SIAM J. OPTIM.
Vol. 15, No. 1, pp. 17–38

A MULTIDIMENSIONAL FILTER ALGORITHM FOR NONLINEAR EQUATIONS AND NONLINEAR LEAST-SQUARES[*]

NICHOLAS I. M. GOULD[†], SVEN LEYFFER[‡], AND PHILIPPE L. TOINT[§]

Abstract. We introduce a new algorithm for the solution of systems of nonlinear equations and nonlinear least-squares problems that attempts to combine the efficiency of filter techniques and the robustness of trust-region methods. The algorithm is shown, under reasonable assumptions, to globally converge to zeros of the system, or to first-order stationary points of the Euclidean norm of its residual. Preliminary numerical experience is presented that shows substantial gains in efficiency over the traditional monotone trust-region approach.

Key words. nonlinear equations, nonlinear least-squares, nonlinear fitting, filter methods, trust-region algorithms

AMS subject classifications. 65K05, 90C30

DOI. 10.1137/S1052623403422637

1. Introduction.

We analyze a filter algorithm for solving systems of nonlinear equations. More formally, we consider the problem of solving

$$c(x) = 0, \tag{1.1}$$

where c is a twice continuously differentiable function from $\mathbb{R}^n$ into $\mathbb{R}^m$. We partition the equations of (1.1) into p (not necessarily disjoint) sets $\{c_i(x)\}_{i\in\mathcal{I}_j}$ for $j = 1,\ldots,p$, with $\{1,\ldots,n\} = \mathcal{I}_1 \cup \mathcal{I}_2 \cup \cdots \cup \mathcal{I}_p$, and define

$$\theta_j(x) \stackrel{\text{def}}{=} \|c_{\mathcal{I}_j}(x)\| \text{ for } j = 1,\ldots,p, \tag{1.2}$$

where $\|\cdot\|$ is the ordinary Euclidean norm and where $c_{\mathcal{I}_j}$ is the vector whose components are the components of c indexed by $\mathcal{I}_j$. The point x is therefore a solution of (1.1) if and only if $\theta_j(x) = 0$ for $j = 1,\ldots,p$. The quantity $\theta_j(x)$ may be interpreted as the size of the residual of the jth set of equations at the point x. We will also use the abbreviations $\theta(x) = (\theta_1(x),\ldots,\theta_p(x))^T$, $\theta_k \stackrel{\text{def}}{=} \theta(x_k)$, and $\theta_{j,k} \stackrel{\text{def}}{=} \theta_j(x_k)$. In the simplest case, that is, when $p = m$ and $\mathcal{I}_j = \{j\}$, we have that $\theta_{j,k} = |c_j(x_k)|$ and $\|\theta_k\| = \|c(x_k)\|$.

We follow the classical approach for solving (1.1), which consists of minimizing a merit function involving some norm of the residual. For simplicity, we choose to consider

$$\min_{x\in\mathbb{R}^n} f(x) = \tfrac{1}{2}\|\theta(x)\|^2. \tag{1.3}$$

Note that this is a least-squares formulation of (1.1), which justifies the second part of the paper's title. (The least-squares are weighted with integer weights if the subsets

[*]Received by the editors February 7, 2003; accepted for publication (in revised form) January 1, 2004; published electronically October 1, 2004.
http://www.siam.org/journals/siopt/15-1/42263.html

[†]Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, England (n.gould@rl.ac.uk).

[‡]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439 (leyffer@mcs.anl.gov).

[§]Department of Mathematics, University of Namur, B-5000 Namur, Belgium (philippe.toint@fundp.ac.be).

$\mathcal{I}_j$ are not disjoint.) Our objective is therefore to find a (local) minimizer x_* of $f(x)$. If $f(x_*) = 0$, x_* is also a solution of (1.1).

The class of algorithms that we discuss for achieving this objective belongs to the class of trust-region methods and also to that of *filter methods* introduced by Fletcher and Leyffer in 1997 and subsequently published as [8]. Although originally intended for the solution of constrained optimization problems, we claim here that the main idea of the approach, namely, that of a filter to decide on acceptability of the successive iterates, may be extended to the context of (1.1). The question is of interest, since most of the recent contributions to the field of filter algorithms (see, for instance, Fletcher and Leyffer [8], Ulbrich, Ulbrich, and Vicente [24], Chin and Fletcher [2], Fletcher, Leyffer, and Toint [10], Gould and Toint [17], Fletcher et al. [7], and Wächter and Biegler [25]) rely on an external "restoration procedure" whose purpose is to reduce constraint infeasibilities (i.e., when only equality constraints are present, solve a problem of the type (1.1), albeit possibly approximately). The proposal by Gonzaga, Karas, and Vanti [11] goes even further as it uses such a computation as part of the step calculation at every iteration. The fact that (1.1) can itself be handled by a filter algorithm, which, as will be seen below, does not explicitly depend on any external restoration procedure, therefore significantly extends the range of applicability of the filter idea.

