

INTERIOR-POINT METHODS FOR NONCONVEX NONLINEAR PROGRAMMING: FILTER METHODS AND MERIT FUNCTIONS

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

Operations Research and Financial Engineering
Princeton University
ORFE-00-06

Revised December 20, 2001

ABSTRACT. Recently, Fletcher and Leyffer proposed using filter methods instead of a merit function to control steplengths in a sequential quadratic programming algorithm. In this paper, we analyze possible ways to implement a filter-based approach in an interior-point algorithm. Extensive numerical testing shows that such an approach is more efficient than using a merit function alone.

1. INTRODUCTION

In three recent papers, [11], [8], and [6], the authors describe an infeasible, primal-dual, interior-point algorithm for nonconvex nonlinear programming and provide numerical results using the code LOQO, which implements the algorithm. With nonlinear programming, it is generally impossible to use a two-phased approach where one first finds a feasible solution and then moves toward optimality while preserving feasibility. Hence, one generally resorts to *infeasible methods* which

Date: December 20, 2001.

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

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

work simultaneously on feasibility and optimality. In designing infeasible methods, we are therefore confronted at each iteration with two competing and contradictory aims: reducing infeasibility and improving the objective value. Such trade-offs are common. For example, in finance one is reminded of Markowitz's notion of an efficient frontier which gives a trade-off between risk and reward. In statistics, there is the related notion of admissibility. In the realm of nonlinear optimization, this old idea has been introduced recently and has been coined *filter methods*. As introduced by Fletcher and Leyffer in [4], a filter is completely analogous to an efficient frontier. In this paper, we discuss how to implement filter-based methods in the context of our interior-point algorithm, which is introduced in the next section.

2. AN INFEASIBLE INTERIOR-POINT METHOD

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

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

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

Expressing nonlinear optimization problems in the form given by (1) is rather restrictive—the actual implementation of LOQO allows for much greater variety in the form of expression, including bounds on variables, ranges on inequalities, and equality constraints. In this paper, we focus on form (1) for purely pedagogical reasons. For further details on the general case, see [10].

Adding slacks, (1) becomes

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

where $h(x)$ and w are vectors representing the $h_i(x)$'s and w_i 's, respectively. The inequality constraints are then eliminated by incorporating them in a logarithmic barrier term in the objective function, $b_\mu(x, w) = f(x) - \mu \sum_{i=1}^m \log(w_i)$, transforming (2) to

$$(3) \quad \begin{aligned} & \text{minimize } b_\mu(x, w) \\ & \text{subject to } h(x) - w = 0. \end{aligned}$$

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

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

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

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

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

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

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

and

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

the Newton system for (5) is then

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

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

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

where

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

LOQO generally solves a modification of (6) in which λI is added to $H(x, y)$. For $\lambda = 0$, we have the Newton directions. As $\lambda \rightarrow \infty$, the directions approach the steepest descent directions. Choosing λ so that

$H(x, y) + A(x)^T W^{-1} Y A(x) + \lambda I$ is positive definite ensures that the step directions are descent directions. See [11] for further details.

Having computed step directions, Δx , Δw , and Δy , LOQO then proceeds to a new point by

$$(7) \quad \begin{aligned} x^{(k+1)} &= x^{(k)} + \alpha^{(k)} \Delta x^{(k)}, \\ w^{(k+1)} &= w^{(k)} + \alpha^{(k)} \Delta w^{(k)}, \\ y^{(k+1)} &= y^{(k)} + \alpha^{(k)} \Delta y^{(k)}, \end{aligned}$$

where $\alpha^{(k)}$ is chosen to ensure that $w^{(k+1)} > 0$, $y^{(k+1)} > 0$, and either the barrier function, b_μ , or the infeasibility, $\|\rho\|$, is reduced. To obtain such a reduction, we use a linear combination of these two quantities, that is, a merit function:

$$(8) \quad \Psi_{\beta, \mu}(x, w) = b_\mu(x, w) + \frac{\beta}{2} \|\rho(x, w)\|^2.$$

(Here, and throughout the paper, all norms are Euclidean, unless otherwise indicated). It is shown in [11] that, at each step, there exists a β such that the solution of (6) is a descent direction for this merit function.

Recently, Fletcher and Leyffer [4] studied solving problem (1) using a sequential-quadratic-programming algorithm. In their work, slack variables are not added to the problem, but rather the search direction Δx and the Lagrange multiplier y are determined as the solution to the quadratic programming approximation to (1)

$$(9) \quad \begin{aligned} \min \quad & \frac{1}{2} \Delta x^T Q \Delta x + \nabla f(x)^T \Delta x \\ \text{s.t.} \quad & A(x) \Delta x + h(x) \geq 0, \\ & \|\Delta x\|_\infty \leq r, \end{aligned}$$

where Q is some positive definite approximation to the Hessian matrix of $f(x)$, typically a quasi-Newton approximation, and r is a trust region radius. Sequential-quadratic-programming algorithms also require a merit function to ensure a balance between reducing the objective function and reducing infeasibility. A merit function akin to (8) for this problem is

$$(10) \quad \Psi_\beta(x) = f(x) + \frac{\beta}{2} \|\rho^-(x)\|^2$$

where $\rho^-(x)$ is the vector with elements $\rho_i^-(x) = \min(h_i(x), 0)$. Here, reducing $\Psi_\beta(x)$ clearly ensures that either the objective function or the infeasibility is reduced.

