SIAM J. OPTIM.
Vol. 19, No. 4, pp. 1674–1693

ADAPTIVE BARRIER UPDATE STRATEGIES FOR NONLINEAR INTERIOR METHODS*

JORGE NOCEDAL†, ANDREAS WÄCHTER‡, AND RICHARD A. WALTZ†

Abstract. This paper considers strategies for selecting the barrier parameter at every iteration of an interior-point method for nonlinear programming. Numerical experiments suggest that heuristic adaptive choices, such as Mehrotra's probing procedure, outperform monotone strategies that hold the barrier parameter fixed until a barrier optimality test is satisfied. A new adaptive strategy is proposed based on the minimization of a *quality function*. The paper also proposes a globalization framework that ensures the convergence of adaptive interior methods, and examines convergence failures of the Mehrotra predictor-corrector algorithm. The barrier update strategies proposed in this paper are applicable to a wide class of interior methods and are tested in the two distinct algorithmic frameworks provided by the IPOPT and KNITRO software packages.

Key words. interior-point methods, barrier methods, nonlinear programming, constrained optimization

AMS subject classifications. 49M37, 65K05, 90C06, 90C30, 90C51

DOI. 10.1137/060649513

1. Introduction.

In this paper we describe interior methods for nonlinear programming that update the barrier parameter adaptively, as the iteration progresses. The goal is to design algorithms that are both efficient in practice and that enjoy global convergence guarantees. The adaptive strategies studied in this paper allow the barrier parameter to increase or decrease at every iteration and provide an alternative to the so-called Fiacco–McCormick approach that fixes the barrier parameter until an approximate solution of the barrier problem is computed. Our motivation for this work stems from our belief that robust interior methods for nonlinear programming must be able to react swiftly to changes of scale in the problem and to correct overly aggressive decreases in the barrier parameter.

Adaptive barrier update strategies are well established in interior methods for linear and convex quadratic programming. The most popular approach of this type is Mehrotra's predictor-corrector (MPC) method [22]. It computes, at every iteration, a probing (affine scaling) step that determines a target value of the barrier parameter, and then takes a primal-dual step using this target value. A corrector step is added to better follow the trajectory of the central path to the solution. Mehrotra's method has proved to be very effective for linear and convex quadratic programming, but is not supported by global convergence guarantees. Indeed, as we show in section 7, its reliability is heavily dependent upon an appropriate choice of the starting point.

When solving nonlinear nonconvex programming problems, much caution must be exercised to prevent the iteration from failing. Nonminimizing stationary points can attract the iteration, and aggressive decreases in the barrier parameter can lead to failure. Our numerical experience shows that the direct extension of Mehrotra's

*Received by the editors January 9, 2006; accepted for publication (in revised form) August 12, 2008; published electronically January 28, 2009.
http://www.siam.org/journals/siopt/19-4/64951.html

†Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, 60208-3118 (nocedal@eecs.northwestern.edu, rwaltz@usc.edu). The work of these authors was supported by National Science Foundation grants CCR-0219438 and DMI-0422132, and Department of Energy grant DE-FG02-87ER25047-A004.

‡Department of Mathematical Sciences, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598 (andreasw@watson.ibm.com).

predictor-corrector method to nonlinear programming does not result in a robust method. As we discuss below, the main source of instability is the corrector step. Further adaptive barrier update strategies designed specifically for nonlinear programming include [2, 11, 15, 23, 24, 25].

The global convergence properties of interior methods for nonlinear programming have recently received much attention [4, 8, 11, 17, 20, 23, 24, 26, 32]. Some of these studies focus on the effects of merit functions or filters, and on regularization techniques. With the exception of [11, 23, 24], however, these papers do not consider the numerical or theoretical properties of adaptive barrier update techniques.

The organization of this paper is as follows. After stating the basic nonlinear interior method in section 2, we start our exploration of adaptive barrier updates by examining several established techniques in section 3. In this initial investigation, we do not impose a rigorous globalization scheme on the methods, but simply compare their practical behavior on a standard test set. Motivated by the initial observations of these experiments, we then

- propose (in section 4) a new strategy for choosing the barrier parameter that, in contrast to previously proposed update rules, is not based on heuristics but follows a clear-cut objective, namely, the minimization of a "quality function";
- present two simple frameworks that ensure global convergence for interior methods that use *any* update rule for the barrier parameter (section 5);
- explore the numerical performance of the proposed strategy on standard test sets (section 6); and
- discuss the shortcomings of the Mehrotra corrector step (which can be observed even in the linear case) and propose a remedy (section 7).

To show the generality of our quality function approach, we implement it in the two different algorithmic contexts provided by the IPOPT [27] and KNITRO [6, 28] software packages.

Notation. For any vector z, we denote by Z the diagonal matrix whose diagonal entries are given by z. We let e denote the vector of ones, of appropriate dimension, that is, $e = (1, 1, \ldots, 1)^T$.

2. Primal-dual nonlinear interior methods. The problem under consideration will be written as

$$\min_x \ f(x), \tag{2.1a}$$

$$\text{s.t. } c(x) = 0, \tag{2.1b}$$

$$x \geq 0, \tag{2.1c}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$ are twice continuously differentiable functions. For conciseness we will refer to interior-point methods for nonlinear programming as "nonlinear interior methods." A variety of these methods have been proposed in the last 10 years; they differ mainly in some aspects of the step computation and in the globalization scheme. Most of the nonlinear interior methods are related to the simple primal-dual iteration described next; our discussion of barrier parameter choices will be phrased in the context of this iteration.

We associate with the nonlinear program (2.1) the barrier problem

$$\min_x \ \varphi_\mu(x) \equiv f(x) - \mu \sum_{i=1}^{n} \ln x^{(i)}, \tag{2.2a}$$

$$\text{s.t. } c(x) = 0, \tag{2.2b}$$

where $\mu > 0$ is the barrier parameter. As is well known, the KKT conditions of the barrier problem (2.2) can be written as

$$\nabla f(x) - A(x)^T y - z = 0, \tag{2.3a}$$

$$Xz - \mu e = 0, \tag{2.3b}$$

$$c(x) = 0, \tag{2.3c}$$

where $A(x)$ denotes the Jacobian matrix of the constraint function $c(x)$. Condition (2.3b), the positivity of μ, and the requirement that the log function be well-defined in (2.2a) implicitly requires that

$$x > 0, \qquad z > 0. \tag{2.4}$$

Applying Newton's method to (2.3), in the variables (x, y, z), gives the *primal-dual* system

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L} & -A(x)^T & -I \\ Z & 0 & X \\ A(x) & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = - \begin{bmatrix} \nabla f(x) - A(x)^T y - z \\ Xz - \mu e \\ c(x) \end{bmatrix}, \tag{2.5}$$

where $\mathcal{L}$ denotes the Lagrangian of the nonlinear program, that is,

$$\mathcal{L}(x, y, z) = f(x) - y^T c(x) - z^T x. \tag{2.6}$$

After the step $\Delta = (\Delta x, \Delta y, \Delta z)$ has been determined, we compute primal and dual steplengths, α_p and α_d, and define the new iterate (x^+, y^+, z^+) as

$$x^+ = x + \alpha_p \Delta x, \quad y^+ = y + \alpha_d \Delta y, \quad z^+ = z + \alpha_d \Delta z. \tag{2.7}$$

The steplengths are computed in two stages. First we compute

$$\alpha_x^{\max} = \max\{\alpha \in (0, 1] : x + \alpha \Delta x \geq (1 - \tau)x\}, \tag{2.8a}$$

$$\alpha_z^{\max} = \max\{\alpha \in (0, 1] : z + \alpha \Delta z \geq (1 - \tau)z\}, \tag{2.8b}$$

with $\tau \in (0, 1)$ (e.g., $\tau = 0.995$). Next, we perform a backtracking line search to compute the final steplengths

$$\alpha_p \in (0, \alpha_x^{\max}], \qquad \alpha_d \in (0, \alpha_z^{\max}], \tag{2.9}$$

which provide sufficient decrease of a merit function or ensure acceptability by a filter.

The other major ingredient in this simple primal-dual iteration is the procedure for choosing the barrier parameter μ. Two types of barrier update strategies have been studied in the literature: adaptive and monotone. Adaptive strategies [11, 15, 24, 25] allow changes in the barrier parameter at every iteration, and are often efficient in practice, but as already mentioned, they generally do not enjoy global convergence properties. (The analyses presented in [11, 24] provide certain convergence results to stationary points, but these methods do not explicitly aim to decrease the objective function—they only enforce reduction of a measure of stationarity.)