The question of using filter techniques for the solution of nonlinear equations has already been considered by Fletcher and Leyffer [9], but their approach is very different from the one proposed here. It relies on a quadratic programming solver to partition the constraints into a set of violated constraints and a set of satisfied or active constraints. These sets are refined from iteration to iteration. It is not clear how this approach can be extended to large-scale systems of equations where only approximate subproblem solves are available and this limits the applicability of [9] to problems that allow active-set-type solvers to be used. By contrast, our approach takes all violations into account together and does not require the revision of any subset during the algorithm. This makes it applicable to very large systems of equations that require inexact subproblem solves to be efficient.

We introduce our algorithm for problem (1.1) in section 2 and discuss its convergence in section 3. Numerical experience is reported in section 4, and some preliminary conclusions are drawn in section 5.

2. Nonlinear equations, filter, and trust regions.

An efficient technique for solving (1.1) is to use the Newton or the Gauss–Newton method (or one of their many variants like secant methods), which, from a current iterate x_k (and possibly from information gathered at iterations preceding iteration k), computes a trial step s_k and thus yields a *trial point*

$$x_k^+ = x_k + s_k.$$

Unfortunately, such algorithms may not be convergent from an arbitrary starting point x_0, and we may have to include it within a framework that guarantees this very desirable property. Our proposal is to use a suitable combination of the trust region (see Conn, Gould, and Toint [4]) and filter techniques.

The main idea of filter algorithms for constrained optimization is that new iterates of the underlying iterative algorithm can be accepted if they do not perform, compared to past iterates kept in the filter, worse on both important and typically conflicting accounts for this type of problem, namely, feasibility and low objective function value. In the context of nonlinear equations, we no longer have to consider an objective function, but we still face conflicting purposes. Indeed, we may consider driving each of the $\{c_i(x)\}_{i=1}^m$ (or each of the $\{\theta_i(x)\}_{i=1}^p$) to zero as an independent task, which is typically conflicting with that of driving the other components of c (or θ) to zero. Thus we will consider a multidimensional filter, instead of a two-dimensional one.

2.1. The multidimensional filter. To define our filter, we first say that a point x_1 *dominates* a point x_2 whenever

$$\theta_j(x_1) \leq \theta_j(x_2) \text{ for all } j = 1, \ldots, p.$$

Thus, if iterate x_{k_1} dominates iterate x_{k_2}, the latter is of no real interest to us since x_{k_1} is at least as good as x_{k_2} for each of the equation sets. All we need to do now is to remember iterates that are not dominated by other iterates using a structure called a filter. A *filter* is a list $\mathcal{F}$ of p-tuples of the form $(\theta_{1,k}, \ldots, \theta_{p,k})$ such that

$$\theta_{j,k} < \theta_{j,\ell} \text{ for at least one } j \in \{1, \ldots, p\}$$

for $k \neq \ell$. Filter methods propose to accept a new trial iterate x_k^+ if it is not dominated by any other iterate in the filter and x_k. In the vocabulary of multicriteria optimization, this amounts to building elements of the efficient frontier associated with the p-criteria problem of reducing infeasibility on each of the p sets of equations.

While the idea of not accepting dominated trial points is simple and elegant, it needs to be refined a little to provide an efficient algorithmic tool. In particular, we do not wish to accept a new point x_k^+ if $\theta_k^+ \overset{\text{def}}{=} \theta(x_k^+)$ is arbitrarily close to being dominated by another point already in the filter. To avoid this situation, we slightly strengthen our acceptability condition. More formally, we say that a new trial point x_k^+ is *acceptable for the filter* $\mathcal{F}$ if and only if

$$\forall \theta_\ell \in \mathcal{F} \quad \exists j \in \{1, \ldots, p\} \quad \theta_j(x_k^+) < \theta_{j,\ell} - \gamma_\theta \delta(\|\theta_\ell\|, \|\theta_k^+\|), \tag{2.1}$$

where $\gamma_\theta \in (0, 1/\sqrt{p})$ is a small positive constant and where $\delta(\cdot, \cdot)$ is one of the following:

$$\delta(\|\theta_\ell\|, \|\theta_k^+\|) = \|\theta_\ell\|, \tag{2.2}$$

$$\delta(\|\theta_\ell\|, \|\theta_k^+\|) = \|\theta_k^+\|, \quad \text{or} \tag{2.3}$$

$$\delta(\|\theta_\ell\|, \|\theta_k^+\|) = \min(\|\theta_\ell\|, \|\theta_k^+\|). \tag{2.4}$$

The first of these formulas measures the margin as a function of the constraints's violation corresponding to the existing filter point; the second measures it as a function of the constraints's violation at the trial point. The third formula attempts to weaken these conditions even further by considering only the smallest of the two violations. The upper bound of $1/\sqrt{p}$ on γ_θ ensures that the right-hand side of (2.1) is always positive for some j for the choices (2.2) and (2.4) and thus that points acceptable for the filter always exist in these cases. Note that such points must exist if (2.3) is considered provided $\|\theta_\ell\| > 0$, but a small value for γ_θ clearly makes it more likely that (2.1) holds for a given θ_k^+.