The goal of Fletcher and Leyffer's work was to replace the use of a merit function in sequential-quadratic-programming with its two components: (a) a measure of objective progress and (b) a measure of progress toward feasibility. So, Fletcher and Leyffer define a *filter* to be a set of pairs of points $f(x^{(k)})$ and $\|\rho^-(x^{(k)})\|$. A new point $x^{(k+1)}$ is admitted to the filter if no point $x^{(j)}$ already in the filter dominates $x^{(k+1)}$, that is,

$$(11) \quad \begin{aligned} \|\rho^-(x^{(j)})\| &\leq \|\rho^-(x^{(k+1)})\|, \\ f(x^{(j)}) &\leq f(x^{(k+1)}). \end{aligned}$$

If there is a point $x^{(j)}$ in the filter such that (11) is satisfied, an acceptable point $x^{(k+1)}$ is determined either by reducing the trust region radius r or by a feasibility step. Readers interested in the details are referred to [4].

The purpose of this paper is to study replacing the merit function (8) in the interior-point algorithm of LOQO with a filter of the Fletcher-Leyffer type. The next section discusses how to implement a filter for an interior-point algorithm, and suggests three alternative methods. The final section tests the variants on a large suite of problems, comparing them to each other and to the standard LOQO algorithm. Numerical results show that a properly designed filter algorithm improves the performance of an interior-point algorithm.

3. FILTER ALGORITHMS FOR INTERIOR-POINT METHODS

As noted in the previous section, at each iteration, there exists a β such that the search direction which solves (6) is a descent direction for the merit function $\Psi_{\beta,\mu}(x, w)$ given by (8). This implies that a steplength α can be found to reduce $\Psi_{\beta,\mu}(x, w)$, which in turn implies that either the barrier objective function

$$(12) \quad b_\mu(x, w) = f(x) - \mu \sum_{i=1}^m \log(w_i)$$

or the norm of the infeasibility $\|\rho(x, w)\|$ will be reduced at each iteration. This fact immediately suggests using a filter consisting of pairs of points $b_\mu^{(k)}$ and $\|\rho^{(k)}\|$, where

$$b_\mu^{(k)} = b_\mu(x^{(k)}, w^{(k)}) \quad \text{and} \quad \rho^{(k)} = \rho(x^{(k)}, w^{(k)}).$$

An example of such a filter consisting of four points is shown in Figure 1.

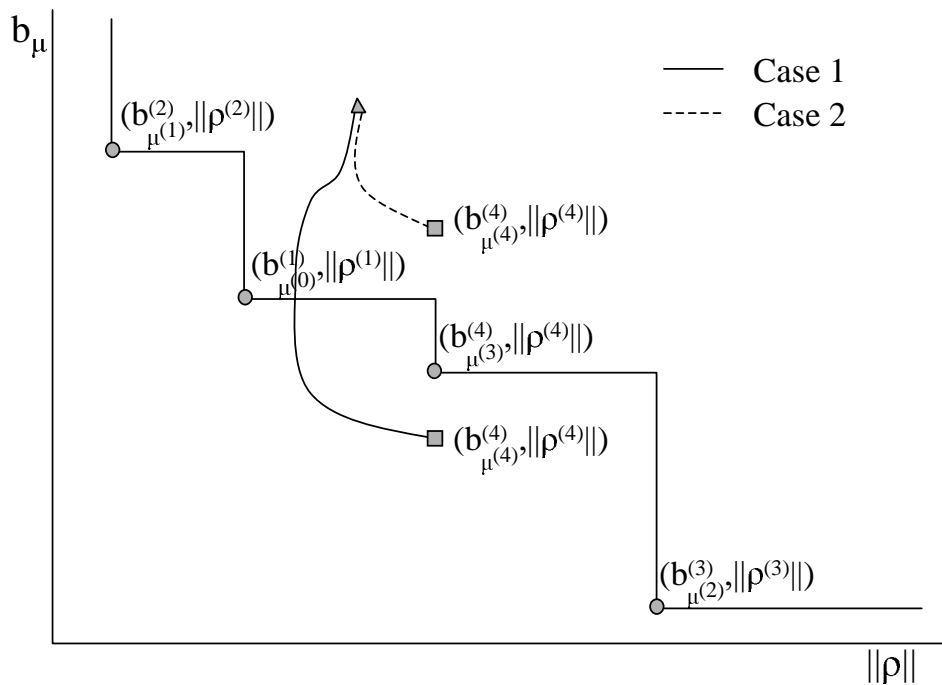


FIGURE 1. The filter shown incorporates the barrier objective. It consists of four points. Two possible cases for what might happen in iteration 5 are shown. The circles denote the points in the filter, the squares represent locations of filter points after updating the barrier parameter, and the triangle is the point obtained by taking an uncut step in the step direction computed in iteration 5.

The main difficulty in applying filter methods in the context of interior-point algorithms stems from the fact that the barrier parameter changes from one iteration to the next. We will denote by $\mu^{(k-1)}$ the barrier parameter used in iteration k , since it is computed from $(x^{(k-1)}, w^{(k-1)})$. As discussed above, a steplength α exists at iteration $k+1$ that reduces either $b_{\mu^{(k)}}^{(k)}$ or $\|\rho^{(k)}\|$. But, since $b_{\mu^{(k)}}^{(k)}$ is different from $b_{\mu^{(k-1)}}^{(k)}$, we might not find a steplength that will give a point acceptable to the filter at iteration k . In fact, Figure 1 depicts two possible locations for $(b_{\mu^{(4)}}^{(4)}, \|\rho^{(4)}\|)$. In Case 1, we have $b_{\mu^{(4)}}^{(4)} < b_{\mu^{(3)}}^{(4)}$, and we are guaranteed to find a point acceptable to the filter in iteration 5. However, in Case 2, it is impossible to find a steplength that will give us such a point.

