INTERIOR-POINT METHODS FOR NONCONVEX NONLINEAR PROGRAMMING: PRIMAL-DUAL METHODS AND CUBIC REGULARIZATION

HANDE Y. BENSON AND DAVID F. SHANNO

ABSTRACT. In this paper, we present a primal-dual interior-point method for solving nonlinear programming problems. It employs a Levenberg-Marquardt (LM) perturbation to the Karush-Kuhn-Tucker (KKT) matrix to handle indefinite Hessians and a line search to obtain sufficient descent at each iteration. We show that the LM perturbation is equivalent to replacing the Newton step by a cubic regularization step with an appropriately chosen regularization parameter. This equivalence allows us to use the favorable theoretical results of [15], [19], [5], and [6], but its application at every iteration of the algorithm, as proposed by these papers, is computationally expensive. We propose a hybrid method: use a Newton direction with a line search on iterations with positive definite Hessians and a cubic step, found using a sufficiently large LM perturbation to guarantee a steplength of 1 otherwise. Numerical results are provided on a large library of problems to illustrate the robustness and efficiency of the proposed approach on both unconstrained and constrained problems. This paper extends our previous work [2] from purely primal barrier methods to the primal-dual interior-point method framework.

1. INTRODUCTION

Many algorithms for nonlinear programming (NLP) employ a quadratic model of the problem. Newton's Method and Quasi-Newton Methods use a second-order Taylor series expansion of the objective function (in the unconstrained case) or a merit function (in the constrained case), with either an explicit or approximated Hessian matrix. Successful implementations include interior-point codes LOQO [22], IPOPT [24], and KNITRO [4], as well as active set codes FILTERSQP [11] and SNOPT [13]. In theory and practice, these codes are robust on problems that obey certain regularity conditions and exhibit super-linear rates of local convergence on those problems. A quadratic model of the problem requires the solution of a linear system at each iteration, and the solution of this system typically dominates the solution time.

Recently, papers by Nesterov and Polyak and Cartis, et.al. have proposed using a cubic regularization when computing the step direction within a nonlinear programming algorithm. In fact, the consideration of using a cubic regularization to the quadratic model when computing a step direction appears in literature as early as three decades ago, in a paper by Griewank. In [15], he proposes a cubic regularization to develop an algorithm based on Newton's method that is affine-invariant and convergent to second-order critical points.

Date: May 8, 2012.

Key words and phrases. interior-point methods, nonlinear programming, cubic regularization, Newton's method.

In [19], Nesterov and Polyak re-introduce cubic regularization of Newton's method for unconstrained NLPs and provide global complexity results for certain classes of problems. Their main motivation is that the cubic model is a global upper estimate for the function to be minimized and, therefore, enforces global performance guarantees of the resulting method. Local convergence results are also provided. This proposed method is further accelerated for convex unconstrained NLPs by Nesterov in [18] using a multi-step approach. In [17], Nesterov also introduces an appropriate cubic regularization scheme using support functions and variational inequalities for convex constrained NLPs. While the authors give implementation pointers in all three papers, there are no numerical results provided.

Further work on cubic regularization was documented by Cartis, Gould, and Toint in [5] and [6], where the authors proposed the Adaptive Regularisation algorithm using Cubics (ARC). This algorithm uses an approximate Hessian and finds an approximate minimizer of the cubic model, which reduces the computational work while retaining the local and global convergence properties of [15] and [19] and the worst-case complexity results of [19]. The complexity results are further improved on by the authors in [7] for convex unconstrained NLPs, corresponding to similar work in [18], and they also extend their method to convex constraints in [8]. We should note that there is a distinct trust-region method focus to this work, since the regularization parameter is related by the authors to the inverse of the trust-region parameter and controlled accordingly throughout the algorithm. Numerical results are provided by the authors in [5] on small unconstrained NLPs from the CUTEr [9] test set and by Gould, Porcelli, and Toint in [14] on large nonlinear-least squares problems. Both papers show that ARC gives an improvement in iteration counts. However, the CPU time results provided in [14] for large problems show that the traditional trust-region approach based on the quadratic model is more efficient.

In all of the above mentioned papers, the regularization parameter for the cubic term is initalized at the beginning of the algorithm and then updated at each iteration using a scheme based on sufficient descent. This scheme can have a significant impact on the number of iterations performed by the algorithm. In addition, using the cubic regularization at each iteration of the algorithm means that a nonlinear system will need to be solved at each iteration. As a result, even if the overall iteration count is reduced due to obtaining better step directions, each iteration with a positive definite Hessian will require more computational effort than one iteration with a quadratic model, which requires the solution of a single linear system.

In [2], we proposed a hybrid approach that uses cubic regularization only during those iterations in which the Hessian is indefinite. We showed that computing the cubic step is equivalent to solving the linear system for a certain value of the Levenberg-Marquardt perturbation parameter, allowing us to naturally set the cubic regularization parameter in a problem dependent way for each iteration where it is needed. This selective use of cubic regularization allows us to use some of its theoretical properties without adversely affecting the convergence properties of our solution approach and to do so without costing significant extra time for the overall solution method. In addition, in [2], we began to extend this method to line-search interior-point methods for constrained NLPs. This work on constrained NLPs was preliminary and used a primal barrier method to solve the problems. Numerical results on both unconstrained and constrained NLPs were provided, and

they showed that our approach resulted in fewer iterations than and comparable solution times to a Newton's Method with line-search at each iteration.

In this paper, we present our work on a similar hybrid approach within the framework of *primal-dual* interior-point methods for general NLPs. Primal-dual algorithms represent the state-of-the-art in interior-point methods, and, as shown in [2], they can be significantly more efficient than the primal barrier method we considered in that paper. However, the adaptation of cubic regularization to the primal-dual framework is a challenge, since the regularization is obtained from the norm of the step direction and the role that the dual step direction will play in the regularization needs to be determined. We will show that the cubic regularization is equivalent to using a Levenberg-Marquardt perturbation with an appropriately chosen value of the parameter in the primal-dual setting, too, with the regularization applied only to the Hessian matrix of the linear system. Numerical tests again indicate that our hybrid approach results in fewer iterations without significantly increasing the solution time.

The outline of the paper is as follows. In the next section, we present our primal-dual interior-point method that uses cubic regularization during iterations with indefinite Hessians. We will refer to this method as the *hybrid approach* since it will use Newton's Method and a line-search in the remaining iterations. We have chosen to implement the hybrid approach within LOQO [23], and, in Section 3, we provide implementation details and the results of our extensive numerical testing on the CUTEr [9] suite of problems. We have also included results on unconstrained problems for completeness.

2. The Algorithm

We begin with a description of the primal-dual interior-point method that will be the foundation of our hybrid approach and is implemented in LOQO. The basic problem we consider is

$$\begin{array}{lll} \text{minimize} & f(x) & \\ \text{subject to} & g(x) & \geq \quad 0 \end{array} \tag{1}$$

where $x \in \mathbb{R}^n$ are the decision variables, $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^m$. We assume f and g to be twice Lipschitz continuous.