The most important monotone strategy is the so-called Fiacco–McCormick approach that fixes the barrier parameter until an approximate solution of the barrier problem is computed. It has been employed in various nonlinear interior algorithms [3, 5, 14, 16, 27, 29, 31] and has been implemented, for example, in the IPOPT and

KNITRO software packages. The Fiacco–McCormick strategy provides a framework for establishing global convergence [4, 26], but suffers from important limitations. It can be very sensitive to the choice of the initial point, the initial value of the barrier parameter, and the scaling of the problem, and it is often unable to recover quickly when the iterates approach the boundary of the feasible region prematurely. The numerical experience with IPOPT and KNITRO reported below suggests that more dynamic update strategies are needed to improve the efficiency of nonlinear interior methods.

The algorithms considered in this paper guarantee only convergence to first-order stationary points; enforcing convergence to second-order points would require an estimation of the smallest eigenvalue of the reduced Hessian, which is too expensive in the large-scale case. However, the algorithms presented here generate steps that promote convergence to minimizers by ensuring descent properties for the barrier problem.

3. Choosing the barrier parameter. In this section we discuss two adaptive barrier strategies proposed in the literature and compare them numerically with the monotone Fiacco–McCormick approach. These numerical results motivate the techniques presented in the following sections.

Given an iterate (x, y, z), consider an interior method that computes primal-dual search directions by (2.5). The most common approach for choosing the barrier parameter μ is to make it proportional to the current complementarity value, that is,

$$\mu = \sigma \frac{x^T z}{n}, \tag{3.1}$$

where $\sigma > 0$ is a *centering parameter* and n denotes the number of variables. Mehrotra's predictor-corrector (MPC) method [22] for linear programming determines the value of σ using a preliminary step computation (an affine scaling step). We now describe a direct extension of Mehrotra's strategy to the nonlinear programming case.

First, we calculate an affine scaling step

$$\left(\Delta x^{\text{aff}}, \Delta y^{\text{aff}}, \Delta z^{\text{aff}}\right) \tag{3.2}$$

by setting $\mu = 0$ in (2.5), that is,

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L} & -A(x)^T & -I \\ Z & 0 & X \\ A(x) & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x^{\text{aff}} \\ \Delta y^{\text{aff}} \\ \Delta z^{\text{aff}} \end{bmatrix} = - \begin{bmatrix} \nabla f(x) - A(x)^T y - z \\ Xz \\ c(x) \end{bmatrix}. \tag{3.3}$$

We then compute α_x^{aff} and α_z^{aff} to be the largest steplengths in $(0, 1]$ that can be taken along the direction (3.2) before violating the nonnegativity conditions $(x, z) \geq 0$. Explicit formulae for these values are given by (2.8) with $\tau = 1$.

Next, we define μ^{aff} to be the value of complementarity that would be obtained by a full step to the boundary, that is,

$$\mu^{\text{aff}} = \left(x + \alpha_x^{\text{aff}} \Delta x^{\text{aff}}\right)^T \left(z + \alpha_z^{\text{aff}} \Delta z^{\text{aff}}\right) / n, \tag{3.4}$$

and set the centering parameter to be

$$\sigma = \left(\frac{\mu^{\text{aff}}}{x^T z / n}\right)^3. \tag{3.5}$$

This heuristic choice of σ is based on experimentation with linear programming problems and has proved to be effective for convex quadratic programming as well. Note that when good progress is made along the affine scaling direction, we have $\mu^{\text{aff}} \ll x^T z/n$, so the σ obtained from this formula is small. In other cases, σ may be chosen to be greater than 1.

Mehrotra's algorithm also computes a *corrector* step, but we take the view that the corrector is not part of the selection of the barrier parameter, and is simply a mechanism for improving the quality of the step. In section 7 we study the complete MPC algorithm including the corrector step.

Other adaptive procedures of the form (3.1) have been proposed specifically for nonlinear interior methods [11, 15, 24, 25]. The strategy employed in the LOQO software package [25] is particularly noteworthy because of its success in practice. It defines σ as

$$\sigma = 0.1 \min\left(0.05\frac{1-\xi}{\xi}, 2\right)^3, \quad \text{where} \quad \xi = \frac{\min_i \{x^{(i)} z^{(i)}\}}{x^T z/n}. \tag{3.6}$$

Note that ξ measures the deviation of the smallest complementarity product $x^{(i)} z^{(i)}$ from the average. When $\xi = 1$ (all individual products are equal to the average) we have that $\sigma = 0$ and the algorithm takes an aggressive step. The rule (3.6) always chooses $\sigma \leq 0.8$, so that even though the value of μ may increase from one iteration to the next, it will never be chosen to be larger than the current complementarity value $x^T z/n$.

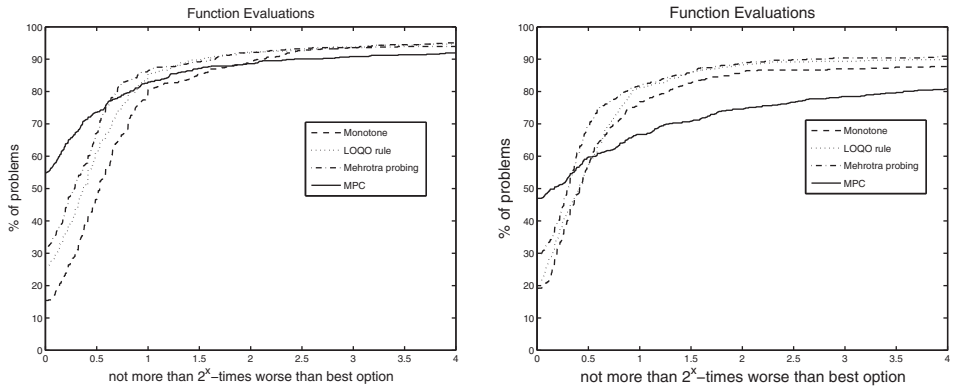
Our first set of numerical experiments compares the effectiveness of the two adaptive strategies mentioned above with the monotone Fiacco–McCormick approach. For these experiments, we use the IPOPT and KNITRO software packages, which have a globalization mechanism for the monotone variant but none for adaptive barrier parameter choices. These codes implement significantly different variations of the simple primal-dual iteration (2.5).

The experiments with KNITRO were done using the default Interior/Direct option (we will refer to this version as KNITRO-DIRECT henceforth), which implements a line search approach that is occasionally safeguarded by a trust region iteration [29]. The trust-region safeguard is needed, for example, to handle negative curvature directions. A merit function is used to promote global convergence, and when an adaptive barrier update rule is used, the penalty parameter associated with the merit function is reset at every iteration.

In our experiments with IPOPT, we go a step further and disable the line search within each iteration and always accept the full fraction-to-the-boundary step with step sizes from (2.8). In this way, we can examine the performance of pure primal-dual steps generated with the adaptive barrier schemes.

The barrier parameter strategies tested in our first set of experiments are as follows:

- LOQO *rule*. The barrier parameter is chosen by (3.1) and (3.6).
- *Mehrotra probing*. At every iteration, the barrier parameter μ is given by (3.1) and (3.5). Since this requires the computation of the affine scaling step (3.2), this strategy is more expensive than the LOQO rule. For KNITRO-DIRECT, in the iterations in which the safeguarding trust region algorithm is invoked (e.g., when the reduced Hessian is not positive definite), the barrier parameter is computed by the LOQO rule instead of Mehrotra probing. This is done because Mehrotra probing is expensive to implement in the trust region algorithm, which uses a conjugate gradient iteration.



(a) KNITRO (b) IPOPT

FIG. 1. *Results for four barrier parameter updating strategies.*

- *MPC.* The complete Mehrotra predictor-corrector algorithm as described later in section 7. As in the Mehrotra probing rule, when KNITRO-DIRECT falls back on the safeguarded trust region algorithm, the barrier parameter is computed using the LOQO rule for efficiency, and no corrector step is used.
- *Monotone.* (Also known as the Fiacco–McCormick approach.) The barrier parameter is fixed, and a series of primal-dual steps is computed, until the optimality conditions for the barrier problem are satisfied to some accuracy. At this point the barrier parameter is decreased. IPOPT and KNITRO implement somewhat different variations of this monotone approach; see [27, 29] for details about the initial value of μ, the rule for decreasing μ, and the form of the barrier stop tests.

For the numerical comparison, we select all the nonlinear programming problems in the CUTEr test set from January 2005 that contain at least one general inequality or bound constraint. We exclude those problems that seem infeasible, unbounded, or are given with initial points at which the model functions cannot be evaluated; see [27]. This gives a total of 599 problems. For all scalable models we use default sizes. Figure 1 reports the number of function evaluations for IPOPT and KNITRO, comparing the performance of the four barrier strategies. All the plots in the paper use the logarithmic performance profiles proposed by Dolan and Moré [10]. To account for the fact that different local solutions might be computed, problems with significantly different final objective function values for successful runs were excluded (for example, for the results in Figure 1, 37 problems we excluded for IPOPT, and 54 for KNITRO).