In general, in order to guarantee that we can find a point

$$\left(b_{\mu^{(k)}}^{(k+1)}, \|\rho^{(k+1)}\| \right)$$

that is acceptable to the filter, it is sufficient to have

$$(13) \quad b_{\mu^{(k)}}^{(k)} < b_{\mu^{(k-1)}}^{(k)}.$$

This inequality holds if

$$\mu^{(k)} \leq \mu^{(k-1)}$$

and

$$-\sum_{i=1}^m \log(w_i^{(k+1)}) \geq 0.$$

In fact, it is usually the case that the barrier parameter, μ , is monotonically decreasing, and always so as the optimum is approached. Thus, (13) will most likely hold as the algorithm approaches the optimum, and the suggested filter is plausible.

However, (13) may fail to hold in early iterations when LOQO does not reduce the barrier parameter μ monotonically. We cannot, therefore, implement a filter method in our algorithm without modifying Fletcher and Leyffer's approach or modifying the μ calculation to ensure monotonicity. We tried the latter approach, but several problems in our test set failed due to μ becoming too small in the first few iterations. In fact, since the inception of barrier methods, the problem of choosing an initial value for μ has proved highly problematic. Even in linear programming, it is necessary to let μ vary both upward and downward in any practical implementation.

In the rest of this section, we will present three filter-based algorithms and discuss their properties.

3.1. The Barrier Filter. As discussed above, it is natural, in the context of interior-point algorithms, to use the barrier function instead of the objective function. Our first variant implements this approach. Of course, we must address the issues raised at the beginning of this section, which we now do.

Suppose that, at each iteration, we save three pieces of information: $(f(x), \sum_{i=1}^m \log(w_i), \|\rho(x, w)\|)$. Then, when a new $\mu^{(k)}$ is calculated, each barrier objective function can be recalculated using this new value, and a new filter constructed. A new point $(x^{(k+1)}, w^{(k+1)})$ is then computed by successively halving the steplength as necessary until it is acceptable to the filter; that is, until there is no point $(x^{(j)}, w^{(j)})$ corresponding to an entry in the filter that satisfies:

$$(14) \quad b_{\mu^{(k)}}(x^{(j)}, w^{(j)}) \leq b_{\mu^{(k)}}(x^{(k+1)}, w^{(k+1)}), \quad 1 \leq j \leq k$$

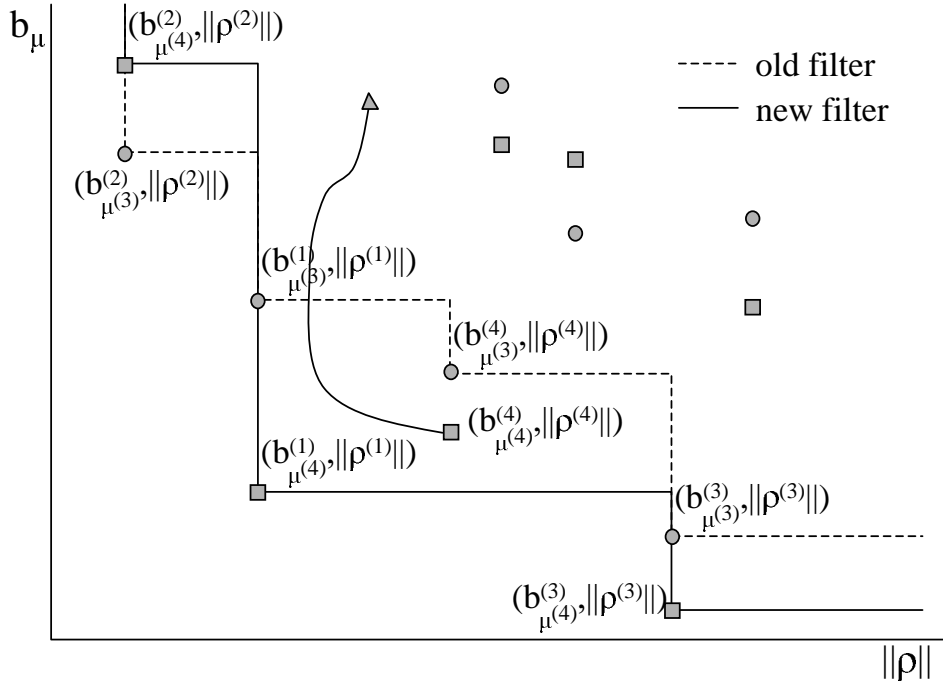


FIGURE 2. A barrier objective filter that is updated with the barrier parameter μ at each iteration. The circles denote the points in the filter, the squares represent locations of filter points after updating the barrier parameter, and the triangle is the point obtained by taking an uncut step in the step direction computed in iteration 5.

and

$$\|\rho(x^{(j)}, w^{(j)})\|^2 \leq \|\rho(x^{(k+1)}, w^{(k+1)})\|^2, \quad 1 \leq j \leq k.$$

This variant is illustrated in Figure 2. (While successive halving of the step length may seem inefficient compared to quadratic or cubic interpolation, in almost all cases no backtracking needs to be done. If it is required, one cutback is usually sufficient. Thus a more sophisticated backtracking algorithm is unwarranted.)