While the form of (1) only uses free variables and inequality constraints, our implementation allows for a wider range of problems. For now, we will focus on (1) for pedagogical purposes, and we will point out extensions of our approach to handle variable bounds, ranges on inequalities, and equality constraints later in this section.

We start by adding slacks, $w \in \mathbb{R}^m$, to the inequality constraints in (1), which becomes

$$\begin{array}{lrcl} \text{minimize} & f(x) & & \\ \text{subject to} & g(x) - w & = & 0 \\ & w & \geq & 0. \end{array} \tag{2}$$

The inequality constraints are then eliminated by incorporating them in a logarithmic barrier term in the objective function, transforming (2) into

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

where $\mu \geq 0$ is the barrier parameter. Denoting the Lagrange multipliers for the equality constraints by y, the first order conditions for (3) are

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

where W is the diagonal matrix with $W_{ii} = w_i$, e is the vector of all ones, and $A(x)$ is the transpose of the Jacobian matrix of the vector $g(x)$. The primal-dual system is obtained from (4) by multiplying the second equation by W, giving the system

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

where Y is the diagonal matrix with $Y_{ii} = y_i$.

Starting at an initial solution $(x^{(0)}, w^{(0)}, y^{(0)})$, Newton's method is employed to iterate to a triple (x, w, y) satisfying (5). Letting

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

the Newton system for (5) is

$$\begin{bmatrix} H(x,y) & 0 & -A(x)^T \\ 0 & Y & W \\ A(x) & -I & 0 \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta w \\ \Delta y \end{pmatrix} = \begin{pmatrix} -\nabla f(x) + A(x)^T y \\ \mu e - WYe \\ -g(x) + w \end{pmatrix} \tag{6}$$

and yields a step direction $(\Delta x, \Delta w, \Delta y)$ at each iteration. This system is not symmetric, but can be symmetrized by negating the first equation and multiplying the second by $-W^{-1}$, and further reduced by eliminating Δw without producing any additional fill-in in the off-diagonal entries. Doing so gives the so-called *reduced Karush-Kuhn-Tucker* (KKT) system:

$$\begin{bmatrix} -H(x,y) & A(x)^T \\ A(x) & WY^{-1} \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} \nabla f(x) - A(x)^T y \\ g(x) - \mu Y^{-1} e \end{pmatrix}. \tag{7}$$

with

$$\Delta w = WY^{-1}(\mu W^{-1} e - y - \Delta y).$$

When $H(x, y)$ is positive definite, the matrix in (7) is quasidefinite and strongly factorizable. This resulting system is solved using modified Cholesky factorization and a backsolve step. Since such a procedure benefits from sparsity, a symbolic factorization is performed first and an appropriate permutation to improve the sparsity structure of the Cholesky factor of the reduced KKT matrix is found.

Having computed step directions, Δx, Δw, and Δy, we proceed to a new point by

$$\begin{aligned} x^{(k+1)} &= x^{(k)} + \alpha^{(k)} \Delta x^{(k)}, \\ w^{(k+1)} &= w^{(k)} + \alpha^{(k)} \Delta w^{(k)}, \\ y^{(k+1)} &= y^{(k)} + \alpha^{(k)} \Delta y^{(k)}, \end{aligned} \tag{8}$$

where $\alpha^{(k)}$ is chosen to ensure that $w^{(k+1)} > 0$, $y^{(k+1)} > 0$, and either the barrier function or the primal infeasibility is reduced. A sufficient reduction is determined by an Armijo rule. Details of this "filter" linesearch are discussed in [3].

2.1. **Problems in General Form.** As mentioned above, our implementation accommodates problems with features other than (1). While the basic outline of the above algorithm remains the same, it is important to note how variable bounds, range constraints, and equality constraints are included in the framework and how they affect the structure of (7). In fact, let us define the general reduced KKT system

$$\begin{bmatrix} -(H(x,y)+E) & A(x)^T \\ A(x) & F \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} \sigma \\ \rho \end{pmatrix}, \tag{9}$$

where $E \in \mathbb{R}^{n\times n}$ and $F \in \mathbb{R}^{m\times m}$ are diagonal matrices, $\sigma \in \mathbb{R}^n$, and $\rho \in \mathbb{R}^m$ to aid with the below discussion.

If we have that $l_j \leq x_j \leq g_j$, the bounds are included in the problem by introducing slack variables to transform them into equality constraints:

$$\begin{aligned} x_j - g_j &= 0 \\ x_j + t_j &= 0 \\ g_j, t_j &\geq 0, \end{aligned}$$

and the slack variables are incorporated into the barrier objective function. While these bounds would then appear as constraints in the Newton System (6), they are eliminated to obtain (9) with

$$\begin{aligned} E_{jj} &= \frac{z_j}{g_j} + \frac{s_j}{t_j} \\ \sigma_j &= \nabla_j f(x) - A(x)_{\cdot j}^T y - \frac{s_j}{t_j}\left(u_j - x_j - \frac{\mu}{s_j}\right) - \frac{z_j}{g_j}\left(l_j - x_j + \frac{\mu}{z_j}\right), \end{aligned}$$

where $A(x)_{\cdot j}$ is the j-th column of $A(x)$, z_j and s_j are the Lagrange multipliers of the lower and upper bounds, respectively. Note that if $l_j = -\infty$ and/or $u_j = \infty$, the term(s) corresponding to the infinite bound(s) vanishes.

A range constraint is defined as $0 \leq g_i(x) \leq r_i$ and converted into the following form:

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

where w_i and p_i are the slack variables incorporated into the barrier objective function. Treating range constraints in this manner, rather than splitting them into two inequalities, ensures that we do not introduce linear dependencies into the Jacobian and allows us eliminate the second equality easily. Thus, the corresponding

(9) components are

$$\begin{aligned} F_{ii} &= \left(\frac{v_i}{w_i} + \frac{q_i}{p_i}\right)^{-1} \\ \rho_i &= w_i - g_i(x) - F_i\left(y_i + q_i - \frac{\mu}{w_i} + \frac{q_i}{p_i}\left(r_i - w_i - \frac{\mu}{q_i}\right)\right), \end{aligned}$$

where y_i and q_i are the Lagrange multipliers of the lower and upper ranges, respectively, and $v_i = y_i + q_i$. If the upper range does not exist, the corresponding terms vanish and the system reduces to (7). If the lower range does not exist, the modeling system AMPL [12] multiplies the constraint by -1 and shifts it to convert the problem into (1).

