq & 0 \\ & x_1, x_2 & \geq & 0. \end{array}$$

In fact, LOQO enters dual feasibility restoration mode and finds that the optimal solution has a nonzero penalty. Therefore, the problem is determined to be dual infeasible.

In order to enter dual feasibility restoration mode, we monitored the increase in the dual penalty parameters due to the increase in the primal slacks. In this case, as x_1 and x_2 tend to infinity, so do the slack variables on their lower bounds. Therefore, the dual penalty parameters, which act as their upper bounds, are increased too many times.

In comparison, the primal feasibility restoration mode is also entered when the penalty approach has problems with the steplength control. This is due to the fact that the steplength control mechanism, which monitors the barrier objective and the primal infeasibility, is purely primal. Therefore, any issues with obtaining

sufficient decrease may be remedied by re-initializing the penalty parameters and focusing solely on reducing primal feasibility.

If dual feasibility measures are to be considered within the steplength control scheme, we may be able to detect situations in which a dual feasibility restoration may also be used. Such a scheme is used in [21] as a filter with criteria from the KKT conditions (3).

4.3. **Warmstarts.** As discussed in the previous section, an important benefit of the primal-dual penalty approach is the ability to warmstart by using the solution of an initial problem to solve a closely related, possibly perturbed, problem. Gathering a test suite is not challenging in the case of linear programming, as discussed in [4], because we can simply perturb the data for any such problem. In nonlinear programming, however, a meaningful test set is hard to obtain, and we report here on several examples from literature. We will also report on reoptimization of the same problem from a boundary point close to the optimum, which has issues similar to warmstarting.

4.4. **Goal Programming.** Goal programming is a technique for solving multicriteria optimization problems. There are several objective functions, representing several goals, that are prioritized in order of importance. For each goal, a new problem is solved with the objective function corresponding to the goal and additional constraints implying that the level of attainment of the more important goals cannot be weakened.

In this paper, we consider a nonlinear goal programming problem from [15]. The problem is as follows:

- Priority 1 goal: $3x_1^2 + x_2^2 - 12x_1 - 8x_2 \leq -25$
- Priority 2 goal: $(1 - x_1)^2 + (2 - x_2)^2 \leq 1$.

We would like to get each of these goals satisfied as much as possible, so we start by adding nonnegative slack variables, d^- and d^+, into each goal. The first subproblem to solve is

$$\begin{array}{lrcl} \text{minimize} & d_1^+ & & \\ \text{subject to} & 3x_1^2 + x_2^2 - 12x_1 - 8x_2 + d_1^- - d_1^+ & = & -25 \\ & (1 - x_1)^2 + (2 - x_2)^2 + d_2^- - d_2^+ & = & 1 \\ & x_1, x_2, d_1^-, d_1^+, d_2^-, d_2^+ & \geq & 0. \end{array}$$

At the optimal solution to this problem, we have that d_1^+ is equal to 0. Therefore, we have been able to satisfy our Priority 1 goal. The second subproblem must try to achieve the Priority 2 goal without weakening our attainment of the first goal. It can be expressed as:

$$\begin{array}{lrcl} \text{minimize} & d_2^+ & & \\ \text{subject to} & 3x_1^2 + x_2^2 - 12x_1 - 8x_2 + d_1^- & = & -25 \\ & (1 - x_1)^2 + (2 - x_2)^2 + d_2^- - d_2^+ & = & 1 \\ & x_1, x_2, d_1^-, d_2^-, d_2^+ & \geq & 0. \end{array}$$

This problem has one less variable and has a different objective function. The primal infeasibility does not change, but the dual infeasibility gets affected by the change in the objective function. Therefore, it is important that a penalty method to relax this problem should have a dual component which relaxes the dual slacks. In fact, without the penalty approach, LOQO solves the second problem in 20 iterations.

With the primal-dual penalty turned on to perform a warmstart from the first problem's solution, LOQO solves the second problem in 10 iterations.