However, there is no guarantee that a new point acceptable to the filter can be found at each iteration. In Figure 2, we depict one possible scenario, where $b^{(1)}$ is reduced by such an amount that $b^{(4)}$ is no longer in the filter. In that situation, we cannot find a steplength that will give a point acceptable to the filter.

The question then arises as to what to do when no new trial point $(x^{(k+1)}, w^{(k+1)})$ is acceptable to the filter.

Since we know that there exists a β such that the search vector $(\Delta x, \Delta w, \Delta y)$ is a descent vector for $\Psi_{\beta, \mu}(x, w)$, one strategy is to compute the β as in standard LOQO [11] and do a line search to find a steplength that will reduce $\Psi_{\beta, \mu}(x, w)$. While the new point must improve either the infeasibility or the barrier objective over the previous point, it need not be acceptable to the filter. Nonetheless, we accept the new point as the current point and continue.

To summarize, our first filter-based approach uses the objective function, the barrier term and the norm of the infeasibility to create a dynamically changing filter. The filter is recomputed for each new value of the barrier parameter μ . Since it may still be the case that we cannot find a steplength to give a point acceptable to the filter, we also employ, as a fall-back, a merit function to test and accept a new point.

3.2. The Fletcher-Leyffer Filter. For our second variant, we abandon the barrier function and its desirable properties described above and use instead the objective function itself as in the original work of Fletcher and Leyffer. That is, at iteration k , we shorten the steplength until there is no prior point $(x^{(j)}, w^{(j)})$ that dominates the new point:

$$\begin{aligned} f(x^{(j)}) &\leq f(x^{(k+1)}), \\ \|\rho(x^{(j)}, w^{(j)})\|^2 &\leq \|\rho(x^{(k+1)}, w^{(k+1)})\|^2. \end{aligned}$$

The rationale for this approach is that if $\mu^{(k)} \leq \mu^{(k+1)}$ and the pair $(x^{(k+1)}, w^{(k+1)})$ is feasible and minimizes $b(x, w, \mu^{(k)})$, then $f(x^{(k+1)}) \leq f(x^{(k)})$ (see Fiacco and McCormick [3]).

However, it may still be the case that we cannot find a steplength to give a point acceptable to the filter. Again, we employ a merit function as in the Barrier Filter to resolve this issue.

3.3. The Markov Filter. Our third variant is not really a filter at all since we abandon the idea of keeping information from the previous iterations and instead base our steplength reduction entirely on the current point and step direction. Of course, the merit function approach was also myopic, but it restricted our choice of steplength more. This can be seen in Figure 3, where less shortening of the steplength is required at iteration $k + 1$ to find a point acceptable to the Markov filter.

With this approach, we simply backtrack until a reduction in either the infeasibility or the barrier objective over the previous point is achieved, that is,

$$(15) \quad b_{\mu^{(k)}}(x^{(k+1)}, w^{(k+1)}) \leq b_{\mu^{(k)}}(x^{(k)}, w^{(k)})$$

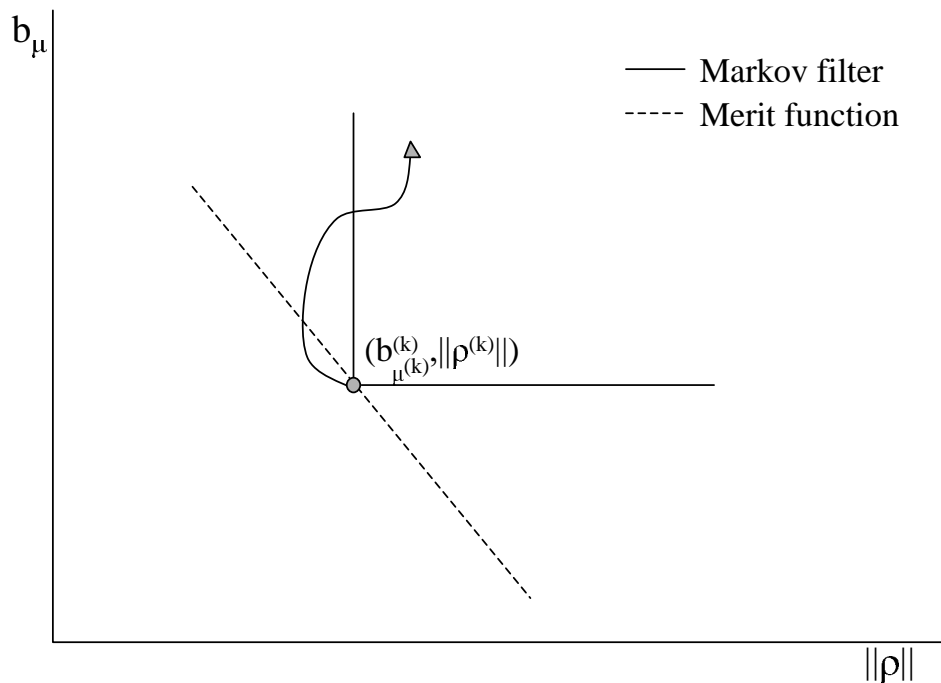


FIGURE 3. The Markov filter vs. the merit function at iteration k . The circle denotes the point in the filter, and the triangle is the point obtained by taking an uncut step in the step direction computed in iteration $k + 1$.

or

$$\|\rho(x^{(k+1)}, w^{(k+1)})\|^2 \leq \|\rho(x^{(k)}, w^{(k)})\|^2.$$

Doing so clearly avoids the need for the penalty parameter in the merit function, and is in the spirit of the filter, but is less complex.