In the current distribution of LOQO, equality constraints are handled by treating them as range constraints with $r_i = 0$. However, this forces both $w_i = 0$ and $p_i = 0$ in the optimal solution, which means that $y_i - v_i - q_i = 0$, $v_i \geq 0$, and $q_i \geq 0$ are the only requirements to determine the values of the Lagrange multipliers v_i and q_i. This results in the optimal face of dual solutions being unbounded, and an interior-point method seeking the analytic center of the face can encounter numerical problems as v_i and q_i both tend to infinity. In linear programming, it can be shown that such a split is beneficial and a simple shifting mechanism can be used to keep the multipliers bounded. Such shifts can either be accompanied by a shift to the corresponding slack variables (preserving complementarity) or not (preserving primal feasibility), both of which can be recovered in the next step. However, in nonlinear programming, the increased number of variables can slow down progress and the algorithm can have trouble easily recovering complementarity or primal feasibility after a shift. Therefore, we have instead decided not to introduce slack variables for handling equality constraints for the purposes of this paper. Thus, as needed, we apply a small regularization to (9) in the form of

$$F_{ii} = \delta, \tag{10}$$

where δ is a small constant equal to the square root of the machine precision. This small constant can be beneficial to ensure a nonzero pivot element during the solution of (9).

Similarly, in the current distribution of LOQO, free variables are split using a scheme described in [22]. As with equality constraints, we have also stopped splitting free variables in the implementation for this paper. As documented in [2], for the case of unconstrained NLPs, we have seen a significant decrease in the number of iterations when we did not split free variables, and we have observed the same behavior for constrained NLPs as well.

2.2. **Numerical Cholesky Factorization.** Related to the issue of handling problems in general form is the solution of (9). In LOQO, there are both primal and dual orderings provided for the modified Cholesky factorization routine. A primal ordering gives precedence to the columns corresponding to Δx in (9), that is

$$\begin{aligned} \Delta x &= -(H(x,y) + E)^{-1}(\sigma - A(x)^T \Delta y) \\ \Delta y &= \left(A(x)(H(x,y) + E)^{-1}A(x)^T + F\right)^{-1}(\rho + A(x)(H(x,y) + E)^{-1}\sigma). \end{aligned} \tag{11}$$

On the other hand, a dual ordering gives precedence to the columns corresponding to Δy in (9), that is

$$\begin{array}{rcl} \Delta y & = & F^{-1}(\rho - A(x)\Delta x) \\ \Delta x & = & -\left(H(x,y) + E + A(x)^T F^{-1} A\right)(\sigma - A(x)^T F^{-1}\rho). \end{array} \tag{12}$$

The default ordering in LOQO is the dual ordering, which necessitates the use of (10) when not splitting equality constraints. For further information on the orderings implemented in LOQO, we refer the reader to [20].

2.3. **Levenberg-Marquardt Perturbation.** At each iteration of the algorithm described above

$$\mathcal{M}(x,w,y) := \begin{bmatrix} -H(x,y)+E & A(x)^T \\ A(x) & F \end{bmatrix}$$

is factored using modified Cholesky factorization, that is,

$$\mathcal{M}(x,w,y) = LDL^T$$

where L is a lower-triangular and D is a diagonal matrix. If $H(x,y)+E$ is positive definite, we have that $\mathcal{M}(x,w,y)$ is quasidefinite, which in turn implies $D_{ii} < 0$ for $i = 1,\ldots,n$ and $D_{ii} > 0$ for $i = n+1,\ldots,n+m$. (As mentioned above, a sparsity-preserving permutation is applied to these matrices, which we omit here for ease of notation.) It also implies that the system has a well-defined solution which is a direction of descent for an appropriately chosen filter or merit function.

However, if the inertia of $\mathcal{M}(x,w,y)$ is not correct, a descent direction cannot be guaranteed, and, in fact, a solution to (7) may not be well-defined. Therefore, once the factorization is complete and $\max_{i=1,\ldots,n} D_{ii} > 0$, we perturb $\mathcal{M}(x,w,y)$ by replacing $H(x,y)+E$ with $H(x,y)+E+\lambda I$. It is possible to use incomplete Cholesky factorization here as well, and stop once the first positive diagonal entry is encountered. In our numerical experience, however, the minimum value of λ to ensure that $H(x,y)+E+\lambda I \succ 0$ is related to $\max_{i=1,\ldots,n} D_{ii}$ and a few complete factorizations using a better estimate of the perturbation can prevent many incomplete factorizations.

The choice of λ is important for the quality of the step direction. It is of course possible to pick a very large value of λ, ensuring that

$$\tilde{\mathcal{M}}(x,w,y;\lambda) := \begin{bmatrix} -(H(x,y)+E+\lambda I) & A(x)^T \\ A(x) & F \end{bmatrix}$$

is quasidefinite in most cases, and increasing it rarely as needed. In fact, as $\lambda \to \infty$, the matrix $H(x,y)+\lambda I$ becomes diagonally dominated, and the resulting step direction from solving

$$\tilde{\mathcal{M}}(x,w,y;\lambda)\begin{pmatrix}\Delta x \\ \Delta y\end{pmatrix} = \begin{pmatrix}\sigma \\ \rho\end{pmatrix} \tag{13}$$

approaches the steepest descent direction obtained by simply replacing $H(x,y)+E$ with λI. Additionally, $\|\Delta x\|$ is a strictly decreasing function of λ (see (12)), so a large value of λ can also result in a very short step. While this direction guarantees descent, it exhibits a worse convergence rate than Newton's Method and results in slow progress for the algorithm. On the other hand, if λ is too small, we could have numerical issues due to $H+E+\lambda I$ being nearly singular. Most theoretical results

ignore such concerns, since the subsequent line search will ensure that we take a small step that nevertheless gives progress for the algorithm.

It should also be noted that when using a dual ordering (12), there is a natural, positive semidefinite perturbation to $H+E$ from $A(x)^T F^{-1} A(x)$, which may mean that we need fewer and smaller additional perturbations in the form of λI when using a dual ordering than when using a primal ordering.

Most implementations use a similar approach for finding a suitable value of λ. If our initial guess $\lambda^{(0)}$ is such that $H(x,y) + E + \lambda^{(0)} I \not\succ 0$, we double our guess, that is $\lambda^{(i)} = 2\lambda^{(i-1)}$ until $H(x,y)+E+\lambda^{(n)} I \succ I$, and we let $\lambda = \lambda^{(n)}$. Otherwise, we halve our guess, that is $\lambda^{(i)} = \frac{\lambda^{(i-1)}}{2}$ until $H(x,y) + E + \lambda^{(n)} I \not\succ I$, and we let $\lambda = 2\lambda^{(n)}$. The resulting value of λ is used to compute the search direction and a line search is performed, as described above, to find a steplength.

In implementation, λ must be carefully chosen, and this popular approach of doubling/halving is not sufficient to guarantee a favorable computational performance. The issue is even more complicated since the determination of a suitable λ can significantly affect the efficiency of the algorithm as it requires the factorization of the Hessian for every trial value.

2.4. **Cubic Regularization for Constrained NLP.** In [2], we gave an overview of a cubic regularization approach for unconstrained NLP and how it can be used in the context of a primal barrier method. We give an overview here to motivate our discussion on the adaptation of cubic regularization to primal-dual interior-point methods.