2.1.1. Adding a new point to the filter. To avoid cycling, and assuming the trial point is acceptable in the sense of (2.1), we may wish to add it to the filter, so as to avoid other iterates that are worse. The procedure is extremely simple: we simply perform the operation $\mathcal{F} \leftarrow \mathcal{F} \cup \{\theta_k\}$. But this may cause an existing filter entry θ_ℓ to be *strongly dominated* in the sense that every new entry θ_k^+ that is acceptable for the other filter points must also be acceptable for θ_ℓ. We therefore remove θ_ℓ from the filter if

$$\exists\, \theta_q \in \mathcal{F} \quad \forall j \in \{1,\ldots,p\} \quad \theta_{j,\ell} - \gamma_\theta \|\theta_\ell\| \geq \theta_{j,q} - \gamma_\theta \|\theta_q\|$$

if we have chosen to use (2.2),

$$\exists\, \theta_q \in \mathcal{F} \quad \forall j \in \{1,\ldots,p\} \quad \theta_{j,\ell} \geq \theta_{j,q} \tag{2.5}$$

if we have chosen to use (2.3), or

$$\exists\, \theta_q \in \mathcal{F} \quad \forall j \in \{1,\ldots,p\} \quad \theta_{j,\ell} - \gamma_\theta \|\theta_\ell\| \geq \theta_{j,q}$$

if we have chosen to use (2.4). That this last formula is adequate results from

$$\theta_{j,k}^+ < \theta_{j,q} - \gamma_\theta \min[\|\theta_k^+\|, \|\theta_q\|] \leq \theta_{j,q} \leq \theta_{j,\ell} - \gamma_\theta \|\theta_\ell\| \leq \theta_{j,\ell} - \gamma_\theta \min[\|\theta_k^+\|, \|\theta_q\|],$$

which shows that if θ_k^+ is acceptable for θ_q, then it must be acceptable for θ_ℓ.

2.1.2. Computing a trial point. We also need to indicate how to compute the trial point $x_k^+ = x_k + s_k$ for some step s_k. This assumes we have a model $m_k(x)$ of $f(x)$ and a *trust region*

$$\mathcal{B}_k = \{x_k + s \mid \|s\|_k \leq \Delta_k\}, \tag{2.6}$$

where we believe this model to be reasonably accurate. The norms $\|\cdot\|_k$ are allowed to vary from iteration to iteration provided they remain uniformly equivalent to the Euclidean norm. The convergence analysis that follows requires, as is common in trust-region methods, that this step provides, at iteration k, a *sufficient decrease* on the model: we require that

$$m_k(x_k) - m_k(x_k + s_k) \geq \kappa_{\text{mdc}} \|g_k\| \min\left[\frac{\|g_k\|}{\beta_k}, \Delta_k\right], \tag{2.7}$$

where $g_k = \nabla m_k(x_k)$, β_k is a positive upper bound on the norm of the Hessian of m_k, and κ_{mdc} is a constant in $(0,1)$. This is a natural requirement in the context of trust-region algorithms (see [4, p. 131], for instance), and there are efficient algorithms for achieving (2.7) in the trust-region framework for both small- and large-scale cases. This condition is also satisfied if the model is further reduced outside the trust region, as is the case in our algorithm when $\|s_k\| > \Delta_k$.

At variance with classical trust-region methods, we do not require that

$$\|s_k\|_k \leq \Delta_k \tag{2.8}$$

at every iteration to obtain our most important convergence result. In particular, this means that whenever (2.8) is not enforced and $m = n$, we might use the step $s_k = -J_k^{-1} c_k$, where J_k is the Jacobian of $c(x)$ at x_k (assumed here to be full rank).