The results given in Figure 1 indicate that the adaptive strategies outperform the monotone variant, and in particular that Mehrotra probing appears to be the most successful in terms of function evaluations, both in the KNITRO experiment (using steplength control) and in the IPOPT experiment (with no steplength control). Furthermore, the results obtained with IPOPT show that the quality of the pure (unglobalized) steps is good enough to promote convergence in most problems. This observation suggests that the globalization scheme that we propose in section 5 should interfere minimally with the iteration; it should be active only when the algorithms appear to be making no progress.

We also note from Figure 1 that the complete MPC algorithm is very fast on some problems, but is not sufficiently robust. The latter can be seen most clearly in

Figure 1(b) where full MPC steps are taken at every iteration. The reason for the lack of robustness of the MPC strategy will be discussed in section 7, together with a globalization safeguarding procedure.

4. Quality functions. The Mehrotra and LOQO rules rely on the heuristic parameters (3.5) and (3.6). We now consider an approach in which μ is selected using a clear-cut objective, formulated in terms of a *quality function* to be minimized. As before, we assume that $\mu = \sigma \frac{x^T z}{n}$, where the centering parameter $\sigma \geq 0$ is to be determined, and define $\Delta(\sigma)$ to be the solution of the primal-dual equations (2.5) as a function of σ. We also let $\alpha_x^{\max}(\sigma), \alpha_z^{\max}(\sigma)$ denote the steplengths satisfying the fraction to the boundary rule (2.8) for the step $\Delta = \Delta(\sigma)$, and we define the *probing points*

$$x(\sigma) = x + \alpha_x^{\max}(\sigma)\Delta x(\sigma),$$

$$y(\sigma) = y + \alpha_z^{\max}(\sigma)\Delta y(\sigma), \quad z(\sigma) = z + \alpha_z^{\max}(\sigma)\Delta z(\sigma).$$

Our goal is to choose the value of σ that provides significant improvement toward the solution of the nonlinear program (2.1). For example, we could choose σ so as to minimize the following nonlinear quality function based on the KKT error:

$$\begin{aligned} q_N(\sigma) = &\left\|\nabla f(x(\sigma)) - A(x(\sigma))^T y(\sigma) - z(\sigma)\right\|^2 + \|c(x(\sigma))\|^2 \\ &+ \|Z(\sigma)X(\sigma)e\|^2. \end{aligned} \tag{4.1}$$

The evaluation of q_N is, however, expensive since it requires the evaluation of the problem functions and derivatives for every value of σ. We can avoid this expense by using a *linear quality function*. If we assume that f and c are linear functions, we have that (4.1) can be expressed as

$$\begin{aligned} q_L(\sigma) = &(1 - \alpha_z^{\max}(\sigma))^2 \left\|\nabla f(x) - A(x)^T y - z\right\|^2 + (1 - \alpha_x^{\max}(\sigma))^2 \|c(x)\|^2 \\ &+ \|(X + \alpha_x^{\max}(\sigma)\Delta X(\sigma))(Z + \alpha_z^{\max}(\sigma)\Delta Z(\sigma))e\|^2, \end{aligned} \tag{4.2}$$

where $\Delta X(\sigma)$ is the diagonal matrix with $\Delta x(\sigma)$ on the diagonal, and similarly for $\Delta Z(\sigma)$. We point out that by design the function q_L measures the KKT error exactly at the probing points $(x(\sigma), y(\sigma), z(\sigma))$ for linear programming problems.

Note that $\Delta(\sigma) = \Delta(0) + \sigma(\Delta(1) - \Delta(0))$. Therefore, $\Delta(\sigma)$ can be computed easily for any value of σ once the linear system (2.5) has been solved twice to obtain $\Delta(0)$ and $\Delta(1)$. Having computed $\Delta(\sigma)$, the dominant cost in the evaluation of q_L lies in the computation of the maximal steplengths $\alpha_x^{\max}(\sigma), \alpha_z^{\max}(\sigma)$ and the last term in (4.2), which requires a few vector operations.

We have defined the quality function q_L using squared norms to severely penalize any large components in the KKT error. Note that $q_L(\sigma)$ is not a convex function of σ, in general. Moreover, due to the complicated dependence of the steplengths $\alpha_x^{\max}(\sigma), \alpha_z^{\max}(\sigma)$ on the parameter σ, it does not seem possible to obtain an analytic expression for the minimizers of q_L. Nevertheless, we have observed that, in practice, this function is usually unimodal.

Therefore, we implement a one-dimensional search scheme to compute an approximate minimizer of q_L. It uses a golden trisection procedure (see, e.g., [21]), and ignores the fact that q_L may not necessarily be unimodal. We first choose $\sigma^{\min}$ and $\sigma^{\max}$, which define a minimum and maximum limit on the σ value, and

define the two intervals $[\sigma^{\min}, 1]$ and $[1, \sigma^{\max}]$. In our implementation, the value $\sigma^{\min} = \max(\gamma, \mu^{\min} n/x^T z)$, where $\mu^{\min}$ ($= 10^{-9}$ in our implementation) defines a minimal permissible value of the barrier parameter, γ is some small number (say, 10^{-6} or 10^{-8}), and $\sigma^{\max} = 1000$. We first evaluate the quality function for $\sigma = 1$ and for some σ value slightly less than 1 (say 0.99). If $q_L(0.99) \leq q_L(1)$, then we perform our golden trisection procedure in the interval $[\sigma^{\min}, 1]$, otherwise we search in the interval $[1, \sigma^{\max}]$. (It is important that σ be allowed to take on values greater than one so that the algorithm can recover from overly aggressive reductions of the barrier parameter.) Our trisection procedure terminates if either 12 evaluations of the quality functions are performed, or if the search interval $[a, b]$ becomes smaller than $b \times 10^{-2}$.

The expected advantages of the quality function approach are twofold. First, we have defined a procedure that ties the choice of the barrier parameter to a measurable and achievable decrease in the (linearized) KKT error. Therefore, we expect this approach to converge in fewer iterations compared with previously proposed approaches based on heuristic formulas. Second, our choice of the barrier parameter takes into account the fraction to the boundary steplengths (2.8) (the functions (4.1) and (4.2) are based on the steps *after* applying the fraction to the boundary rule). Thus the implicit constraints (2.4) are taken into account in choosing μ. This is similar to the Mehrotra update formulas and should discourage choices of the barrier parameter that generate steps which quickly violate the bounds (2.4) and need to be truncated.

More implementation details are given in section 6. Before presenting our numerical results with the quality function, we study how to guarantee the global convergence of nonlinear interior methods that choose the barrier parameter adaptively.

5. A globalization framework. The adaptive strategies described in section 3 can be seen from the numerical results in that section to be quite robust, even without a rigorous globalization scheme. (We show in the next section this is also the case with the quality function approach.) Yet, since the barrier parameter is allowed to change at every iteration in these algorithms, there is no mechanism that indeed enforces global convergence of the iterates in all cases. In contrast, the monotone barrier strategy employed in the Fiacco–McCormick approach allows us to establish global convergence results by combining two mechanisms. First, the algorithms that minimize a given barrier problem (2.2) use a line search or trust region to enforce a decrease in a merit function (as in KNITRO) or to guarantee acceptability by a filter (as in IPOPT). This ensures that an optimality test for the barrier function is eventually satisfied to some tolerance ϵ. Second, by repeating this minimization process for decreasing values of μ and ϵ that converge to zero, one can establish global convergence results [4, 12] to stationary points of the nonlinear programming problem (2.1).

We now propose two globalization frameworks that monitor the performance of the iterations in reference to a mechanism that enforces global convergence. As long as the adaptive primal-dual steps make sufficient progress towards the solution, the algorithm is free to choose a new value for the barrier parameter at every iteration; here, the barrier parameter can be chosen by *any* desired rule. We call this the *free mode*. However, if the iteration fails to maintain progress, then the algorithm reverts to a *monotone mode*, in which a Fiacco–McCormick strategy is applied. Here, the value of the barrier parameter remains fixed, and a robust globalization technique (e.g., based on a merit function or a filter) is employed to ensure progress for the corresponding barrier problem. Once the barrier problem is approximately minimized,

the barrier parameter is decreased. The monotone mode continues until an iterate is generated that makes sufficient progress for the original problem, at which point the free mode resumes.

We stress that also in the free mode we might want to choose steplengths α_p, α_d that are shorter than the maximal step sizes $\alpha_x^{\max}, \alpha_z^{\max}$, in order to promote convergence to minimizers. In our implementations, we make sure that the steps have descent properties with respect to the barrier problem (2.2) corresponding to the current value of μ, and we perform a line search to enforce progress in a merit function or a filter (without history), both of which are defined with respect to this barrier problem. In this way, we force the algorithm to consider the objective function when determining a new trial point, and not only the norm of the optimality conditions, so that convergence to stationary points that are not minimizers is less likely.

There are various ways to measure whether steps in the free mode make sustained progress toward the solution of the nonlinear program (2.1). We have developed two mechanisms, one based on a measure of KKT error, and the other using a filter based on the value of the objective (2.1a) and a measure of the constraint violation. Both aim to interfere with adaptive steps as little as possible so as not to slow down convergence.