4. PRACTICAL CONSIDERATIONS.

In this section, we present some of the implementation details for the filter-based algorithms presented above.

4.1. Sufficient reduction. In all of the above, we require a sufficient decrease in either the infeasibility or the barrier objective. In practice, we impose an Armijo-type condition on the decrease. Specifically, in the Barrier Filter we require that either

$$(16) \quad b_{\mu^{(k)}}^{(k+1)} \leq b_{\mu^{(k)}}^{(j)} + \epsilon\alpha \begin{bmatrix} \nabla_x b_{\mu^{(k)}}^{(k)} \\ \nabla_w b_{\mu^{(k)}}^{(k)} \end{bmatrix}^T \begin{bmatrix} \Delta x^{(k)} \\ \Delta w^{(k)} \end{bmatrix}$$

or

$$(17) \quad \|\rho^{(k+1)}\|^2 \leq \|\rho^{(j)}\|^2 + 2\epsilon\alpha\rho^{(k)T} [\nabla_x\rho^{(k)}\nabla_w\rho^{(k)}] \begin{bmatrix} \Delta x^{(k)} \\ \Delta w^{(k)} \end{bmatrix}$$

for all $(x^{(j)}, w^{(j)})$ in the filter. Here, α is the steplength and ϵ is a small constant, usually taken to be around 10^{-6} .

Note that the Armijo condition imposed on the barrier objective and the infeasibility are different from the standard Armijo condition. In its standard form, a measure of sufficient decrease from the j th iteration would be based on $\nabla b_{\mu^{(k)}}^{(j)}$, $\nabla\rho^{(j)}$, $\Delta x^{(j)}$, and $\Delta w^{(j)}$. However, we are not guaranteed that $(\Delta x^{(j)}, \Delta w^{(j)})$ are descent directions for $b_{\mu^{(k)}}^{(j)}$. To see this, note that if $(\Delta x^{(j)}, \Delta w^{(j)})$ are indeed descent directions for $b_{\mu^{(j)}}^{(j)}$, the following inequality holds:

$$\nabla f(x^{(j)})^T \Delta x^{(j)} - \mu^{(j)} e^T (W^{(j)})^{-1} \Delta w^{(j)} \leq 0.$$

The inequality that we want to hold is the same except that $\mu^{(k)}$ appears in place of $\mu^{(j)}$:

$$\nabla f(x^{(j)})^T \Delta x^{(j)} - \mu^{(k)} e^T (W^{(j)})^{-1} \Delta w^{(j)} \leq 0.$$

If $e^T (W^{(j)})^{-1} \Delta w^{(j)} \leq 0$, this inequality is guaranteed to hold only if $\mu^{(k)} \leq \mu^{(j)}$. Otherwise, it is guaranteed to hold only if $\mu^{(k)} > \mu^{(j)}$. Since neither of these conditions can be assumed to hold, we cannot use $(\Delta x^{(j)}, \Delta w^{(j)})$ as descent directions for $\nabla b_{\mu^{(k)}}^{(j)}$.

However, we should note that the aim of the Armijo condition is to create an ‘‘envelope’’ around the filter, and so it would be sufficient to note that $\Delta x^{(k)}$ and $\Delta w^{(k)}$ are descent directions for either $b_{\mu^{(k)}}^{(k)}$ or $\|\rho^{(k)}\|^2$. Therefore, the condition given by (16) achieves our goal and is easy to implement.

For the case where $\Delta x^{(k)}$ and $\Delta w^{(k)}$ are not descent directions for $b_{\mu^{(k)}}^{(k)}$, it is still easy to obtain a sufficient reduction. First, note that

$$\rho^{(k)T} [\nabla_x\rho^{(k)}\nabla_w\rho^{(k)}] \begin{bmatrix} \Delta x^{(k)} \\ \Delta w^{(k)} \end{bmatrix} = -\rho^{(k)T} \rho^{(k)} \leq 0.$$

The equality follows from the fact $\Delta x^{(k)}$ and $\Delta w^{(k)}$ are obtained using Newton’s Method, and satisfy the third equation of (6). Therefore, $\Delta x^{(k)}$ and $\Delta w^{(k)}$ are descent directions for $\|\rho^{(k)}\|^2$. We can always define a valid envelope for the infeasibility, so we can use this information to approximate what the corresponding envelope for the barrier objective would be. The dual variables are a measure of the proportion of

marginal change in the barrier objective to marginal change in the infeasibility. Using the ℓ_∞ norm of the vector of dual variables to obtain a proportional envelope for the barrier objective will suffice.

We have thus shown that we can guarantee the existence of an envelope around the filter simply by using information from the previous iteration. Furthermore, we are guaranteed that we can always find an α that will give us a sufficient decrease over the previous iteration. This allows us to obtain a sufficient decrease at each iteration.

For the filter algorithm that uses the objective function, (16) is replaced with

$$(18) \quad f^{(k+1)} \leq f^{(j)} + \epsilon\alpha (\nabla f^{(j)})^T \Delta x^{(j)}$$

which is the standard Armijo condition.