This step leads to the root of the first-order model for $c(x)$, which is also the global minimizer of

$$m_k^{\rm GN}(x_k+s) = \tfrac{1}{2}\sum_{i=1}^{p}\|c_{\mathcal{I}_i}(x_k)+J_{\mathcal{I}_i}(x_k)s\|^2, \tag{2.9}$$

a Gauss–Newton model for $f(x)$, where $J_{\mathcal{I}_i}$ is the Jacobian of $c_{\mathcal{I}_i}$. This model may also be minimized if $m > n$ and (2.8) is not enforced, or one could choose in this case to minimize the second-order truncation of the Taylor series for $f(x)$,

$$m_k^{\rm N}(x_k+s) = m_k^{\rm GN}(x_k+s) + \tfrac{1}{2}\sum_{i=1}^{p}\sum_{j\in\mathcal{I}_i} c_j(x_k)\langle s, \nabla^2 c_j(x_k)s\rangle, \tag{2.10}$$

which corresponds to a full Newton model, whenever this last model is convex.

Algorithm 2.1. Filter-Trust-Region Algorithm.

Step 0: Initialization.
An initial point x_0 and an initial trust-region radius $\Delta_0 > 0$ are given, as well as constants $0 < \gamma_0 \le \gamma_1 < 1 \le \gamma_2$, $\gamma_\theta \in (0, 1/\sqrt{p})$, $0 < \eta_1 < \eta_2 < 1$. Assume that the sets $\{\mathcal{I}_j\}_{j=1}^p$ are also given. Compute $c_0 = c(x_0)$ and θ_0. Set $k = 0$, define a flag `RESTRICT` to be unset and initialize the filter to the empty set.

Step 1: Test for optimality.
If $\theta_k = 0$ or $\|\nabla f(x_k)\| = 0$, stop.

Step 2: Determine a trial step.
Compute a step s_k that satisfies (2.7) and that also satisfies (2.8) if `RESTRICT` is set. Compute the trial point $x_k^+ = x_k + s_k$.

Step 3: Evaluate the residual at the trial step.
Compute $c(x_k^+)$ and $\theta_k^+ = \theta(x_k^+)$. Define

$$\rho_k = \frac{f(x_k) - f(x_k^+)}{m_k(x_k) - m_k(x_k^+)}.$$

Step 4: Test to accept the trial step.
- If x_k^+ is acceptable for the current filter:
  set $x_{k+1} = x_k^+$, unset `RESTRICT`, and add θ_k^+ to the filter if either $\rho_k < \eta_1$ or (2.8) fails.
- If x_k^+ is not acceptable for the current filter:
  If (2.8) holds and $\rho_k \ge \eta_1$, set $x_{k+1} = x_k^+$ and unset `RESTRICT`. Else, set $x_{k+1} = x_k$ and set `RESTRICT`.

Step 5: Update the trust-region radius.
If $\|s_k\|_k \le \Delta_k$, update the trust-region radius by choosing

$$\Delta_{k+1} \in \begin{cases} [\gamma_0\Delta_k, \gamma_1\Delta_k] & \text{if } \rho_k < \eta_1, \\ [\gamma_1\Delta_k, \Delta_k] & \text{if } \rho_k \in [\eta_1, \eta_2) \\ [\Delta_k, \gamma_2\Delta_k] & \text{if } \rho_k \ge \eta_2; \end{cases}$$

otherwise, set $\Delta_{k+1} = \Delta_k$. Increment k by one and go to Step 1.

2.2. The algorithm. We now combine these ideas into an algorithm. The main objective of Algorithm 2.1 is to let the filter play the major role in ensuring global convergence and to fall back on the usual ℓ_2-norm reduction algorithm if things do not go well, or if convergence occurs to a local minimizer of f which is not a zero of c.

This algorithm provides only a canvas for a practical implementation, as we have not fully specified each step. In particular, we have not detailed how the step s_k can be computed in practice to satisfy the sufficient decrease condition (2.7) and remain in the trust region when required, but the literature provides several choices of direct or iterative techniques; see Moré and Sorensen [20], Gould et al. [12], or Chapter 7 of [4], for example. Similarly, techniques to initialize and update the trust-region radius can be found in Chapter 17 of this last reference. As is usual, we say that iteration k is successful if $x_{k+1} = x_k + s_k$.

3. Global convergence. We now investigate the convergence properties of Algorithm 2.1, under the following set of assumptions:

A1: $c(x)$ is twice continuously differentiable on $\mathbb{R}^n$.
A2: The iterates x_k remain in a bounded domain of $\mathbb{R}^n$.
A3: $m_k(x)$ is twice differentiable on $\mathbb{R}^n$ for all k.
A4: For all k, $m_k(x_k) = f(x_k)$ and $g_k = \nabla m_k(x_k) = \nabla f(x_k)$.

For the purpose of the convergence analysis, we also assume that the algorithm does not terminate in Step 1.