5.1. Nonmonotone decrease of the KKT error. In our first globalization framework, we monitor the KKT error of the original nonlinear program,

$$\Phi(x, y, z) = \left\|\nabla f(x) - A(x)^T y - z\right\|^2 + \|c(x)\|^2 + \|ZXe\|^2. \tag{5.1}$$

We require that this measure be reduced by a factor of $\kappa \in (0, 1)$ over at most a fixed number $l^{\max}$ of iterations, when the algorithm is in the free mode.

Algorithm A: KKT-Error-Based Globalization Framework

Given (x_0, y_0, z_0) with $(x_0, z_0) > 0$, a constant $\kappa \in (0, 1)$ and an integer $l^{\max} \geq 0$.
Set $k \leftarrow 0$.
Repeat
  Choose a target value of the barrier parameter μ_k, based on any rule.
  Compute the primal dual search direction Δ from (2.5).
  Determine step sizes $\alpha_p \in (0, \alpha_x^{\max}]$ and $\alpha_d \in (0, \alpha_z^{\max}]$.
  Compute the new trial iterate $(\tilde{x}_{k+1}, \tilde{y}_{k+1}, \tilde{z}_{k+1})$ from (2.7).
  Compute the KKT error $\tilde{\Phi}_{k+1} \equiv \Phi(\tilde{x}_{k+1}, \tilde{y}_{k+1}, \tilde{z}_{k+1})$.
  Set $M_k = \max\{\Phi_{k-l}, \Phi_{k-l+1}, \ldots, \Phi_k\}$ with $l = \min\{k, l^{\max}\}$.
  If $\tilde{\Phi}_{k+1} \leq \kappa M_k$
    Accept $(\tilde{x}_{k+1}, \tilde{y}_{k+1}, \tilde{z}_{k+1})$ as the new iterate, and set $\Phi_{k+1} \leftarrow \tilde{\Phi}_{k+1}$.
    Set $k \leftarrow k + 1$ and return to the beginning of the loop.
  else
    Start Monotone Mode:
    Starting from $(\tilde{x}_{k+1}, \tilde{y}_{k+1}, \tilde{z}_{k+1})$, and for an initial value $\bar{\mu}$, solve a sequence of barrier problems with a monotonically decreasing sequence of barrier parameters to obtain a new iterate $(x_{k+1}, y_{k+1}, z_{k+1})$ such that
      $\Phi_{k+1} \equiv \Phi(x_{k+1}, y_{k+1}, z_{k+1}) \leq \kappa M_k$.
    Set $k \leftarrow k + 1$ and resume the free mode at the beginning of the loop.
  end if
End (repeat).

In the monotone mode, it is not required to solve each barrier problem to the specified tolerance before checking whether the method can revert to the free mode. Instead, we compute the optimality error $\Phi(x, y, z)$ for all intermediate iterates in the monotone mode, and return to the free mode, as soon as $\Phi(x, y, z) \leq \kappa M_k$.

Typical values for the algorithmic parameters are $\kappa = 0.9999$ and $l^{\max} = 5$. An important issue when switching to the monotone mode is the initialization of the barrier parameter $\bar{\mu}$. This can be chosen, for example, to be some fraction of the current complementarity value. The rule used in our implementations is $\bar{\mu} = 0.8(x_k^T z_k)/n$.

5.2. Two-dimensional filter. In the second globalization method, we make use of a filter that accepts a trial point if it provides sufficient progress in terms of the constraint violation $\theta(x) = \|c(x)\|$ *or* the objective function $f(x)$, compared to the previous iterates generated in the free mode. We let $\mathcal{F}_k \subseteq \{(f, \theta) \in \mathbb{R}^2 : \theta \geq 0\}$ denote the (f, θ) pairs that are not acceptable at the current iteration k. The concept of acceptability by the filter is made precise below.

Algorithm B: Filter-Based Globalization Framework

Given (x_0, y_0, z_0) with $(x_0, z_0) > 0$, and constants $\kappa_1, \kappa_2 > 0$; initialize the filter $\mathcal{F}_0 = \emptyset$.

Set $k \leftarrow 0$.

Repeat

Choose a target value of the barrier parameter μ_k, based on any rule.
Compute the primal dual search direction Δ from (2.5).
Determine step sizes $\alpha_p \in (0, \alpha_x^{\max}]$ and $\alpha_d \in (0, \alpha_z^{\max}]$.
Compute the new trial iterate $(\tilde{x}_{k+1}, \tilde{y}_{k+1}, \tilde{z}_{k+1})$ from (2.7).
Compute the filter margin $\delta_k = \kappa_1 \min\{\kappa_2, \Phi(x_k, y_k, z_k)\}$.
If $(f(\tilde{x}_{k+1}) + \delta_k, \|c(\tilde{x}_{k+1})\| + \delta_k) \notin \mathcal{F}_k$
Accept $(\tilde{x}_{k+1}, \tilde{y}_{k+1}, \tilde{z}_{k+1})$ as the new iterate.
Update the filter $\mathcal{F}_{k+1} = \mathcal{F}_k \cup \{(f, \theta) : f \geq f(\tilde{x}_{k+1}) \text{ and } \theta \geq \|c(\tilde{x}_{k+1})\|\}$.
Set $k \leftarrow k + 1$ and return to the beginning of the loop.
else
Start Monotone Mode:
Starting from $(\tilde{x}_{k+1}, \tilde{y}_{k+1}, \tilde{z}_{k+1})$, and for an initial value $\bar{\mu}$, solve a sequence of barrier problems with a monotonically decreasing sequence of barrier parameters to obtain a new iterate $(x_{k+1}, y_{k+1}, z_{k+1})$ such that
$$(f(x_{k+1}) + \delta_k, \|c(x_{k+1})\| + \delta_k) \notin \mathcal{F}_k.$$
Augment the filter:
$$\mathcal{F}_{k+1} = \mathcal{F}_k \cup \{(f, \theta) : f \geq f(x_{k+1}) \text{ and } \theta \geq \|c(x_{k+1})\|\}.$$
Set $k \leftarrow k + 1$ and resume the free mode at the beginning of the loop.
end if

End (repeat).

Similar to the KKT-error based globalization framework, the monotone mode is terminated as soon as an iterate is encountered that is acceptable to the filter.

We have tested the KKT and filter globalization approaches using IPOPT and KNITRO, and found both to be effective in practice. For the sake of brevity, we report results only for the filter globalization framework in the next section. We set $\kappa_1 = 10^{-5}$ and $\kappa_2 = 1$ for these tests, and we choose $\bar{\mu} = 0.8(x_k^T z_k)/n$ for the barrier parameter when entering the monotone mode.

5.3. Global convergence results. In the following we summarize the theoretical convergence guarantees for the two globalization frameworks presented above.

THEOREM 5.1. *Let $\{(x_k, y_k, z_k)\}$ be the sequence generated by either Algorithm A or Algorithm B, and assume that the monotone mode always terminates successfully. For Algorithm B, further assume that $\{f(x_k)\}$ is bounded below and that $\{\|c(x_k)\|\}$ is bounded above. Then, the KKT error $\Phi(x_k, y_k, z_k)$ converges to zero.*

Proof. Algorithm A: Since this framework ensures that the optimality measure $\Phi_k = \Phi(x_k, y_k z_k)$ is reduced by a factor of $\kappa \in (0, 1)$ in at most every $l^{\max}$ iterations, it is clear that $\Phi_k \to 0$.