Let us first start with the unconstrained case, so let $m = 0$ and let L be the Lipschitz constant for $\nabla^2 f(x,y)$. Then, we can also define

$$f_L(x^k + \Delta x) := f(x^k) + \nabla f(x^k)^T \Delta x + \frac{1}{2}\Delta x^T \nabla^2 f(x^k) \Delta x + \frac{L}{6}\|\Delta x\|^3, \tag{14}$$

which has the property that $f_L(x^k + \Delta x) \geq f(x^k + \Delta x)$ for all $\Delta x \in \mathbb{R}^n$. While this gives us a means of building a solution method for (1) with $m = 0$, explicit determination of L for general unconstrained NLP requires significant computational effort. Instead, we use an approximation, M, to the Lipschitz constant and define

$$f_M(x^k + \Delta x) := f(x^k) + \nabla f(x^k)^T \Delta x + \frac{1}{2}\Delta x^T \nabla^2 f(x^k) \Delta x + \frac{M}{6}\|\Delta x\|^3. \tag{15}$$

The cubic step direction is found by solving the problem

$$\Delta x \in \arg\min_s f_M(x^k + s). \tag{16}$$

In [15], [19], and [5], it is shown that for sufficiently large M, $x^k + \Delta x$ will satisfy an Armijo condition. On the other hand, for $x^k \in \mathcal{N}$, having a small M so that $f_M(x^k + \Delta x) \approx f_N(x^k + \Delta x)$ would take advantage of the fast local convergence of Newton's Method close to the optimal solution. Therefore, to ensure sufficient descent at each iteration and to allow for an efficient solution method, we need to control the approximation to the Lipschitz constant, M, rather than impose a steplength α as in Newton's Method.

The basic outline of the Adaptive Cubic Regularization method to solve (1) with $m = 0$ is given as Algorithm 1. The main step of the algorithm determines Δx, and it is nontrivial as it involves the solution of an unconstrained NLP, albeit one with a special form that can be exploited. In [5], the authors propose solving the

Set $k = 0$ and pick a suitable x^0, $M > 0$, $\epsilon > 0$, $\beta > 1$, $\nu > 0$, $0 < \chi < 1$.
while $\|\nabla f(x^k)\| > \epsilon$ **do**
  Find $\Delta x \in \arg\min_s f_M(x^k + s)$.
  if $f(x^k + \Delta x) > f(x)$ **then**
    $M \leftarrow \beta M$.
  else
    if $f(x^k + \Delta x) < f(x) + \nu \nabla f(x)^T \Delta x$ **then** $M \leftarrow \chi M$.
    $x^{k+1} \leftarrow x^k + \Delta x$, $k \leftarrow k + 1$.
  end
end

Algorithm 1: Basic outline of the adaptive cubic regularization algorithm for unconstrained NLPs as given in [15], [19], and [5].

optimization problem only approximately and show that the resulting algorithm retains the convergence properties of [19].

Another important issue here is the choice of the regularization parameter, M. A simple approach to initializing M is to start with a very small value and let the first iteration of Algorithm 2 raise it to be sufficiently large. Then, picking β and χ close to 1 will ensure controlled progress. In practice, however, it may be good to be aggressive when decreasing M, especially for well-behaved convex problems, to reduce both iteration counts and the runtime.

For constrained NLP, in [2], we presented a primal barrier method applied to the problem after relaxing the constraints in order to alleviate the initialization difficulties encountered by a feasible approach. Letting the relaxed problem be

$$\begin{array}{lll} \text{minimize} & f(x) + d^T\xi & \\ \text{subject to} & g(x) + \xi & \geq \quad 0 \\ & \xi & \geq \quad 0, \end{array}$$

where $\xi \in \mathbb{R}^m$ are relaxation variables and $d \in \mathbb{R}^m$ are the penalty parameters, we converted it to an unconstrained NLP using the following barrier problem

$$\text{minimize } f(x) + d^T\xi - \mu \sum_{i=1}^{m} \log(g_i(x) + \xi_i) - \mu \sum_{i=1}^{m} \log(\xi_i)$$

with $\mu \geq 0$ as the barrier parameter. For each value of μ, we can use Algorithm 1 to solve this problem using adaptive cubic regularization. Outer loops provide decreasing values of μ and increasing values of d, as needed, in order to obtain a solution of (1). While the algorithm needs to ensure that $g_i(x) + \xi_i > 0$ and $\xi_i > 0$ at each iteration, no additional linesearch is needed to guarantee sufficient descent. However, as in the unconstrained case, choice of the regularization parameter, M, and the solution of a system that is nonlinear in the search directions at each iteration can significantly affect the number of iterations and the runtimes of the algorithm.

In order to adapt this approach to the primal-dual case, let us modify the approach we took in [2] for the constrained case. We start with exploring properties of the Lagrangian function associated with (3):

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

Note that for each value of μ, we are trying to find a saddle point of $L(x, w, y; \mu)$, that is, we are trying to solve the problem

$$\max_{y} \min_{x,w} L(x, w, y; \mu).$$

Since the cubic regularization works by providing an upper bound on the function, it can be directly applied to the inner problem.

Another issue for a primal-dual method is the dualization of the second constraint in (4) to obtain the primal-dual system (5). Therefore, we have also decided to separate the minimization of $L(x, w, y; \mu)$ with respect to w as well. Thus, as we solve

$$\max_{y} \min_{w} \min_{x} L(x, w, y; \mu)$$

for some value of w and y, we use cubic regularization only on the innermost problem:

$$\begin{aligned} L_M(x^k + \Delta x, w, y; \mu) := \\ \left(f(x^k) - y^T g(x^k) + (\nabla f(x) - A^T y)^T \Delta x + \tfrac{1}{2}\Delta x^T H \Delta x + \tfrac{M}{6}\|\Delta x\|^3\right) \\ -\mu \sum_{i=1}^{m} \log(w_i) + y^T w. \end{aligned}$$

The corresponding nonlinear primal-dual system to be solved at each iteration is

$$\begin{bmatrix} H(x,y) + \frac{M}{2}\|\Delta x\| & 0 & -A(x)^T \\ 0 & Y & W \\ A(x) & -I & 0 \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta w \\ \Delta y \end{pmatrix} = \begin{pmatrix} -\nabla f(x) + A(x)^T y \\ \mu e - WYe \\ -g(x) + w \end{pmatrix} \tag{17}$$

which can then be reduced to

$$\begin{bmatrix} -\left(H + \frac{M}{2}\|\Delta x\| I\right) & A^T \\ A & W^{-1}Y \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} \nabla f(x) - A^T y \\ g(x) - \mu Y^{-1} e \end{pmatrix}, \tag{18}$$

by symmetrizing as before and eliminating Δw.

For a general form NLP, (18) becomes

$$\begin{bmatrix} -\left(H + \frac{M}{2}\|\Delta x\| I + E\right) & A^T \\ A & F \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} \sigma \\ \rho \end{pmatrix}, \tag{19}$$

where E, F, σ, and ρ have the same form as before.