Note that A1, A2, and A3 together imply that there is a constant $\kappa_{\rm u} > 0$ such that

$$\|c(x)\| \le \kappa_{\rm u}, \quad \|\nabla^2 c_i(x)\| \le \kappa_{\rm u}, \quad \text{and} \quad \|\nabla^2 m_k(x)\| \le \kappa_{\rm u} \tag{3.1}$$

for all k and all x in the convex hull of $\{x_k\}$. The second of these inequalities then ensures that $\kappa_{\rm u}$ can also be chosen such that

$$\|\nabla^2 f(x)\| \le \kappa_{\rm u}.$$

We could have assumed the three conditions (3.1) independently instead of imposing A2, but we have chosen not to do so for the sake of simplicity. Assumptions A1, A3, A4, and (3.1) are typical of convergence theory for trust-region methods (see Chapters 6 and 16 of [4]).

We first investigate what happens if infinitely many values are added to the filter in the course of the algorithm.

LEMMA 3.1. *Suppose that* A1 *and* A2 *hold and that infinitely many values of* θ_k *are added to the filter by the algorithm. Then*

$$\lim_{k\to\infty} \|c_k\| = \lim_{k\to\infty} \|\nabla f(x_k)\| = 0. \tag{3.2}$$

Proof. Let $\{k_i\}$ index the subsequence of iterations at which $\theta_{k_i} = \theta^+_{k_i-1}$ is added to the filter. Now assume, for the purpose of a contradiction, that there exists a subsequence $\{k_j\} \subseteq \{k_i\}$ such that

$$\|\theta_{k_j}\| \ge \epsilon_1$$

for some $\epsilon_1 > 0$. Since our assumptions imply that $\{\|\theta_{k_j}\|\}$ is bounded above and below, there must exist a further subsequence $\{k_\ell\} \subseteq \{k_j\}$ such that

$$\lim_{\ell\to\infty} \theta_{k_\ell} = \theta_\infty \qquad \text{with} \qquad \|\theta_\infty\| \ge \epsilon_1. \tag{3.3}$$

Moreover, by definition of $\{k_\ell\}$, θ_{k_ℓ} is acceptable for the filter for every ℓ, which implies in particular that for each ℓ, there exists a $j \in \{1, \ldots, p\}$ such that

$$\theta_{j,k_\ell} - \theta_{j,k_{\ell-1}} < -\gamma_\theta \delta(\|\theta_{k_{\ell-1}}\|, \|\theta_{k_\ell}\|). \tag{3.4}$$

Furthermore, this conclusion does not depend on $\theta_{k_{\ell-1}}$ still being in the filter. Indeed, if $\theta_{k_{\ell-1}}$ has been removed from the filter, it must be strongly dominated by another entry in the filter, which then guarantees that (3.4) holds. But (3.3) and our assumptions on δ imply that there exists an $\epsilon_2 > 0$ such that

$$\delta(\|\theta_{k_{\ell-1}}\|, \|\theta_{k_\ell}\|) \geq \epsilon_2$$

for all ℓ sufficiently large. Hence we deduce from (3.4) that

$$\theta_{j,k_\ell} - \theta_{j,k_{\ell-1}} < -\gamma_\theta \epsilon_2$$

for ℓ sufficiently large. But the left-hand side of this inequality tends to zero when ℓ tends to infinity because of (3.3), yielding the desired contradiction. Hence

$$\lim_{i\to\infty} \|\theta_{k_i}\| = 0. \tag{3.5}$$

Consider now any $\ell \notin \{k_i\}$ and let $k_{i(\ell)}$ be the last iteration before ℓ such that $\theta_{k_{i(\ell)}}$ was added to the filter. Since, by construction, every successful iteration where the value of θ at the trial point is not included in the filter must result in a decrease of the objective function (since $\rho_k \geq \eta_1$ on such iterations), we deduce that, for all $\ell \notin \{k_i\}$,

$$f(x_\ell) \leq f(x_{k_{i(\ell)}}),$$

and therefore that

$$\|\theta(x_\ell)\| \leq \|\theta(x_{k_{i(\ell)}})\|.$$

Combining this inequality with (3.5) and the fact that $k_{i(\ell)} \in \{k_i\}$, we obtain that

$$\lim_{k\to\infty} \|\theta_k\| = 0,$$

which implies (3.2) and completes the proof. □

Let us now consider the case where only finitely many values are added to the filter in the course of the algorithm, and let us examine first the case where the number of trial points acceptable for the filter is also finite. Hence, the main test of Step 4 fails for all $k \geq k_0$, say, and the method is then identical to an ordinary trust-region method for minimizing $f(x)$, since iterations for which (2.8) fails have no effect on the current iterate nor on the trust region radius and may be ignored in the convergence theory. Our assumptions are sufficient to use the results of Chapter 6 in [4] and we may therefore deduce the following two properties.