For the third filter-based algorithm, we only compare against the last iterate, so (16) is replaced with

$$(19) \quad b_{\mu^{(k)}}^{(k+1)} \leq b_{\mu^{(k)}}^{(k)} + \epsilon\alpha \begin{bmatrix} \nabla_x b_{\mu^{(k)}}^{(k)} \\ \nabla_w b_{\mu^{(k)}}^{(k)} \end{bmatrix}^T \begin{bmatrix} \Delta x^{(k)} \\ \Delta w^{(k)} \end{bmatrix},$$

and (17) is replaced with

$$(20) \quad \|\rho^{(k+1)}\|^2 \leq \|\rho^{(k)}\|^2 + 2\epsilon\alpha \rho^{(k)T} [\nabla_x \rho^{(k)} \nabla_w \rho^{(k)}] \begin{bmatrix} \Delta x^{(k)} \\ \Delta w^{(k)} \end{bmatrix}$$

Note that the last two expressions correspond to the standard Armijo condition.

We have also incorporated into our code measures to avoid a large increase in either the barrier objective (or objective) or the infeasibility in exchange for a small decrease in the other. If we have a solution that satisfies

$$\|\rho(x^{(k)}, w^{(k)})\| \leq 10^{-10},$$

then we require a sufficient decrease in the barrier objective (or objective) function for the next iteration or a decrease in the infeasibility of at least one order of magnitude in the next iteration.

Also, if the primal infeasibility is exactly 0, we insist on a sufficient decrease in the barrier objective (or objective) function for the next iteration.

Finally, we should note that in order to save time, a maximum of 10 step cuts can be performed at each iteration. This is the value used in LOQO originally, and it has been retained in the filter-based algorithms as well.

4.2. Feasibility Restoration. The Markov Filter, as presented above, may run into further numerical difficulties when the iterates are feasible and close to optimality. It may, in fact, be the case that the current point is superoptimal and the infeasibility is a very small value less than 10^{-10} . Then, the maximum number of 10 step cuts may not reduce the barrier objective function to be less than the superoptimal value, and we may not be able to reduce the infeasibility by one order of magnitude, either. The algorithm simply gets stuck at this point, doing 10 step cuts at each iteration, with the result that it will either fail without achieving the default levels of accuracy or it will slow down considerably.

Fortunately, this is an easy situation to remedy. When the infeasibility level is small and 10 step cuts are being performed at each iteration, the infeasibility reduction required is changed from one order of magnitude back to the Armijo condition, and this allows the algorithm to step back to a more feasible solution and attain an optimal solution with the default levels of accuracy.

5. NUMERICAL TESTING: COMPARISON OF ALGORITHMS

Fletcher and Leyffer report encouraging numerical results for the performance of the filter method on their sequential quadratic programming code, filterSQP, as compared to their original code, ℓ_1 SQP, and to the Conn, Gould, Toint trust-region code, LANCELOT. In this study, it is our goal to ascertain the effects of using filter-based methods in place of a merit function in an interior-point method setting. To the best of our knowledge, no such previous study exists that compares the two approaches as implemented within the framework of the same interior-point algorithm.

We have experimented with the three variants of the filter-based approach (and others). In this section, we provide comparisons between the current version of LOQO and the approaches discussed in the previous section. Thus, the four algorithms in our numerical tests are:

- Merit Function (LOQO)
- Barrier Filter (BF)
- Fletcher-Leyffer Filter (FLF)
- Markov Filter (MF)

As any code using Newton's method requires second partial derivatives, we have chosen to formulate the models in AMPL [5], a modelling language that provides analytic first and second partial derivatives. In order to construct a meaningful test suite, we have been engaged in reformulating from *standard input format* (SIF) to AMPL all models in

the CUTE [1] (constrained and unconstrained testing environment) test suite. To date, we have converted and validated 699 models. For those problems with variable size, we have used the largest suggested number of variables and constraints, except in the case of the `ncvxqp` family of problems and `fminsurf`, where the largest suggested sizes were beyond the capabilities of all solvers. In addition, we have expressed the entire Schittkowski [7] test suite in AMPL. Together, this comprises a test suite of 889 AMPL models. Some of these models are infeasible, unbounded, or nondifferentiable, so our final test set for this study uses 871 of them. These models vary greatly in size and difficulty and have proved useful in drawing meaningful conclusions. All of the AMPL models used in our testing are available at [9].

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

We have made the algorithm variants from LOQO Version 5.06, which was called from AMPL Version 20000814. Comparisons of LOQO to other general-purpose nonlinear solvers can be found in two independent papers [6] and [2]. All testing was conducted on a SUN SPARC Station running SunOS 5.8 with 4GB of main memory and a 400MHz clock speed.

We have found in our testing that the filter-based algorithms outperform LOQO. In fact, the ability to take bolder steps improves the algorithm quite markedly, and MF benefits the most from this. As will be documented in the rest of this section, MF is the fastest algorithm of the four in terms of iteration counts and runtimes on a majority of the problems. This is especially true for large-scale problems where the runtime improvements are substantial.

The test suite can be broken down by size of the problems, where we define the size by the number of variables plus the number of constraints in the model. “Small” problems have size less than 100, “Medium” problems have size 100 to less than 1000, “Large” problems have size 1000 to less than 10000, and “Very Large” problems have size 10000 or more. Note that the size reported may differ from the model itself, since AMPL preprocesses a problem before passing it to the solvers. The total number of problems in each category, along with the number of problems solved by each code, is given in Table 1.