2.5. **A Hybrid Approach.** In [2], we noted that the cubic regularization approach is equivalent to using Levenberg-Marquardt regularization for a sufficiently large value of λ, solving for the step direction, and setting $M = \frac{2\lambda}{\|\Delta x\|}$ for the unconstrained case and $M = \frac{2\lambda}{\|(\Delta x, \Delta \xi)^T\|}$ for the constrained case. It can be seen from (18) and (13), that the same equivalence holds for a primal-dual interior-point method, as well. Such an equivalence would also suggest that we use cubic regularization only in cases where the Levenberg-Marquardt regularization would be used, that is, during those iterations where H is not positive definite. Doing so would allow us to use a theoretically robust way to find the step direction when H is not positive definite, and while limiting the additional amount of computational work required by using the cubic step. The hybrid algorithm is described as Algorithm 2.

We should also note an implementation detail here regarding the Shifted Inverse Power Method used in Algorithm 2. A larger than necessary shift is applied to the Hessian first, and we would like to find the smallest magnitude eigenvalue of this

Set $k = 0$.
while $res(x^k, \xi^k) > \epsilon$ **do**
  if $H(x^k, y^k) + E \succeq 0$ **then**
    Solve $\mathcal{M}(x^k, y^k) \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} \sigma \\ \rho \end{pmatrix}$.
    Compute steplength α that provides sufficient descent using a line search in $(0, 1]$ and preserves nonnegativity of slack and dual variables.
  else
    $\lambda = \max_{j=1,\ldots,n} D_{jj}$.
    while $\max_{j=1,\ldots,n} D_{jj} > 0$ **do**
      $\lambda \leftarrow \lambda/5$.
      Find D such that $\tilde{\mathcal{M}}(x, y; \lambda) = LDL^T$.
    end
    while $\max_{j=1,\ldots,n} D_{jj}(\lambda) < 0$ **do**
      $\lambda \leftarrow 5\lambda$.
    end
    Use the Shifted Inverse Power Method (shift = λ) with Aitken acceleration to obtain the minimum eigenvalue of $H(x^k, y^k)$, denoted by $\lambda_{min}(x^k, y^k)$. Set $\lambda \leftarrow -\beta\lambda_{min}(x^k, y^k)$.
    Solve (13) with (x^k, y^k).
    Compute steplength $\alpha \leq 1$ that preserves nonnegativity of slack and dual variables.
  end
  $x^{k+1} \leftarrow x^k + \alpha\Delta x$, $y^{k+1} \leftarrow y^k + \alpha\Delta y$. Update all other slack and dual variables accordingly. Update μ. $k \leftarrow k + 1$.
end

Algorithm 2: A description of the solution algorithm for a general unconstrained or constrained NLP. This algorithm uses a cubic regularization applied only when $H(x^k, y^k) + E$ is indefinite.

shifted Hessian, rather than of the entire shifted $\mathcal{M}$. Therefore, only when applying the inverse power method, we replace the nonzero elements of A with zeros and let the entries of the diagonal matrix F equal twice the shift. Doing so ensures that we find the correct shift for the Hessian matrix for a primal ordering. It may be too large of a shift in the case of a dual ordering, however, the effort required to compute $H(x, y) + E + A(x)^T F^{-1} A(x)$ was prohibitive for large problems in our numerical testing.

3. Numerical Results

In our numerical testing, we compare the performance of a Newton-based code to the hybrid algorithm outlined as Algorithm 2. We have modified LOQO [23] (Version 6.06) by removing the splitting of free variables and equality constraints, and the resulting code serves as the Newton's Method-based infeasible primal-dual interior-point method. We have implemented the hybrid algorithm by modifying this source code. Both implementations use the same linear algebra routines, AMPL

	Unconstrained NLPs		**Constrained NLPs**	
Solver	*Iterations*	*Runtime*	*Iterations*	*Runtime*
LOQO	8411	251.18	24410	698.39
Cubic Hybrid	8110	232.27	22933	729.66

TABLE 1. Overall iteration and runtimes (in CPU seconds) for LOQO and the hybrid cubic codes on unconstrained and inequality constrained NLPs from the CUTEr test set.

[12] interface, and other utilities, so the analysis done within the context of a commercial code is a good way to assess the potential of the hybrid cubic regularization approach in real-world application.

Numerical testing was conducted on a laptop running Fedora Core 8 with 3GB of main memory and dual CPUs with 2.4GHz clock speeds. We did not specifically take advantage of the dual CPUs in our implementation. The numerical testing was conducted on a wide range of problems, including problems from the Hock and Schittkowski [16] and the CUTEr [9] test suites, for which the AMPL models were obtained from [21]. We have removed a small number of problems that were infeasible, unbounded, or nondifferentiable at either the user-supplied initial solution or at the optimal solution.

Table 1 gives the total iteration counts and runtimes for the Newton step and the hybrid cubic step on the whole suite of problems tested. Results on individual problems can be found online at [1].

In our test suite, there were 232 unconstrained problems, all of which were solved by both codes. We also tested 748 constrained NLPs, and LOQO solved 695 of those problems whereas the hybrid cubic code solved 700. Of these, 690 problems were solved by both codes, and LOQO solved 5 problems on which the hybrid cubic code failed, whereas the hybrid cubic code solved 10 problems on which LOQO failed. Overall, the hybrid cubic code dominates LOQO with respect to total runtime and number of iterations on unconstrained NLPs, as seen in Table 1. The values in the table for constrained NLPs are over the whole test set, so the totals are affected by the hybrid cubic code solving more problems than LOQO. When we focus only on those problems solved by both codes, we see that the hybrid cubic code dominates LOQO with respect to both measures again: LOQO takes 24016 iterations in 698.37 seconds, and the hybric cubic code takes 21535 iterations in 674.91 seconds.

The results presented in Table 1 and the previous paragraph indicate that including cubic regularization in our code results in improving our overall iteration counts, total runtime, and number of problems solved. A closer look at the results through performance profiles puts the cubic hybrid in an even more positive light. Performance profiles were proposed by Dolan and Moré [10] and compute an estimate of the probability that an algorithm performs within a multiple of the runtime or iteration count (or any other metric) of the best algorithm. We briefly describe the methodology here and refer the reader to [10] for further details. Assume that we are comparing the performance of n_s solvers on n_p problems using runtime as the performance metric. Letting

$$t_{p,s} = \text{computing time required to solve problem } p \text{ by solver } s$$

and

$$t_p^* = \min\{t_{p,s} : 1 \leq s \leq n_s\},$$

the *performance ratio* of solver s on problem p is defined by

$$r_{p,s} = \frac{t_{p,s}}{t_p^*}.$$

If solver p cannot solve problem s, $r_{p,s} = \infty$. In order to evaluate the performance of the algorithm on the whole set of problems, we can use the following estimate to the cumulative distribution function of the performance ratio:

$$P(t_{p,s} \leq \tau t_p^*) \approx \frac{1}{n_p}\text{card}\{p : 1 \leq p \leq n_p, r_{p,s} \leq \tau\}.$$

Plotting the cumulative distribution function versus τ for the two solvers on the same plot yields a comparison of their performance over the whole test set, while still being based on comparisons on individual problems.