Lemma 3.2. *Suppose that* A1–A4 *hold and that only finitely many trial points* x_k^+ *are acceptable for the filter. Suppose furthermore that there are only finitely many successful iterations. Then the algorithm terminates in Step* 1 *with* $x_k = x_*$*, and* $\nabla f(x_*) = 0$.

Lemma 3.3. *Suppose that* A1–A4 *hold, that only finitely many trial points* x_k^+ *are acceptable for the filter, and that there are infinitely many successful iterations. Then*

$$\liminf_{k\to\infty} \|\nabla f(x_k)\| = 0.$$

We have thus covered all cases, except that where only finitely many values are added to the filter but infinitely many trial points are acceptable for the filter. In this situation, we therefore have for sufficiently large k three kinds of intertwined iterations:

- The first are iterations where x_k^+ is not acceptable for the filter but (2.8) fails. These iterations have no effect and may be ignored, as we argued before Lemma 3.2.
- The second are iterations where x_k^+ is not acceptable for the filter but (2.8) holds. These are perfectly standard trust-region iterations.
- In the third kind, x_k^+ is acceptable for the filter, but θ_k^+ is not added to it because $\rho_k \geq \eta_1$ and (2.8) holds. These are again standard (successful) trust-region iterations.

The sequence of iterations can therefore be viewed as resulting from a standard trust-region algorithm, and known convergence results are again applicable. As a consequence, the conclusions of Lemmas 3.2 and 3.3 are also valid in this context.

We summarize our results in the next theorem.

THEOREM 3.4. *Suppose that* A1–A4 *hold. Then there exists a subsequence* $\{k_i\}$ *such that* $\liminf_{i\to\infty} \|\nabla f(x_{k_i})\| = 0$. *Moreover, if infinitely many values are added to the filter, then* $\lim_{k\to\infty} \|c(x_k)\| = 0$.

Note that this theorem does not guarantee that a subsequence of $\{\|c_k\|\}$ converges to zero in all cases, but only that a stationary point of $f(x)$ is reached. This is expected because it may happen that the equations (1.1) have no solution or that the iterates get trapped near a local minimum of $f(x)$ with positive value. Such an outcome might be troublesome if one aims at finding a root of (1.1), as opposed to a locally minimum residual, but avoiding this situation requires global optimization techniques, which are outside the scope of this paper.

The convergence theory of trust-region algorithms also implies (see [4, Theorem 6.4.6]) that the limit inferior in Lemma 3.3 can be replaced by a true limit provided

$$\|s_k\|_k \leq \kappa_\Delta \Delta_k \quad \text{for all } k \geq k_0, \tag{3.6}$$

for some $k_0 > 0$ and some constant $\kappa_\Delta \geq 1$. In the usual trust-region context, this is automatically guaranteed since the trial point always lies within the trust region itself. However, this property no longer holds for the algorithm described in this paper, because unrestricted steps are possible. But we may still obtain the stronger result if we are ready to assume that (3.6) holds for some κ_Δ possibly larger than one, which might be seen as a reasonable implementation safeguard. The rest of the discussion above then remains unchanged and we deduce our final convergence result.

THEOREM 3.5. *Suppose that* A1–A4 *and* (3.6) *hold. Then either* $\|\nabla f(x_k)\| = 0$ *for some finite* k *or* $\lim_{k\to\infty} \|\nabla f(x_k)\| = 0$. *Moreover, if infinitely many values are added to the filter, then we have that* $\lim_{k\to\infty} \|c(x_k)\| = 0$.

4. Numerical experience.

4.1. Test problems and performance measures.

We now look at the practical behavior of the new method when applied to test problems of the CUTEr collection[1] (see Bongartz et al. [1] and Gould, Orban, and Toint [13]). Table 4.1 indicates the

[1] We have excluded the problems FLOSP2HH, FLOSP2HL, FLOSP2HM, FLOSP2TL, FLOSP2TM, and SEMICON1 because their solution appears to require more specific preconditioning (they all require more than 1 hour of CPU time for every option we tried) and problems CHEMRCTA and CHEMRCTB because they seemed to generate numerical overflow with all the tested methods.

Table 4.1
Test problem characteristics.