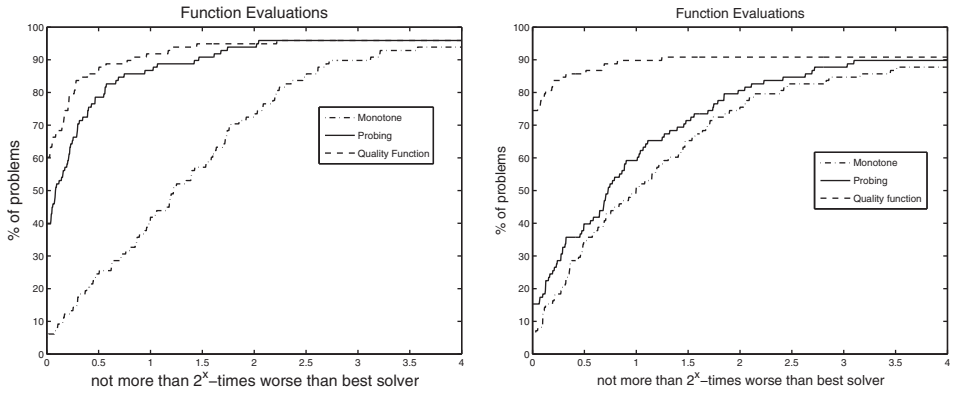
Algorithm B: The proof is by contradiction and is similar to the proof of Lemma 3.3 in [13]. Suppose that there is a subsequence $\{k_j\}$ of iterations in which $\delta_{k_j-1} \geq \epsilon > 0$. Then $f(x_{k_j})$ has to be bounded above, say by f_U, since otherwise we could find a subsequence $\{k_{j_l}\}$ of $\{k_j\}$ with $f(x_{k_{j_l}}) \leq f(x_{k_{j_{l+1}}})$ and $f(x_{k_{j_l}}) \to \infty$ so that the filter update rule would yield $\|c(x_{k_{j_{l+1}}})\| < \|c(x_{k_{j_l}})\| - \delta_{k_{j_l}-1} \leq \|c(x_{k_{j_l}})\| - \epsilon \to -\infty$. Therefore, for each k_j, the area of the region $\mathcal{F}_{k_j} \setminus \mathcal{F}_{k_j-1}$ added to $\mathcal{F}_{k_j-1}$ includes a square of size $\delta^2_{k_j-1} \geq \epsilon^2$ within the set $\bar{\mathcal{F}} = \{(f, \theta) : f_L \leq f \leq f_U \text{ and } 0 \leq \theta \leq \theta_U\}$. Here, f_L denotes a lower bound of $\{f(x_k)\}$ and θ_U an upper bound of $\{\|c(x_k)\|\}$, which exist by assumption. Because of the monotonicity $\mathcal{F}_k \subseteq \mathcal{F}_{k+1}$ of the filter, this leads to a contradiction, since $\bar{\mathcal{F}}$ is finite. ∎

The above result pertains to the cases where the algorithm does not eventually stay in the monotone mode. If it does, the iteration inherits the convergence properties from the underlying Fiacco–McCormick algorithm. In particular, if the nonlinear program (2.1) is infeasible the KKT error cannot converge to zero, and therefore Theorem 5.1 shows that both algorithms must eventually remain in the monotone mode. In that mode, there will be a value of the barrier parameter, say $\bar{\mu}$, for which the corresponding barrier problem is infeasible and cannot be solved to the required convergence tolerance. For KNITRO, it has been shown that the algorithm then generates an infeasible limit point that is a stationary point for the ℓ_2-norm of the constraint violation [4]. For IPOPT, the filter line-search algorithm for that barrier problem will eventually stay in the restoration phase [26]; the current implementation of the restoration phase then minimizes the ℓ_1-norm of the constraint violation. Therefore, if the nonlinear program is infeasible, both algorithms will generate a message indicating that the problem is locally infeasible.

6. Numerical results. We first discuss the choice of norms in the quality function (4.2) and in the optimality measure (5.1). (For consistency, we use the same norms and scaling factors for the individual terms in (4.2) and (5.1).) In IPOPT we use the 2-norm, and each of the three terms is divided by the number of elements in the vectors whose norms are being computed. In KNITRO, we choose the norm and scaling factors to be similar to the terms used in the KNITRO termination test: The first two terms in (4.2) and (5.1) use the infinity-norm, the complementarity term uses the 1-norm divided by n, and we scale these terms using the factors described in [29].

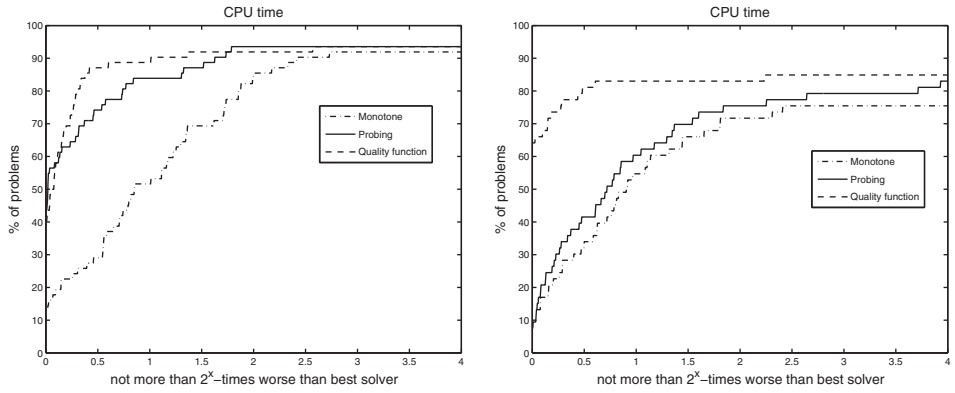
The tests involving IPOPT were run on a Dual-Pentium III, 1GHz machine running Linux. The KNITRO tests were run on a machine with an AMD Athlon XP 3200+ 2.2GHz processor running Linux. For both codes, the maximum number of iterations was set to 3000 and the time limit was set to 1800 CPU seconds. The tests were run using the development versions of IPOPT and KNITRO as of October 2005.

The first results we present are for the linear programming problems in the NETLIB collection, as specified in the CUTEr test set [18]. No preprocessing was



(a) IPOPT (b) KNITRO

FIG. 2. *Function evaluation comparison for the NETLIB test set.*

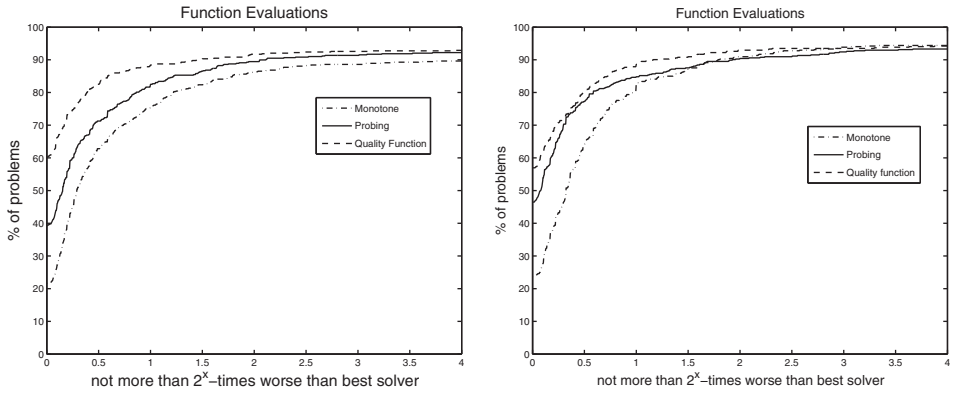


(a) IPOPT (b) KNITRO

FIG. 3. *CPU time comparison for the NETLIB test set.*

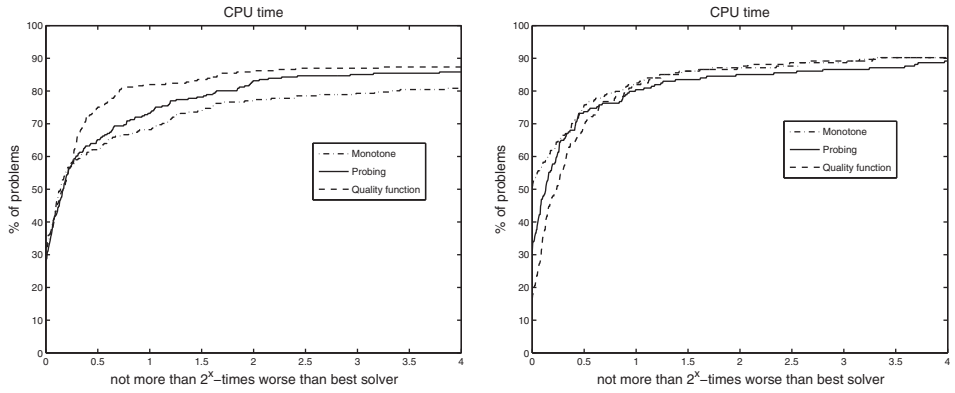
performed, and no initial point strategy was employed (i.e., the default starting point $x_0 = (0, \ldots, 0)$ was used). Figure 2 compares the performance of the quality function approach, in terms of function evaluation count, with two of the strategies described in section 3, namely, the monotone method and the Mehrotra probing heuristic. Figure 3 compares the algorithms in terms of CPU time. Since we are primarily interested in methods with globally convergent frameworks we use the filter based globalization framework described in section 5 for both the Mehrotra probing approach and the quality function approach. The monotone approach is globally convergent on its own. Even though our focus is on nonlinear optimization, linear programming problems are of interest since they allow us to assess the effectiveness of the quality function in a context in which it exactly predicts the KKT error. It is apparent from Figure 2 that the quality function approach is very effective on the NETLIB test set.

The performance (in terms of function evaluations) of the three barrier update strategies on nonlinear programming problems with at least one inequality or bound constraint from the CUTEr collection is reported in Figure 4. The quality function approach again performs significantly better than the monotone method; it also



(a) IPOPT (b) KNITRO

FIG. 4. *Function evaluation comparison for CUTEr test set.*



(a) IPOPT (b) KNITRO

FIG. 5. *CPU time comparison for the CUTEr test set.*

outperforms the Mehrotra probing strategy, which had given the best results in the experiments reported in section 3. The improvements are less pronounced when comparing CPU performance; see Figure 5.