In Figures 1-4, we present performance profiles of the Newton and cubic hybrid codes on unconstrained and constrained problems using two different metrics. Figures 1 and 3 show the performance profiles of the two solvers with respect to iteration counts and Figures 2 and 4 use runtime (in CPU seconds) as their metric. While all of the problems were included for the iteration count comparisons, we have only included 56 unconstrained problems in Figure 2 and 220 constrained problems in Figure 4. The reason for choosing a smaller subset of the problems for the runtime comparisons is that we have not included models whose runtimes were less than 0.05 CPU seconds. For runtimes less than 0.05 CPU seconds, we have found that small differences in system performance could have provided an unfair comparison. We have also limited the runtime comparisons to those problems that were solved by both solvers, which did not affect the conclusions of a pairwise comparison.

We start with a discussion of the unconstrained NLPs, and Figure 1 shows that cubic hybrid approach is superior to the Newton method with respect to iteration count comparisons on individual problems. On 85% of the problems, the cubic hybrid approach gives the same or a smaller iteration count than the Newton approach, and it also retains a higher cumulative probability for $\tau \leq 2$. (Only 9 problems gave performance ratios of more than 2 for LOQO and 3 problems for the cubic hybrid approach.) It should also be noted that the cubic regularization was invoked by 118 of the 232 problems, or roughly 51% of the test set. For the 118 instances where the cubic regularization is invoked, the two codes yield the same number of iterations on 22 instances, the Newton approach of LOQO takes a smaller number of iterations on 35 instances, and the cubic hybrid approach is the winner on the remaining 61. Thus, we can conclude that cubic regularization yields the same or a better number of iterations on 71% of the cases where it is invoked.

Even though the aggregated results of Table 1 would have suggested a similar performance from the two codes, problem-by-problem comparisons of the performance profile indicates that the cubic hybrid approach performed better more frequently and more consistently than the Newton method. The closeness of the total iteration counts in Table 1 can be attributed to there being a large number of problems with small iteration counts and small improvements, and a very small number of ill-behaved problems where the cubic hybrid approach took a significantly larger number of iterations. In general, for problem families such as *pfit* and *palmer*, where there is at least one degree of freedom, the resulting numerical issues near the optimal solution can have a significant impact on the iteration counts. Since

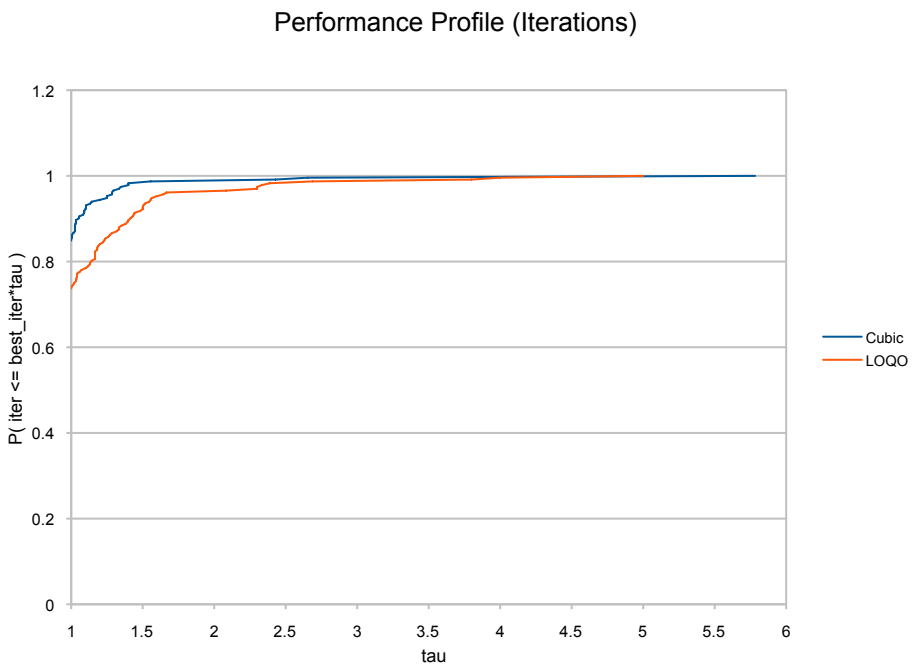

FIGURE 1. Performance profiles of the Newton and cubic hybrid codes on unconstrained NLPs. The metric used in this figure is the iteration count.

the neighborhood around the optimal solution yields a positive definite Hessian, the cubic regularization is used only within the first few iterations for these problems, with the remaining 50-300 iterations taking place in the neighborhood using a Newton step. There is little reason to suspect that a purely cubic approach can yield significantly better performance here, as approaches such as that implemented in [5] are designed to provide small to no perturbation close to the optimal solution to take advantage of the convergence properties of the Newton step.

For the runtime comparison of the two approaches on unconstrained problems, we refer to Figure 2. As discussed above, we have limited our attention to those problems where the best solver took at least 0.05 CPU seconds. Additionally, if the cubic hybrid regularization was not invoked, we disregard the calculated performance ratio (which may be influenced by small perturbations in system performance) and simply declare both solvers to have a performance ratio of 1 for those problems. On 22 of the 56 problems where the runtime was large enough, cubic regularization was never invoked. On the other 34 problems, the two codes took the same amount of time (to 5 digits of accuracy) on 1 problem, the Newton method was the fastest on 17 problems, and the cubic hybrid approach was the fastest on 16 problems. As the problem size increases, the advantage of cutting iterations via the hybrid cubic approach becomes apparent: of the 10 problems where both

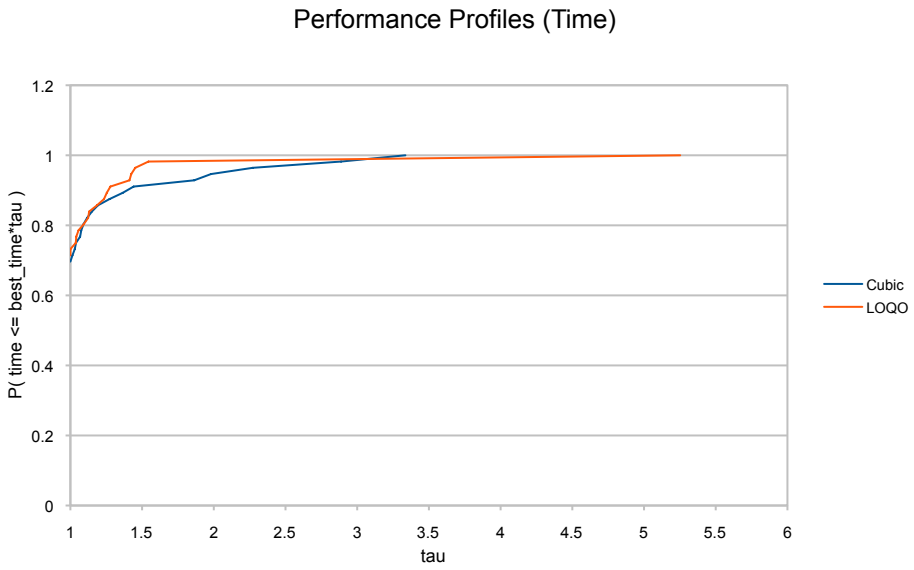

FIGURE 2. Performance profiles of the Newton and cubic hybrid codes on unconstrained NLPs. The metric used in this figure is the runtime and only problems for which the solvers took more than 0.05 CPUs were included.

solvers take more than 1 CPU second and the cubic regularization is invoked, the hybrid cubic code is faster than LOQO on 6 instances.