Problem	n	n free	m	$\|\theta_*\|$	Problem	n	n free	m	$\|\theta_*\|$
ARGLALE	400	400	800	+	DRCAVTY1	4489	3969	3969	0
ARGLBLE	400	400	800	+	DRCAVTY2	4489	3969	3969	0
ARGLCLE	400	400	799	+	DRCAVTY3	4489	3969	3969	0
ARGTRIG	200	200	200	0	EIGENAU	2550	2550	2550	0
ARTIF	100002	100000	100000	0	EIGENB	2550	2250	2250	0
ARWHDNE	500	500	998	+	EIGENC	2652	2652	2652	0
BDVALUES	10002	10000	10000	0	INTEGREQ	1002	1000	1000	0
BRATU2D	123904	122500	122500	0	MSQRTA	4900	4900	4900	0
BRATU2DT	23104	22500	22500	0	MSQRTB	4900	4900	4900	0
BRATU3D	4913	3375	3375	0	POROUS1	5184	4900	4900	0
BROWNALE	1000	1000	1000	0	POROUS2	5184	4900	4900	0
BROYDN3D	100000	100000	100000	0	SEMICN2U	5002	5000	5000	0
BROTDNBD	100000	100000	100000	0	SPMSQRT	27001	27001	44999	0
CBRATU2D	7200	6728	6728	0	WOODSNE	80000	80000	60001	+
CBRATU3D	16000	11664	11664	0	YATP1SQ	123200	123200	123200	0
CHANDHEU	500	500	500	0	YATP2SQ	123200	123200	123200	0
CHANNEL	19200	19198	19198	0					

names and dimensions of the test problems considered. The columns $\|\theta_*\|$ in this table indicate whether the residual at the solution is zero (0) or positive (+). As is apparent from this table, the test set includes a fair mix of problems sizes and over- or underdetermined cases. All experiments reported in this section were run with the current version of the Fortran 95 FILTRANE package (see Gould and Toint [16]) on a Dell Latitude C840 portable computer (1.6 MHz, 1 Gbyte RAM) under the Fujitsu frt Fortran compiler with default optimization. The values $\gamma_0 = 0.625$, $\gamma_1 = 0.25$, $\gamma_2 = 2$, $\eta_1 = 0.01$, $\eta_2 = 0.9$, and

$$\gamma_\theta = \min\left[0.001, \frac{1}{2\sqrt{p}}\right]$$

were used.

In what follows, we compare several variants of the algorithm discussed above for reliability and efficiency. All variants discussed use the Euclidean norm to define the trust region (in (2.6)) and

$$\delta(\|\theta_\ell\|, \|\theta_k^+\|) = \|\theta_k^+\|.$$

This latter choice was made because it simplifies the removal of strongly dominated filter entries (see (2.5)).

Efficiency comparisons are made using the performance profiles introduced by Dolan and Moré [6]. Suppose that a given algorithmic variant i from a set $\mathcal{A}$ reports a statistic $s_{ij} \geq 0$ when run on example j from our test set $\mathcal{T}$, and suppose that the smaller this statistic, the better the variant is considered. Let

$$k(s, s^*, \sigma) = \begin{cases} 1 & \text{if } s \leq \sigma s^*, \\ 0 & \text{otherwise.} \end{cases}$$

Then, the *performance profile* of algorithm i is the function

$$p_i(\sigma) = \frac{\sum_{j\in\mathcal{T}} k(s_{ij}, s_j^*, \sigma)}{|\mathcal{T}|} \quad (\sigma \geq 1),$$

where $s_j^* = \min_{i\in\mathcal{A}} s_{ij}$. ($s_{ij}$ is defined as infinity if variant i failed on problem j, which then implies that $k = 0$.) Thus $p_i(1)$ gives the fraction of the number of examples for which algorithm i was the most effective (according to statistics s_{ij}), $p_i(2)$ gives the fraction of the number for which algorithm i is within a factor of 2 of the best, and $\lim_{\sigma\longrightarrow\infty} p_i(\sigma)$ gives the fraction of the examples for which the algorithm succeeded. We consider such a profile to be a very effective means of comparing the relative merits of our algorithmic variants.

We have chosen to compare unpreconditioned versions of the new methods, NITSOL and LANCELOT-B (see section 4.5), as no preconditioning was reasonably successful on our test problems and also because the CUTEr interface to NITSOL only allows for its use without a preconditioner. It must be noted, however, that all tested methods provide facilities for preconditioning, which would typically be required to solve more difficult problems. We ran additional tests with our new technique using diagonal and band preconditioners, which confirm the findings reported below for the unpreconditioned case.

4.2. Filter versus trust region and pure Newton. We start by examining the most obvious question, namely, that of the potential added value of the filter technique compared to the more traditional trust-region approach. We therefore consider the following algorithmic variants.