4.5. **Hock and Schittkowski problems.** As discussed above, it is particularly challenging to meaningfully perturb an NLP. Therefore, to test our approach, we took a different point of view, and perturbed the optimal solution of a problem and reoptimized it from this nearby point. In order for the test to appropriately challenge the solver, it is not sufficient to just perturb the primal and the dual variables. The warmstart solution should have complementarity preserved, with changes to the active set whenever appropriate. If the nonzero term in the complementary pair is significantly larger than 0, moving it to 0 can move a dual variable or a slack too far from the optimal solution, effectively creating a coldstart that can be worse than the initial solution to the problem. Thus, we have perturbed only those complementary pairs where the nonzero element is no more than 10^{-2}. In this case, we randomly picked the member of the complementary pair to be zero, and then assigned a perturbed value to the nonzero member. Doing so preserved complementarity, and such a solution can be the optimum for a closely related problem. The only issue in this case is that the number of active set changes is not significant. This problem would be remedied by the availability of a good test set for warmstarting.

We tested 103 problems from the Hock and Schittkowski [14] test suite. The variables (decision and slack) were perturbed randomly. The random numbers were obtained using the random number generator Mersenne Twister [17], and we generated a list of numbers uniformly distributed between -1 and 1. For each perturbation, a scalar, ϵ, was read from this list to determine the magnitude of the perturbation. In order to match the scale of the numbers in the problem, the following scheme was used when perturbing variable x_j:

$$x_j^{(0)} = \begin{cases} \delta\epsilon, & \text{if } x_j^* = 0 \\ x_j^*(1+\delta\epsilon), & \text{otherwise.} \end{cases}$$

where the initial, perturbed solution is denoted by $x^{(0)}$ and the optimal solution is denoted by x^*. The dual variables, y, and the primal slacks, w, were perturbed using the same scheme, while ensuring that they did not become negative. The parameter δ was varied to observe the effects of different levels of perturbation. For each complementary pair, (w_i, y_i), $i = 1, \ldots, m$, where the nonzero element is no more than 10^{-2}, another random number ϵ_2 was drawn. If $\epsilon_2 < 0$, we set $w_i = 0$. Otherwise, we set $y_i = 0$. For each original problem, we present the numerical results obtained by 3 different perturbations at three different levels of δ (0.0001, 0.001, 0.01). We report on the distance between the optimal solutions to the original and perturbed problems, along with the number of changes to the active set, and compare iteration counts for the warmstart and the coldstart solutions. These results are summarized in Table 8 and presented in detail online at [3].

The distance between the primal solutions is computed as

$$\frac{\|x^{(0)} - x^*\|}{1.0 + \|x^{(0)}\|},$$

and the distance between the Lagrange multipliers is computed as

$$\frac{\|y^{(0)} - y^*\|}{1.0 + \|y^{(0)}\|}.$$

δ	Reduction in Iters	Primal Dist	Dual Dist
0.0001	39%	3.70e-03	2.13e-03
0.001	33%	6.14e-03	1.72e-02
0.01	24%	1.90e-02	3.40e-02

TABLE 8. Average performance of the primal-dual penalty approach when warmstarting 103 problems from the Hock and Schittkowski test set. δ is the perturbation factor, the second column is the average reduction in iterations from the coldstart to the warmstart, and Primal Dist and Dual Dist are the scaled Euclidean distances between the warmstart and the optimal solutions for the primal and the dual variables, respectively.

When the distance between the solutions is small, generally 10^{-2} or less, the warmstart scheme works very well, as evidenced by the decrease in the number of iterations from solving the perturbed problem with a coldstart. However, when the distance is large, such as *hs055* as seen in the detailed results available from [3], the performance deteriorates. Nonetheless, such a distance represents a significant enough change in the problem that there is no reason to perform a warmstart.

4.6. **MINLPs.** We also implemented a branch-and-bound algorithm to solve mixed-integer nonlinear programming (MINLP) problems, using LOQO to solve the nonlinear relaxation at each node of the tree, and examined the performance of our approach on warmstarting after variable bound perturbations. The test suite is a group of 37 problems from MINLPLib [8], which were chosen among the problems with less than 10 discrete variables. The criteria for choosing these problems were that the coldstart and the warmstart arrived at the same optimal solution (there can be a difference for problems with nonconvex subproblems) and that there be no more than 50 subproblems. Full testing results for the 37 problems are provided at [3].

In Table 9, we provide a few representative examples of our results. The results present a comparison of the iteration counts between a coldstart, which goes back to the original initial solution, and a warmstart, which starts the node from the optimal solution of its parent. For each problem, we show the results for every node in the branch-and-bound tree.