Due to the size of our test set, detailed results are too voluminous to present here. Instead, we summarize the results using the methodology

Size	Total	LOQO	BF	FLF	MF
Small	574	571	570	571	568
Medium	98	90	90	89	90
Large	104	103	103	103	103
Very Large	95	95	95	95	95

TABLE 1. Total number of problems and the number of problems solved by each code in each size category.

recently proposed by Dolan and Moré [2]. Their approach is simply to 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.

Assume that we are comparing the performance of n_s solvers on n_p problems. If we are using runtime as the performance metric, let

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

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

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : 1 \leq s \leq n_s\}}.$$

Here, we also assume that the set of problems we are working on are such that all solvers can find a solution to all of the problems. Thus, $r_{p,s} < \infty$.

In order to evaluate the performance of the algorithm on the whole set of problems, one can use the quantity

$$\phi_s(\tau) = \frac{1}{n_p} \text{card}\{p : 1 \leq p \leq n_p, r_{p,s} \leq \tau\}.$$

The function $\phi_s : \mathbb{R} \rightarrow [0, 1]$ is called the *performance profile* and represents the cumulative distribution function of the performance ratio.

In Figures 4 and 5, we present performance profiles using two different metrics. In this study, we have included problems that were either solved untuned by all solvers or used the same tuning for all, and we also required that all of the solvers find the same local optimum. Our stopping criteria is 8 digits of agreement between the primal and dual objective functions as well as primal and dual infeasibility less than 10^{-6} . With these criteria, two programs were considered to have found the same local optimum if the objective functions agree to 6 decimal places. Figure 4 shows the performance profiles of the four solvers with respect to runtime. There are 186 models considered here—the

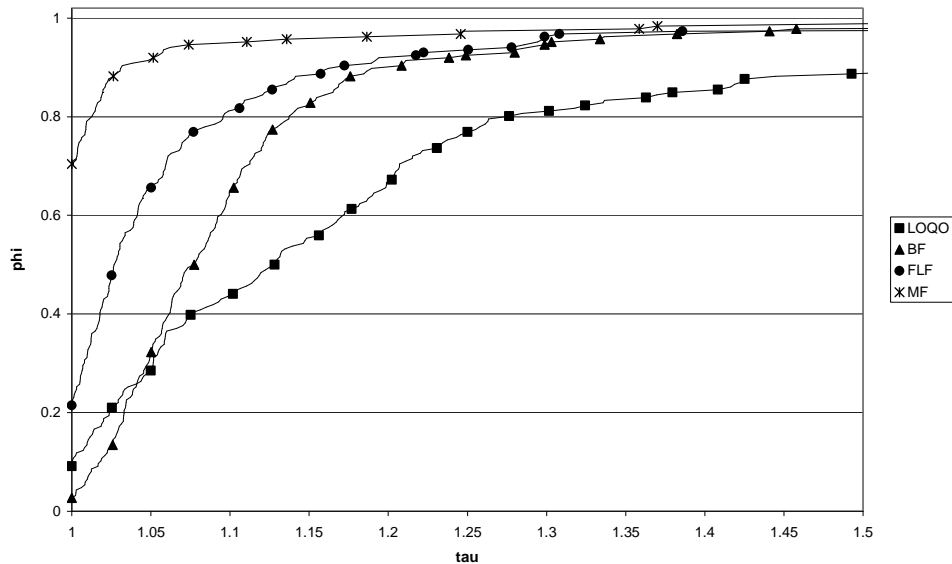


FIGURE 4. Performance profiles of LOQO and the filter algorithms with respect to runtime.

reason for this number being so low is that we have not included models whose runtimes were less than 1 CPU second. On such problems, small differences in system performance would have provided an unfair comparison. For a more detailed comparison, we have included Figure 5, which uses the iteration counts as the system metric. This study included 784 models.

It is clear from both figures that MF is superior to the other three codes, both in terms of runtime and iteration counts. On the other hand, FLF seems to be superior to BF in terms of runtime. This may be due to both of them performing quite similarly on the iteration counts, but with FLF requiring less work per iteration.

We also provide a pairwise comparison of LOQO and MF in Table 2. We present total iteration counts and runtimes for those problems where one of the solvers took less iterations to reach the optimum than the other. We have included problems that were not solved with the original settings of the LOQO parameters but were solvable with tuning. The parameters that we most often tune are initial values of the slack variables (*bndpush*), the primal and dual infeasibility tolerance (*inftol*), and the number of digits of agreement required between the primal and dual solutions (*sigfig*). For a summary of which problems need tuning and their respective tuning parameters, see [6]. In our pairwise comparison, we only include those problems that were either not tuned