To give more insight into the behavior of the quality function approach, we present in Table 1 data about the value of the centering parameter σ_k chosen by the quality function approach, together with statistics about the globalization strategy employed. The data was collected from the results produced by IPOPT. We compare the probing and quality function approaches. The third column gives the percentage of iterations spent in the monotone mode. The rest of the columns report the percentage of iterations in which σ_k lie in the intervals $[\sigma^{\max}, 10]$, $(10, 1]$, $(1, 10^{-1}]$, etc. The percentage numbers in Table 1 were obtained by computing the average of the percentages for each successfully solved problem.

As we can see, only a small percentage of iterations is spent in the monotone mode, showing that the adaptive mode is the main driving force. Note also that the quality function strategy tends to produce larger values of σ_k than the probing approach.

TABLE 1
Average percentage of iterations in monotone and free mode, producing σ_k in certain ranges.

Method	Testset	%mono	free mode, with σ_k in range							
		mode	≥ 10	≥ 1	$\geq 10^{-1}$	$\geq 10^{-2}$	$\geq 10^{-3}$	$\geq 10^{-4}$	$\geq 10^{-5}$	$< 10^{-5}$
Probing	NETLIB	4.77%	0.00	2.15	72.35	10.18	3.29	1.33	1.44	4.50
Qual. fctn.	NETLIB	3.70%	4.90	11.39	53.41	14.69	3.56	0.63	0.65	7.07
Probing	CUTEr	6.92%	0.49	2.25	26.59	19.99	12.08	5.90	4.27	21.50
Qual. fctn.	CUTEr	8.39%	4.53	6.98	32.89	22.50	10.57	2.08	2.46	9.59

The performance profiles in Figures 4 and 5 indicate that there are a number of problems that cannot be solved by the quality function approach. We now make some observations about the behavior of the two codes on these problems.

Considering both the NETLIB and CUTEr test sets, there were 38 failures for the KNITRO implementation of the globalized quality function adaptive barrier rule. Of these, 18 problems were solved when the default time or iteration limits were increased; 5 problems (BRANDY, PALMER2, PALMER7A, SAWPATH, SINEALI) terminated at near optimal approximate solutions, but KNITRO could not get enough accuracy in the dual feasibility measure; 2 problems (CRESC132, QCNEW) terminated because of evaluation errors resulting from IEEE exceptions (`NaN`) in the function evaluation; and 1 problem (HS110) terminated with a message of unboundedness at a feasible point with a very large negative value of the objective function. The optimal objective for HS110 is $-9.960\text{e}+39$,[1] whereas by default KNITRO declares unboundedness for a feasible objective value less than the cutoff limit value $-1.0\text{e}+20$. When this limit is changed, KNITRO solves the problem in 4 iterations. In addition, the problem KTMODEL was discovered to have incorrect gradients which caused KNITRO to terminate at an infeasible point. The remaining 11 problems (COSHFUN, DITTERT, DRUGDISE, GREENBEA, GREENBEB, MANNE, NUFFIELD, PILOT-JA, TENBARS2, TENBARS3, ZIGZAG) constitute unresolved failures.

We give some more information about these 11 problems. GREENBEA, GREENBEB, PILOT-JA, NUFFIELD, and ZIGZAG were not solved even when the time limit was increased to three hours. The problems GREENBEA, GREENBEB, and PILOT-JA are linear programs for which KNITRO experiences numerical difficulties (from rank-deficient Jacobians) that cause it to often fallback on steepest descent like steps and converge slowly. KNITRO appears to be very close to the solution in PILOT-JA when it reaches the time limit. The problems COSHFUN, DRUGDISE, TENBARS2, and TENBARS3 were not solved even when the iteration limit was raised to 100,000 (although it appears in all cases that slow progress is still being made when the iteration limit is reached). For the problem DITTERT, KNITRO terminates at an infeasible point but the code is unable to verify whether or not it is an infeasible stationary point. Finally, for the problem MANNE, KNITRO terminated at a feasible point but with a large dual feasibility error.

For the IPOPT code, there were 39 failures for the globalized quality function adaptive barrier rule. Of these, 13 problems were solved if more iterations (100,000) or CPU time (3 hours) were allowed. In 6 problems (A2NNDNIL, A5NNDNIL, CRESC100, EG3, POLAK3, SPIRAL), IPOPT terminated at a point satisfying the local infeasibility criterion; in 2 problems (A2NSDSIL, BRAINPC9) it failed during the restoration phase. For COSHFUN, the memory requirement in the linear solver was exceeded in iteration 10681, and the problems EQC and ROBOT terminated because the search

[1] This is true for problem size N=200 which was the default size for this SIF model at the time of testing.

direction became too small, but in both cases the problem was almost solved. For problem LIN, the objective function value at the (modified) starting point resulted in an IEEE exception (`NaN`).

In the remaining failures, the maximum iteration count or CPU time were exceeded. Problems AVION2, PALMER5E, YORKNET terminated after 100,000 iterations; IPOPT appeared to be cycling with small primal and dual feasibility errors for two of these problems (AVION2 and PALMER5E). For problem PALMER7E, IPOPT still made very small progress after 100,000 iterations, while for problem KT-MODEL the code seemed to diverge (as a result of incorrect gradient information). Finally, the time limit was exceeded for the linear program QAP15 from NETLIB, and for the CUTEr models A5NSDSIL, CRESC132, GAUSSELM, GLIDER, MANNE, NUFFIELD, ORTHREGE, READING8.

7. Corrector steps. The numerical results of section 3 indicate that, when solving nonlinear problems, including the corrector step in Mehrotra's method (the MPC method) is often not beneficial. This is in stark contrast with the experience in linear programming and convex quadratic programming, where the corrector step is known to accelerate the interior-point iteration without degrading its robustness. In this section we study the effect of the corrector step and find that it can also be harmful in the linear programming and quadratic programming cases *if* an initial point strategy is not used. These observations are relevant because in nonlinear programming it is much more difficult to find a good starting point.

Let us begin by considering the linear programming case. There are several ways of viewing the MPC method in this context. One is to consider the step computation as taking place in three stages (see, e.g., [30]). First, the algorithm computes the affine scaling step (3.2) and uses it to determine the target value of the barrier parameter $\mu = \sigma \frac{x^T z}{n}$, where σ is given by (3.5). Next, the algorithm computes a primal-dual step, say $\Delta^{\rm pd}$, from (2.5) using that value of μ. Finally, a corrector step $\Delta^{\rm corr}$ is computed by solving (2.5) with the right-hand side given by

$$-\left(0, \Delta X^{\rm aff} \Delta Z^{\rm aff} e, 0\right)^T, \tag{7.1}$$

where $\Delta X^{\rm aff}$ is the diagonal matrix with diagonal entries given by $\Delta x^{\rm aff}$, and similarly for $\Delta Z^{\rm aff}$. The complete MPC step is the sum of the primal-dual and corrector steps. We can compute it by adding the right-hand sides and solving the following system:

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L} & -A^T(x) & -I \\ Z & 0 & X \\ A(x) & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x^{\rm mpc} \\ \Delta y^{\rm mpc} \\ \Delta z^{\rm mpc} \end{bmatrix} = - \begin{bmatrix} \nabla f(x) - A^T(x) y - z \\ Xz - \mu e + \Delta X^{\rm aff} \Delta Z^{\rm aff} e \\ c(x) \end{bmatrix}. \tag{7.2}$$

The new iterate (x^+, y^+, z^+) of the MPC method is given by (2.7)–(2.8) with $\Delta = (\Delta x^{\rm mpc}, \Delta y^{\rm mpc}, \Delta z^{\rm mpc})$.

Alternative views of the MPC method are possible by the linearity of the step computation: We can group the right-hand side in (7.2) in different ways and thereby interpret the step as the sum of different components. Yet all these views point out the following inconsistency in the MPC approach.