Even though they match the minimum performance ratios on 70-70% of the problems each, the cubic hybrid code performs slightly worse than the Newton approach for τ values around 2. For example, the Newton approach solves roughly 96% of the problems within 1.5 times the best runtime, whereas the cubic hybrid code does so on roughly 91% of the problems. However, the cubic hybrid code is more consistent, with τ values of no more than 3.34 over the whole set, whereas LOQO solves one problem in 5.25 times the runtime of the cubic hybrid code. In general, then, the reduction in the number of iterations due to the additional work performed per iteration tends to either improve or not appreciably hurt the runtime of the code for unconstrained NLPs. For example, nonconvex problems *msqrtals* and *msqrtbls* each see a reduction of 7-8 iterations, which yields 26-28% reductions in the number of iterations and 30-35% reductions in runtime. The family *curly* and their scaled versions *scurly* have a nearly 30% reduction in the number of iterations while the overall runtime remains nearly the same.

The few cases where the Newton approach significantly outperforms the cubic hybrid (with respect to runtime) generally correspond to those problems where a large number of additional factorizations may be required to find a suitable value of the Levenberg-Marquardt perturbation parameter. For example, *fminsurf* only requires a perturbation of 10^{-13} to give a positive definite modified Hessian, but a perturbation on the order of 10^{-3} is needed to guarantee sufficient descent for the objective function. Given our conservative choice of 1.5 for $\hat{\beta}$, this requires more

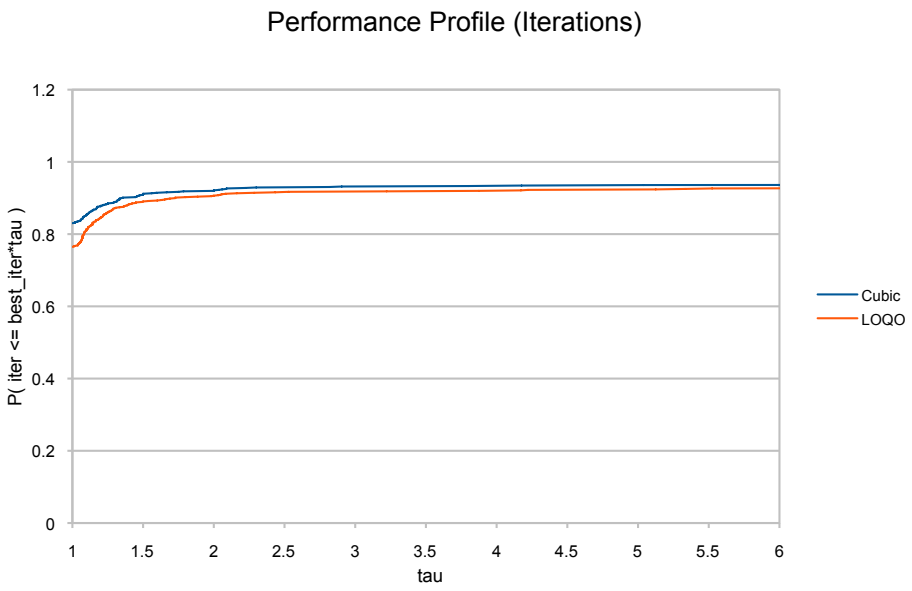

FIGURE 3. Performance profiles of the Newton and cubic hybrid codes on constrained NLPs. The metric used in this figure is the iteration count.

than 50 factorizations of the KKT matrix until we find a suitable perturbation in the first iteration. We have included heuristics to remember this suitable perturbation and use it to speed up our search in subsequent iterations, but the work required to find it in the first iteration dominates our solution time and the cubic hybrid approach is slower on that problem. A more sophisticated approach to the line search for the perturbation may yield faster results and will be considered in future work.

We now switch our attention to constrained NLPs. Performance profiles for the two codes with respect to iteration counts and runtimes are given in Figures 3 and 4, respectively. The overall success of the cubic hybrid approach is similar to the case of unconstrained NLPs. As shown in Figure 3, on 83% of the problems, the cubic hybrid approach gave the same or better iteration counts and also retained a higher cumulative probability for all values of τ. Cubic regularization was invoked on 292 of the 748 constrained problems, with the two codes jointly solving 243 of the 292 problems: giving the same iteration counts on 42, LOQO performing fewer iterations on 78, and the cubic hybrid approach as the winner on the remaining 123. On the 49 problems with one or more failures, both solvers failed on 34 problems, the hybrid cubic method solved 10 problems that LOQO did not, and LOQO solved 5 problems that the hybrid cubic approach did not. This means that, when it is invoked, the cubic regularization yields the same or better performance on 72% of the problems.

It should be noted that significant portion of the difference between the total number of iterations for constrained NLPs, as shown in Table 1, is due to a small

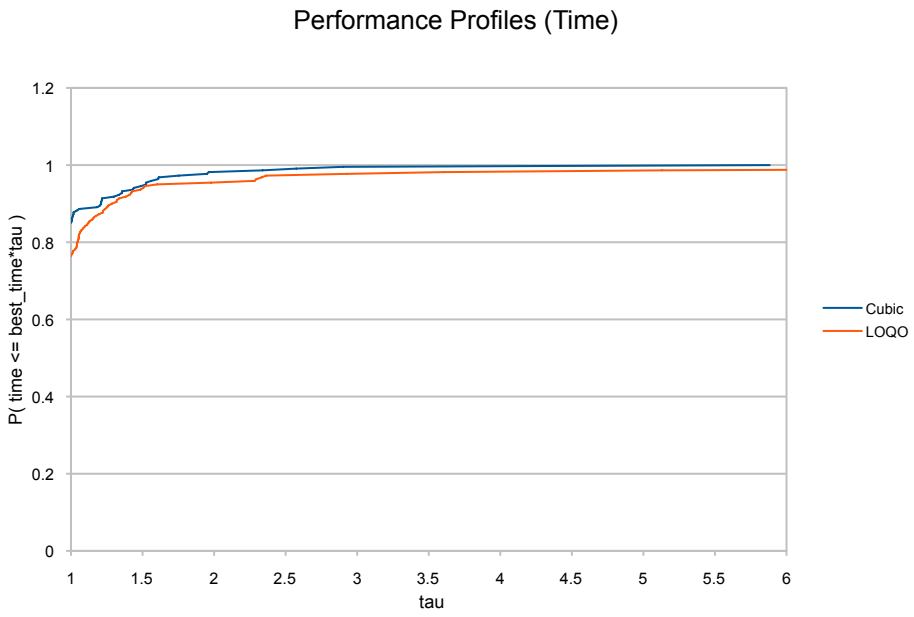

FIGURE 4. Performance profiles of the Newton and cubic hybrid codes on constrained NLPs. The metric used in this figure is the runtime and only problems for which the solvers took more than 0.05 CPUs were included.

group of problems: *deconvc*, *dixchlnv*, *ncvxqp4*, *palmer7e*, *qpnboei1*. On each of these problems, the hybrid cubic approach is 192-756 iterations faster that LOQO, for a total difference of 1950 iterations. Regardless, the pairwise comparisons, as shown by the performance profiles of Figure 3, show that the hybrid cubic code outperforms LOQO with respect to iterations over the whole set of constrained NLPs.