We have made several observations during the solution of the MINLPs. First, many problems included binary variables, and for such a problem, changing variable bounds only changes the slacks for the bounds by at most 1. Therefore, letting the initial values of the relaxation variables on the bounds and δ for the corresponding penalty parameters equal 1 worked the best. Second, note that for many of the subproblems, the scaled distance between the solutions (both primal and dual) of the problem and its parent is quite large. When the distance is small, we can generally see an improvement in the number of iterations, but as is consistent with our observation above, when either of the distances is greater than 10^{-2}, a warmstart may indeed hurt the problem because the initial solution is too far away. Third, perhaps the biggest advantage of using a penalty approach is the infeasibility detection scheme which is necessary for many subproblems. Without this capability, the algorithm will run to iteration limit, wasting runtime and not giving a conclusive

Problem	Node	ColdIters	WarmIters	Δ	PrimalDist	DualDist
ex1223a	1	14	13	1	2.08E-002	9.66E-001
ex1223a	2	11	10	8	3.18E-002	1.70E+000
ex1223a	3	12	11	0	2.81E-002	6.73E-002
ex1223a	4	12	10	1	1.79E-002	1.39E-001
ex1223a	5	12	14	1	5.85E-003	5.10E-002
ex1223a	6	10	10	1	4.47E-018	8.16E-001
ex1223a	7	12	10	2	3.43E-004	1.07E-001
ex1223a	8	12	10	0	2.67E-002	2.38E-001
ex1223a	9	12	11	1	4.93E-015	2.53E-001
ex1223a	10	22	28	16	7.02E+000	5.07E-001
ex1224	1	19	13	2	2.83E-002	1.33E+002
ex1224	2	19	16	2	1.05E-001	5.93E-005
ex1224	3	18	17	1	2.67E-002	1.85E-004
ex1224	4	17	17	2	2.57E-005	1.37E+002
ex1224	5	17	21	2	3.10E-002	3.69E-004
ex1224	6	20	30	2	1.25E-001	8.02E-005
ex1224	7	35	13	1	1.05E-002	1.25E-003
ex1224	8	30	33	2	3.48E-004	9.29E+001
ex1224	9	51	36	0	1.70E-003	1.31E-004
ex1224	10	56	53	0	1.98E-004	1.95E-005
ex1224	11	47	28	3	1.01E-001	6.24E-005
ex1224	12	55	48	3	1.36E-002	1.51E-003
ex1224	13	25	18	1	3.04E-003	6.20E-005
ex1224	14	24	27	1	2.53E-002	1.32E-006
ex1224	15	31	41	2	1.27E-001	3.66E-004
nvs03	1	21	16	3	8.92E-003	9.84E-001
nvs03	2	11	15	0	8.33E-003	7.73E+000
nvs03	3	21	18	2	8.26E-003	9.98E-001
nvs03	4	(IL)	59(INF)	4	1.55E-002	2.21E+002
nvs03	5	21	21	1	3.57E-002	1.44E+002
nvs03	6	(IL)	60(IMP)	2	1.64E-002	2.37E+001
nvs03	7	23	21	2	3.42E-003	6.89E-001
nvs03	8	(IL)	56(INF)	4	9.22E-001	4.87E+000

TABLE 9. Numerical performance of LOQO on the MINLPLib test suite. "Node" indicates the number of the node on the branch-and-bound tree, "ColdIters" is the coldstart iteration count, "WarmIters" is the warmstart iteration count, Δ is the number of changes to the active set, "PrimalDist" and "DualDist" are the relative distances between the parent and the child node for the primal and the dual variables, respectively. (IL) indicates that the algorithm reached the iteration limit, (INF) indicates that the algorithm determined that the problem is infeasible, and (IMP) indicates that further improvement could not be made on a problem.

result as to the status of the subproblem. As seen in problem *nvs03* in Table 9, the penalty approach is able to detect infeasibility of the subproblems after a reasonable number of iterations.

5. Conclusion

As evidenced by the numerical results, the primal-dual penalty approach improves the robustness of an interior-point method on NLPs. In order to retain the efficiency of these methods, however, a hybrid approach that switches over to the penalty method only when needed works best.

While penalty methods have been studied before, either as a regularization or an elastic mode in the context of SQP, they have been limited to being purely primal. In this paper, we have extended the primal-dual penalty approach of [4] to NLPs,

and this approach retains the desirable theoretical properties of its primal counterpart, but improves dual infeasibility identification and warmstarting capabilities. It also preserves the symmetry of a primal-dual interior-point method.