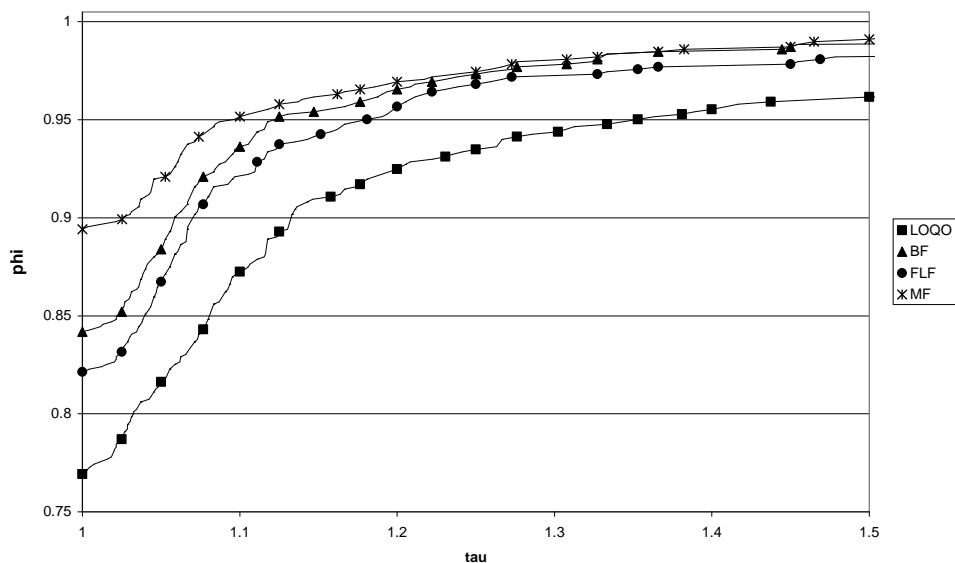


FIGURE 5. Performance profiles of LOQO and the filter algorithms with respect to iteration counts.

			LOQO		MF	
			Iter	Time	Iter	Time
Small	LOQO Better	47	1969	36.77	2649	46.13
	MF Better	105	6533	13.73	4292	5.75
Medium	LOQO Better	8	802	50.42	868	55.22
	MF Better	15	1154	29.93	972	25.95
Large	LOQO Better	5	222	18.09	254	20.00
	MF Better	23	2590	1566.87	2278	1296.61
Very Large	LOQO Better	4	168	202.60	205	255.86
	MF Better	27	1214	2101.54	854	1202.33

TABLE 2. Comparison of LOQO to MF on commonly solved problems.

by either solver or had the same tuning parameters for both, and where both solvers reached the same optimum.

Using a filter approach can allow the algorithm to take bolder steps. A step that may otherwise have been shortened because it did not reduce a linear combination of the barrier objective and the primal infeasibility may be accepted by reducing either one. Table 2 shows that, in general, taking bolder steps allows the algorithm to reach the optimum quicker, as MF performs better than LOQO more often than

vice versa. Also, more freedom in taking steps at the beginning reduces the need to adjust for the initialization of slack variables.

The difference in steplengths also affects the accuracy of the solution. Whereas taking bolder steps may allow us to reach better accuracy levels, it is just as likely according to our numerical testing for less restricted steps to keep us from reaching the default accuracy levels.

Finally, since the test set contains nonconvex problems, the algorithms may end up at different optima, and the quality of the solution attained may be different. We found that each algorithm was just as likely to attain a better optimum than a worse one.

6. CONCLUSIONS.

In this paper, we propose three different ways to implement a filter-based method to control steplengths in an interior-point algorithm. The first two use a multi-point filter as in Fletcher and Leyffer, one with the barrier objective and the other just the objective function. However, for nonmonotonic barrier parameter the algorithm might not produce an acceptable new point no matter how short a step is taken. To remedy this situation, both variants are supplemented with a traditional merit function that is used if the filter approach fails. These two variants work equally well in practice.

The third variant is simpler than Fletcher and Leyffer's filter method, imposing filter-like conditions only on the previous iteration. This approach obviates the need for a merit function, and, numerically, it outperforms the current version of the algorithm as well as the other two filter-based variants.

In general, the numerical results show that filter-based algorithms are superior in terms of efficiency to ones based only on a merit function. This is especially evident on large problems in our test sets, where the nonlinearities are simpler and a filter-based algorithm is able to take longer steps and reach the optimum in fewer iterations.

6.1. Dedication. This paper is dedicated to Garth McCormick on the occasion of his 65th birthday. Without his pioneering work with his colleague Anthony Fiacco, current interior-point algorithms would not exist.

REFERENCES

- [1] A.R. Conn, N. Gould, and Ph.L. Toint. Constrained and unconstrained testing environment. <http://www.dci.clrc.ac.uk/Activity.asp?CUTE>.
- [2] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. Technical report, Argonne National Laboratory, January 2001.

- [3] A.V. Fiacco and G.P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. Research Analysis Corporation, McLean Virginia, 1968. Republished in 1990 by SIAM, Philadelphia.
- [4] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. Technical Report NA/171, University of Dundee, Dept. of Mathematics, Dundee, Scotland, 1997.
- [5] R. Fourer, D.M. Gay, and B.W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Scientific Press, 1993.
- [6] D.F. Shanno H.Y. Benson and R.J. Vanderbei. Interior-point methods for non-convex nonlinear programming: Filter methods and merit functions. Technical Report ORFE 00-06, Department of Operations Research and Financial Engineering, Princeton University, 2000.
- [7] Klaus Schittkowski. *More Test Samples for Nonlinear Programming Codes*. Springer Verlag, New York, 1987.
- [8] 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.
- [9] R.J. Vanderbei. AMPL models. <http://www.sor.princeton.edu/~rvdb/ampl/nlmodels>.
- [10] R.J. Vanderbei. LOQO: An interior point code for quadratic programming. *Optimization Methods and Software*, 12:451–484, 1999.
- [11] R.J. Vanderbei and D.F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13:231–252, 1999.

HANDE Y. BENSON, PRINCETON UNIVERSITY, PRINCETON, NJ

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

ROBERT J. VANDERBEI, PRINCETON UNIVERSITY, PRINCETON, NJ