To examine the benefits of using a cubic step in our hybrid code, we refer to problem *s219*. LOQO takes 30 iterations to locate the optimum that the hybrid cubic approach finds in only 15 iterations. In the first iteration, LOQO uses a perturbation of 0.0438, whereas the cubic hybrid approach uses a perturbation of 0.0360. Using the shifted inverse power method allows for a better estimate of the minimum eigenvalue of the Hessian, which in turns results in a smaller perturbation to achieve both positive definiteness of the Hessian and sufficient descent. Thus, the cubic hybrid approach takes a longer step in the first iteration and continues making similar progress to solve the problem in half the number of iterations of LOQO. We should also note that, in the first iteration, LOQO requires 10 refactorizations of the Hessian to obtain this value, whereas, the cubic hybrid approach only needs 7 refactorizations and 2 backsolves. Since *s219* is a small problem with only 4 variables and 2 constraints, there is no appreciable speed up per iteration, but similar behavior on a larger problem can save a significant amount of time.

For the runtime comparison on constrained NLPs, we refer to the performance profiles in Figure 4. Again, only runtimes of 0.05 CPU seconds or more were

included, so 220 problems were used for this comparison. Of the 220 problems, 135 did not invoke cubic regularization, so we have used performance ratios of 1 for both solvers on those problems. For the remaining 85 problems, LOQO was faster on 33 instances, and the cubic hybrid approach was faster on the other 52. The cubic hybrid approach is also more consistent, with all instances falling within $\tau = 5.88$, whereas there are two instances for which LOQO exceeds $\tau = 20$. These results indicate that the additional work per iteration is offset by the reduction in iterations. The overall improvement in the solution times as exhibited in the performance profiles of Figure 4, vis-a-vis those in Figure 2 for unconstrained NLPs, can be explained by the larger sample set which gives a better comparison and also by the success of the cubic regularization step on problems with nonconvexities and ill-conditioning, both of which are more prevalent among the constrained NLPs.

4. Conclusion

We have incorporated cubic regularization into a line search primal-dual interior-point method for both unconstrained and constrained NLPs. When used only on iterations with indefinite Hessians, this approach is equivalent to picking a suitable value for the Levenberg-Marquardt parameter and, therefore, retains the convergence properties of the original interior-point method. Additionally, this selective use of cubic regularization ensures that we retain the iteration-reduction benefits of the approach without the significant cost to the runtimes. In fact, our numerical studies show that, in general, the cubic hybrid approach improves the iteration count and provides the same or slightly better same runtime than using the Newton step at each iteration. This result is significantly better than that presented in [14], where the iteration counts were improved but the resulting algorithm was significantly slower on a certain set of problems.

In [2], we had studied the cubic hybrid approach within the context of a purely primal barrier method. While the hybrid approach had improved the numbers of iterations and the runtimes, the underlying barrier method was not a viable candidate for implementation as a competitive commercial code. In this paper, we have extended our approach to be used within a state-of-the-art infeasible primal-dual interior-point code and further improved its performance.

References

[1] H.Y. Benson. Numerical testing results for the hybrid cubic approach. http://www.pages.drexel.edu/~hvb22/CubicWeb.

[2] H.Y. Benson and D.F. Shanno. Interior-point methods for nonconvex nonlinear programming: Cubic regularization. Technical report, 2012.

[3] 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.

[4] R.H. Byrd, J. Nocedal, and R.A. Waltz. KNITRO: An integrated package for nonlinear optimization. In *Large-Scale Nonlinear Optimization* by G. Di Pillo and M. Roma, eds., pages 35–59. Springer US, 2006.

[5] C. Cartis, N.I.M Gould, and Ph.L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. Part I: motivation, convergence and numerical results. *Mathematical Programming, Series A*, 127:245–295, 2011.

[6] C. Cartis, N.I.M Gould, and Ph.L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. Part II: worst-case function- and derivative-evaluation complexity. *Mathematical Programming, Series A*, 130:295–319, 2011.

[7] C. Cartis, N.I.M Gould, and Ph.L. Toint. Evaluation copmlexity of adaptive cubic regularization methods for convex unconstrained optimization. *Optimization Methods and Software*, iFirst:1–23, 2011.
[8] C. Cartis, N.I.M Gould, and Ph.L. Toint. An adaptive cubic regularisation algorithm for nonconvex optimization with convex constraints and its function-evaluation complexity. *IMA Journal of Numerical Analysis*, Online, 2012.
[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] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. Technical report, Argonne National Laboratory, January 2001.
[11] R. Fletcher and S. Leyffer. User manual for filterSQP. Technical Report NA-181, University of Dundee Report, 1998.
[12] R. Fourer, D.M. Gay, and B.W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Scientific Press, 1993.
[13] P.E. Gill, W. Murray, and M.A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12:979–1006, 2002.
[14] N.I.M. Gould, M. Porcelli, and Ph. L. Toint. Updating the regularization parameter in the adaptive cubic regularization algorithm. *Computational Optimization and Applications*, 2011.
[15] A. Griewank. The modification of Newton's method for unconstrained optimization by bounding cubic terms. Technical Report NA/12, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, 1981.
[16] W. Hock and K. Schittkowski. *Test examples for nonlinear programming codes*. Lecture Notes in Economics and Mathematical Systems 187. Springer Verlag, Heidelberg, 1981.
[17] Yu. Nesterov. Cubic regularization of Newton's method for convex problems with constraints. Technical Report 39, CORE, 2006.
[18] Yu. Nesterov. Accelerating the cubic regularization of Newton's method on convex problems. *Mathematical Programming, Series B*, 112:159–181, 2008.
[19] Yu. Nesterov and B.T. Polyak. Cubic regularization of Newton method and its global performance. *Mathematical Programming, Series A*, 108:177–205, 2006.
[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] R.J. Vanderbei. AMPL models. http://orfe.princeton.edu/~rvdb/ampl/nlmodels.
[22] R.J. Vanderbei. LOQO user's manual. Technical Report SOR 92-5, Princeton University, 1992. revised 1995.
[23] R.J. Vanderbei and D.F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13:231–252, 1999.
[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, Emeritus, Rutgers University, New Brunswick, NJ