In the linear programming case, primal and dual feasibility are linear functions and hence vanish at the full affine scaling point, defined by

$$(x, y, z) + \left(\Delta x^{\rm aff}, \Delta y^{\rm aff}, \Delta z^{\rm aff}\right). \tag{7.3}$$

The complementarity term takes on the value

$$\left(X + \Delta X^{\rm aff}\right)\left(Z + \Delta Z^{\rm aff}\right) = \Delta X^{\rm aff} \Delta Z^{\rm aff}.$$

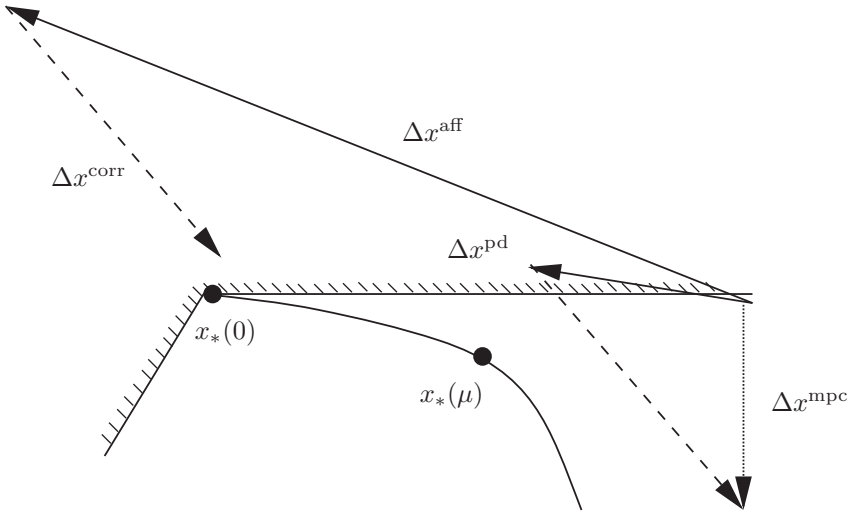


FIG. 6. *An unfavorable corrector step.*

TABLE 2
Output for NETLIB problem FORPLAN *for default PCx with bad starting point.*

Iter	Primal Obj	Dual Obj	PriInf	DualInf	$\alpha_x^{\max}$	$\alpha_z^{\max}$	$\log\left(\frac{x^T z}{n}\right)$	$\|\Delta^{\text{aff}}\|$	$\|\Delta^{\text{mpc}}\|$
0	9.0515e + 01	−4.8813e + 06	1.0e − 00	1.9e + 00	0.0e + 00	0.0e + 00	0.00	0.0e + 00	0.0e + 00
1	9.0216e + 01	−1.3664e + 08	1.0e − 00	1.9e + 00	8.6e − 13	5.0e − 12	0.08	5.6e + 06	1.2e + 13
2	9.0403e + 01	−3.3916e + 08	1.0e − 00	1.9e + 00	7.3e − 13	4.8e − 13	0.16	9.1e + 07	1.9e + 14
3	9.0769e + 01	−1.1343e + 10	1.0e − 00	1.9e + 00	4.0e − 12	1.2e − 11	1.18	2.2e + 08	3.9e + 14
4	9.0860e + 01	−1.8010e + 11	1.0e − 00	1.9e + 00	1.5e − 12	5.0e − 12	2.35	8.0e + 09	1.4e + 16
5	9.1312e + 01	−2.9307e + 12	1.0e − 00	1.9e + 00	4.3e − 12	5.1e − 12	3.56	1.3e + 11	2.2e + 17
6	9.1710e + 01	−8.2787e + 13	1.0e − 00	1.9e + 00	6.0e − 12	9.1e − 12	5.01	2.1e + 12	3.6e + 18
7	9.2036e + 01	−1.5505e + 15	1.0e − 00	1.9e + 00	7.5e − 12	6.0e − 12	6.28	5.9e + 13	1.0e + 20
8	9.2282e + 01	−6.8149e + 16	1.0e − 00	1.9e + 00	7.0e − 12	1.4e − 11	7.93	1.1e + 15	1.9e + 21
9	9.2279e + 01	−4.4155e + 18	1.0e − 00	1.9e + 00	9.2e − 12	2.1e − 11	9.74	4.8e + 16	8.3e + 22
10	9.2244e + 01	−2.8697e + 20	1.0e − 00	1.9e + 00	6.8e − 12	2.1e − 11	11.55	3.1e + 18	5.4e + 24
11	9.2381e + 01	−3.1118e + 22	1.0e − 00	1.9e + 00	1.1e − 11	3.6e − 11	13.58	2.0e + 20	3.5e + 26
12	9.2462e + 01	−7.0471e + 24	1.0e − 00	2.2e + 01	6.2e − 12	7.6e − 11	15.94	2.2e + 22	3.8e + 28
13	9.2523e + 01	−9.9820e + 26	1.0e − 00	2.8e + 03	1.4e − 11	4.7e − 11	18.09	5.0e + 24	8.6e + 30
14	9.2605e + 01	−1.1959e + 30	1.0e − 00	2.2e + 01	2.1e − 11	4.0e − 10	21.17	7.1e + 26	1.2e + 33

Therefore the value of the right-hand side vector in (2.5) at the full affine scaling step (7.3) is given by (7.1). Thus the corrector step can be viewed as a modified Newton step taken from the point (7.3) and using the primal-dual matrix evaluated at the current iterate (x, y, z).

The inconsistency in the MPC approach arises because the corrector step, which is designed to improve the full affine scaling step, is applied at the primal-dual point; see Figure 6. In some circumstances, this mismatch can cause poor steps. In particular, we have observed that if the affine scaling step is very long, in the sense that the steplengths (2.8) are very small, and if the corrector step is even larger, then the addition of the corrector step to the primal-dual step (2.7) can significantly increase the complementarity value $x^T z$. This behavior can be sustained and lead to very slow convergence or failure, as shown in Table 2. The results in this table were obtained using PCx [9], an interior-point code for linear programming that implements the MPC method, applied to problem FORPLAN from the NETLIB collection. Practical implementations of the MPC use a procedure for choosing a favorable starting point

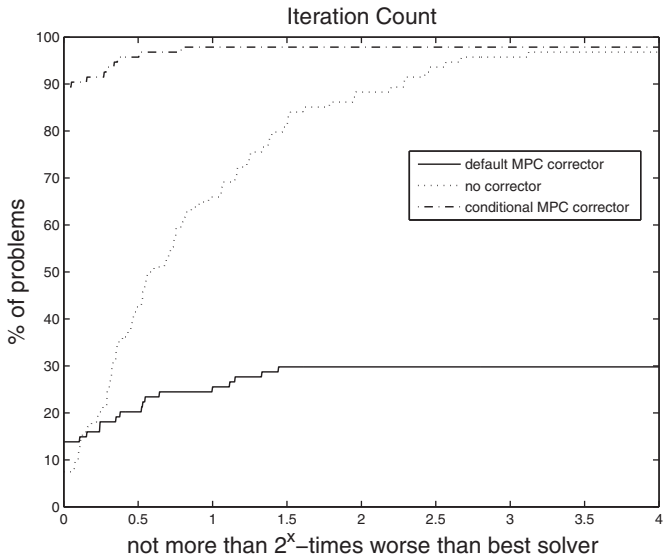


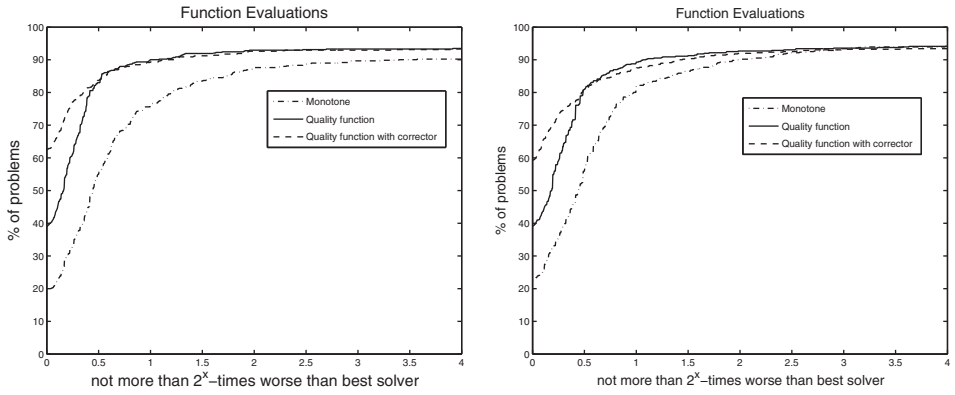
FIG. 7. *Results on the NETLIB test set for three corrector step strategies implemented in PCx. The initial point was set to* $x = e, z = e$.

described by Mehrotra [22]. We disabled this initial point strategy for PCx and set the initial point to $x = e, z = e$. Note from Table 2 that the affine scaling and corrector steps appear to grow without bound, and examination of the results shows that the dual variables diverge.

To provide further support to the claim that the corrector step can be harmful, we ran the complete set of test problems (94 in all) in the NETLIB collection. Using the default settings, which includes a strategy for computing a good starting point, PCx solved 90 problems, and terminated very close to the solution in the remaining 4 cases. Next we disabled the initial point strategy and set the initial point to $x = e, z = e$. PCx was now able to solve only 28 problems (and in only 3 additional cases terminated very close to the solution).

We repeated the experiment, using the initial point $x = e, z = e$, but this time removing the corrector step; this corresponds to the algorithm called *Mehrotra probing* in section 3. We also tested a variant that we call *conditional MPC* in which the corrector step is employed in the MPC method only if it does not result in an increase of complementarity by a factor larger than 2. The results, in terms of iterations, are reported in Figure 7. Note the dramatic increase in robustness of both strategies, compared with the MPC algorithm. The conditional MPC strategy is motivated by the observation that harmful effects of the corrector steps manifest themselves in a significant increase in complementarity. The failure of convergence of the MPC method has also been analyzed by Cartis [7].