While we have demonstrated the effectiveness of the primal-dual penalty approach for warmstarting NLPs, we would like to emphasize that more examples are needed to develop a truly meaningful test suite for warmstarting problems. As the work in this important area progresses, such a test suite will be vital to assessing the success of various proposed methods. We encourage all researchers to submit examples toward the development of this test suite.

References

[1] Mihai Anitescu. Nonlinear programs with unbounded lagrange multiplier sets. Technical Report ANL/MCS-P793-0200, Argonne National Labs.

[2] H. Y. Benson, A. Sen, D.F. Shanno, and R. J. Vanderbei. Interior point algorithms, penalty methods and equilibrium problems. *Computational Optimization and Applications*, 2005. (to appear).

[3] H.Y. Benson. Numerical testing results for the primal-dual penalty approach. http://www.pages.drexel.edu/~hvb22/penaltyweb.

[4] H.Y. Benson and D.F. Shanno. An exact primal-dual penalty method approach to warmstarting interior-point methods for linear programming. Technical report, Working Paper, September 2005.

[5] H.Y. Benson, D.F. Shanno, and R.J. Vanderbei. A comparative study of large scale nonlinear optimization algorithms. In *Proceedings of the Workshop on High Performance Algorithms and Software for Nonlinear Optimization, Erice, Italy*, 2001.

[6] H.Y. Benson, D.F. Shanno, and R.J. Vanderbei. Interior-point methods for nonconvex nonlinear programming: Filter methods and merit functions. *Computational Optimization and Applications*, 23(2):257–272, November 2002.

[7] H.Y. Benson, D.F. Shanno, and R.J. Vanderbei. Interior-point methods for nonconvex nonlinear programming: Jamming and comparative numerical testing. *Mathematical Programming A*, 99(1):35–48, 2004.

[8] M.R. Bussieck, A.S. Drud, and A Meeraus. MINLPLib - a collection of test models for mixed-integer nonlinear programming. *INFORMS Journal on Computing*, 15(1).

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

[10] 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.

[11] R. Fletcher. *Practical Methods of Optimization*. J. Wiley and Sons, Chichester, England, 1987.

[12] 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.

[13] N.I.M. Gould, D. Orban, and Ph.L. Toint. An interior-point l1-penalty method for nonlinear optimization. Technical Report RAL-TR-2003-022, Rutherford Appleton Laboratory Chilton, Oxfordshire, UK, November 2003.

[14] W. Hock and K. Schittkowski. *Test examples for nonlinear programming codes*. Lecture Notes in Economics and Mathematical Systems 187. Springer Verlag, Heidelberg, 1981.

[15] G. Kuriger and A.R. Ravindran. Intelligent search methods for nonlinear goal programming problems. *INFOR: Information Systems and Operational Research*, 43(2):79–92, 2005.

[16] S. Leyffer, G. Lopez-Calva, and J. Nocedal. Interior methods for mathematical programs with complementarity constraints. Technical Report OTC 2004-10, Northwestern University, Evanston, IL, December 2004.

[17] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, 1998.

[18] J. Nocedal, J.L. Morales, R. Waltz, G. Liu, and J.P. Goux. Assessing the potential of interior-point methods for nonlinear optimization. Technical Report Report OTC-2001-6, Optimization Technology Center, 2001.
[19] J. Nocedal and R. A. Waltz. Knitro 2.0 user's manual. Technical Report OTC 02-2002, Optimization Technology Center, Northwestern University, January 2002.
[20] 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.
[21] M. Ulbrich, S. Ulbrich, and L. Vicente. A globally convergent primal-dual interior point filter method for nonconvex nonlinear programming. *Mathematical Programming*, 100:379–410, 2004.
[22] R.J. Vanderbei and D.F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13:231–252, 1999.
[23] A. Wächter and L. Biegler. Failure of global convergence for a class of interior point methods for nonlinear programming. *Mathematical Programming*, 88(3):565–587, 2000.
[24] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Technical Report RC 23149, IBM T. J. Watson Research Center, Yorktown, USA, March 2004.

Hande Y. Benson, Drexel University, Philadelphia, PA

David F. Shanno, RUTCOR - Rutgers University, New Brunswick, NJ