Finally we compare the monotone and quality function approaches described in section 3 with the conditional MPC approach on the nonlinear programming problems used in that section. The conditional MPC method is now implemented so as to reject corrector steps that increase complementarity (this more conservative approach appears to be more suitable in the nonlinear case). Furthermore, if the conditional MPC step does not pass the merit function or filter acceptance test for the current barrier problem, the corrector step is also rejected, and the backtracking line search



(a) IPOPT (b) KNITRO

FIG. 8. *Results for safeguarded corrector steps.*

for the regular primal-dual step is executed. Finally, no corrector step is computed while the algorithm is in the monotone mode. The results, given in Figure 8, indicate that this conditional MPC method requires fewer function evaluations, and is not less robust, than the other strategies.

8. Conclusion. We have seen in this paper that, both for linear and nonlinear programming, classical barrier update strategies with global convergence guarantees are overly conservative, while strategies that are often fastest in practice are based on heuristic formulas and are not globally convergent. We have presented a new update strategy and shown that it is efficient in practice. Instead of basing the update on a heuristic formula, our approach follows a clearly defined objective, namely, the minimization of a quality function. We further proposed a simple globalization framework that makes use of any nonmonotone barrier parameter strategy.

We have also presented results that show the corrector steps employed in MPC can have harmful effects, even in the linear and quadratic programming cases. These observations were unexpected since the MPC method has become widely used in linear and quadratic programming; our tests show that the reliability of the MPC method depends crucially on heuristics, such as the choice of the initial point. We have shown, however, that the selective use of corrector steps can have beneficial effects in interior point methods.

A question we have not addressed is whether the approach presented in this paper enjoys fast local convergence. We have not specifically introduced features that guarantee superlinear convergence. This could be done by using various techniques proposed in the literature for controlling the asymptotic behavior of the barrier parameter; see, e.g., [19] and the references therein. In particular, we could implement the strategies recently proposed by Armand et al. [1, 2] in conjunction with the quality function approach.

We have not done so because the asymptotic behavior of the method proposed in this paper has proved to be acceptable in practice. In fact, the quality function approach promotes fast local convergence because it chooses the barrier parameter so as to (approximately) minimize the quality function in the region defined by (2.8). One can design a superlinearly convergent algorithm by choosing the barrier parameter so that a step to the region defined by (2.8) decreases the KKT error superlinearly.

Since the quality function is an approximation to the KKT error, and it attempts to minimize it, it is not surprising that the quality function approach tends to yield fast local convergence.

Acknowledgments. We would like to thank Richard Byrd for many valuable suggestions during the course of this work, and the referees for suggesting ways of improving the paper.

REFERENCES

[1] P. Armand and J. Benoist, *A local convergence property of primal-dual methods for nonlinear programming,* Math. Program., Ser. A, 115 (2008), pp. 199–222.

[2] P. Armand, J. Benoist, and D. Orban, *Dynamic updates of the barrier parameter in primal-dual methods for for nonlinear programming,* Comput. Optim. Appl., 41 (2008), pp. 1–25.

[3] J. Betts, S. K. Eldersveld, P. D. Frank, and J. G. Lewis, *An interior-point nonlinear programming algorithm for large scale optimization,* in Large-Scale PDE-Constrained Optimization, O. Ghattas, M. Heinkenschloss, D. Keyes, L. T. Biegler, and B. van Bloemen Waanders, eds., Lecture Notes Comput. Sci. Eng., Springer Verlag, 2003, pp. 184–198.

[4] R. H. Byrd, J.-Ch. Gilbert, and J. Nocedal, *A trust region method based on interior point techniques for nonlinear programming,* Math. Program., 89 (2000), pp. 149–185.

[5] R. H. Byrd, M. E. Hribar, and J. Nocedal, *An interior point algorithm for large scale nonlinear programming,* SIAM J. Optim., 9 (1999), pp. 877–900.

[6] R. H. Byrd, J. Nocedal, and R. A. Waltz, *KNITRO: An integrated package for nonlinear optimization,* in Large-Scale Nonlinear Optimization, G. di Pillo and M. Roma, eds., Springer, Berlin, 2006, pp. 35–59.

[7] C. Cartis, *Some disadvantages of a Mehrotra-type primal-dual corector interior point algorithms for linear programming,* Appl. Numer. Math., to appear.

[8] L. Chen and D. Goldfarb, *Interior-point ℓ_2 penalty methods for nonlinear programming with strong global convergence properties,* Math. Program., 108 (2006), pp. 1–36.

[9] J. Czyzyk, S. Mehrotra, and S. J. Wright, *PCx User Guide,* Technical report, Argonne National Laboratory, Argonne, IL, 1996.

[10] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles,* Math. Program., Ser. A, 91 (2002), pp. 201–213.

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

[12] A. V. Fiacco and G. P. McCormick, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques.* J. Wiley and Sons, Chichester, England, 1968. Reprinted as *Classics in Applied Mathematics 4*, SIAM, Philadelphia, 1990.

[13] R. Fletcher, N. I. M. Gould, S. Leyffer, P. L. Toint, and A. Wächter, *Global convergence of a trust-region SQP-filter algorithms for general nonlinear programming,* SIAM J. Optim., 13 (2002), pp. 635–659.

[14] A. Forsgren and P. E. Gill, *Primal-dual interior methods for nonconvex nonlinear programming,* SIAM J. Optim., 8 (1998), pp. 1132–1152.

[15] D. M. Gay, M. L. Overton, and M. H. Wright, *A primal-dual interior method for nonconvex nonlinear programming,* in Advances in Nonlinear Programming (Beijing, 1996), Y. Yuan, ed., Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998, pp. 31–56,

[16] E. M. Gertz and P. E. Gill, *A primal-dual trust region algortihm for nonlinear programming,* Math. Program., 100 (2004), pp. 49–94.

[17] N. I. M. Gould, D. Orban, and P. Toint, *An interior-point L1-penalty method for nonlinear optimization,* Technical report RAL-TR-2003-022, Rutherford Appleton Laboratory Chilton, Oxfordshire, UK, 2003.

[18] N. I. M. Gould, D. Orban, and P. L. Toint, CUTEr *and* sifdec*: A constrained and unconstrained testing environment, revisited,* ACM Trans. Math. Software, 29 (2003), pp. 373–394.

[19] N. I. M. Gould, D. Orban, and P. L. Toint, *Numerical methods for large-scale nonlinear optimization,* Acta Numer., (2005), pp. 299–361.

[20] I. Griva, D. F. Shanno, and R. J. Vanderbei, *Convergence Analysis of a Primal-Dual Method for Nonlinear Programming,* Report, Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ, 2004.

[21] D. G. Luenberger, *Linear and Nonlinear Programming,* 2nd ed., Addison-Wesley Publishing Company, Reading, MA, 1984.
[22] S. Mehrotra, *On the implementation of a primal-dual interior point method,* SIAM J. Optim., 2 (1992), pp. 575–601.
[23] A. L. Tits, A. Wächter, S. Bakhtiari, T. J. Urban, and C. T. Lawrence, *A primal-dual interior-point method for nonlinear programming with strong global and local convergence properties,* SIAM J. Optim., 14 (2003), pp. 173–199.
[24] M. Ulbrich, S. Ulbrich, and L. Vicente, *A globally convergent primal-dual interior point filter method for nonconvex nonlinear programming,* Math. Program., 2 (2004), pp. 379–410.
[25] R. J. Vanderbei and D. F. Shanno, *An interior point algorithm for nonconvex nonlinear programming,* Comput. Optim. Appl., 13 (1999), pp. 231–252.
[26] A. Wächter and L. T. Biegler, *Line search filter methods for nonlinear programming: Motivation and global convergence,* SIAM J. Optim., 16 (2005), pp. 1–31.
[27] A. Wächter and L. T. Biegler, *On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming,* Math. Program., 106 (2006), pp. 25–57.
[28] R. A. Waltz, Knitro 4.0 *User's Manual,* Technical report, Ziena Optimization, Inc., Evanston, IL, USA, 2004.
[29] R. A. Waltz, J. L. Morales, J. Nocedal, and D. Orban, *An interior algorithm for nonlinear optimization that combines line search and trust region steps,* Math. Program., Ser. A, 107 (2006), pp. 391–408.
[30] S. Wright, *Primal-dual interior-point methods,* SIAM, Philadelphia, 1997.
[31] H. Yamashita, *A globally convergent primal-dual interior-point method for constrained optimization,* Optim. Methods Software, 10 (1998), pp. 443–469.
[32] H. Yamashita, H. Yabe, and T. Tanabe, *A globally and superlinearly convergent primal-dual interor point trust region method for large scale constrained optimization,* Math. Program., Ser. A, 102 (2005), pp. 111–120.