GNFTR. This variant is Algorithm 2.1, where, at each iteration, the trial point is computed by approximately minimizing $m^{\mathrm{GN}}(x_k + s)$ using the generalized Lanczos trust-region algorithm of Gould et al. [12] as implemented in the GALAHAD library [14]. This procedure is terminated at the first s for which

$$\|\nabla m^{\mathrm{GN}}(x_k + s)\| \le \min\left[0.1, \sqrt{\max(\epsilon_M, \|\nabla m^{\mathrm{GN}}(x_k)\|)}\right] \|\nabla m^{\mathrm{GN}}(x_k)\|, \tag{4.1}$$

where ϵ_M is the machine precision. The choice $\theta(x) = c(x)$ is also made, which implies that an m-dimensional filter is used. The bound (3.6) is imposed with $\kappa_\Delta = 1000$ at all iterations following the first one at which a restricted step was taken. In addition, the condition

$$f(x_k + s_k) \le \min(10^6 f(x_0), f(x_0) + 1000)$$

is imposed for the trial point to be acceptable for the filter. Instead of the formal requirement of Step 1, the algorithm stops if

$$\|\nabla f(x_k)\| \le 10^{-6}\sqrt{n} \ \text{ or } \ \|c(x_k)\|_\infty \le 10^{-6}, \tag{4.2}$$

or if the number of iterations exceeds 1000, or if the computing time exceeds 1 hour. Other details of the implementation are discussed in Gould and Toint [16] and do not matter in our comparison.

GNTR. This variant is identical to GNFTR except that all trial steps are considered as unacceptable for the filter. The resulting method is therefore nothing but a basic trust-region algorithm (Algorithm BTR in [4]).

GNDD. This is a dare-devil variant identical to GNFTR except that every trial step is considered to be acceptable for the filter. Note that this implies that the filter entries need not be stored, since comparisons are no longer performed. Note also that this method does not include any mechanism for guaranteeing global convergence.

All three variants are reliable, even if they all reported failure (while still producing very good approximations to the solution) on problems ARGLBLE and ARGLCLE, two very ill-conditioned linear least-squares problems. GNFTR required more than 1000 iterations for `DRCAVTY3`. GNDD was stopped after 1 hour of CPU time on `SEMICN2U` and required more than 1000 iterations on `ARWHDNE`. It furthermore failed on `YATP2SQ` because it wandered off into a region where the Hessian of the Gauss–Newton model is so ill-conditioned that negative curvature was erroneously reported in the course of the conjugate-gradients iteration, causing divergence. The good reliability of GNDD is surprising, given that Newton's method is known to fail on some problems if not augmented by a globalization mechanism. We therefore investigated if this behavior could be caused by the starting points for the test problems being too close to the problem's solution. To test this hypothesis, we ran all test problems again, perturbing the starting point using the transformation $x_0 \leftarrow 10x_0 + 1$, but this gave essentially the same results (except that fewer problems were suitable for comparison as local minimizers of the constraints violation were found more often). Another potential explanation of this phenomenon is simply that the CUTEr test problem collection contains too few examples of large-scale nonlinear systems where an algorithm like GNDD diverges,[2] although these test problems are derived from a number of different application areas including nonlinear PDEs, nonlinear ODEs, nonlinear matrix equations, and general algebraic nonlinear systems. Unfortunately, we have no real means of assessing the generality of our problem collection.[3] A last possible explanation of GNDD's remarkable reliability is that GNDD does in fact converge on a large class on nonlinear systems but that our understanding is currently too limited to explain this behavior.

The relative efficiency of these methods is illustrated by the performance profile in Figures 4.1 and 4.2. The overall performance of the filter variants is impressive. These profiles show that GNFTR is the clear winner in CPU time: the GNFTR filter variant appears to be within a factor of two of the best for approximately 95% of the test problems (it is fastest on 74%). The dominance of the GNFTR variant over the GNTR variants is even stronger if one considers the number of iterations (which, in our setting, is equal to the number of constraints evaluations and approximately[4] equal to the number of Jacobian evaluations). The efficiency gains obtained by augmenting the trust-region algorithm with a filter-based acceptance rule for trial points thus appear to be very significant. It is also interesting to note that GNFTR is very comparable to GNDD for the problems where the latter does not fail, indicating that the filter technique may have the efficiency of a "blind" Newton's method while preserving guaranteed convergence (in the sense of Theorem 3.4). Finally, the dominance of GNDD over GNTR for small values of σ indicates that the trust-region mechanism might impose too heavy a burden on Newton's method to ensure its global convergence.

Given that each filter entry is an m-dimensional vector, storing many filter entries may impose large memory requirements. Fortunately, large filters appear to be exceptional, as can be seen in Figure 4.3, where we show the relation between m and the maximal filter size (over all iterations and all test examples involving m equations). With the exception of problem `SEMICN2U`, where 293 filter entries were required for $m = 5000$, all runs required less than 45, irrespective of the value of m. The memory requirements thus remain typically modest.

[2]GNDD also fails on a few small-scale problems from the CUTEr collection, like `HEART8`, `PFIT1`, `PFIT3`, `PFIT4`, and `RSNBRNE`.

[3]However, we are always pleased to include further problems that are brought to our attention.

[4]Equality does not hold because the Jacobian is not evaluated at unsuccessful iterations.

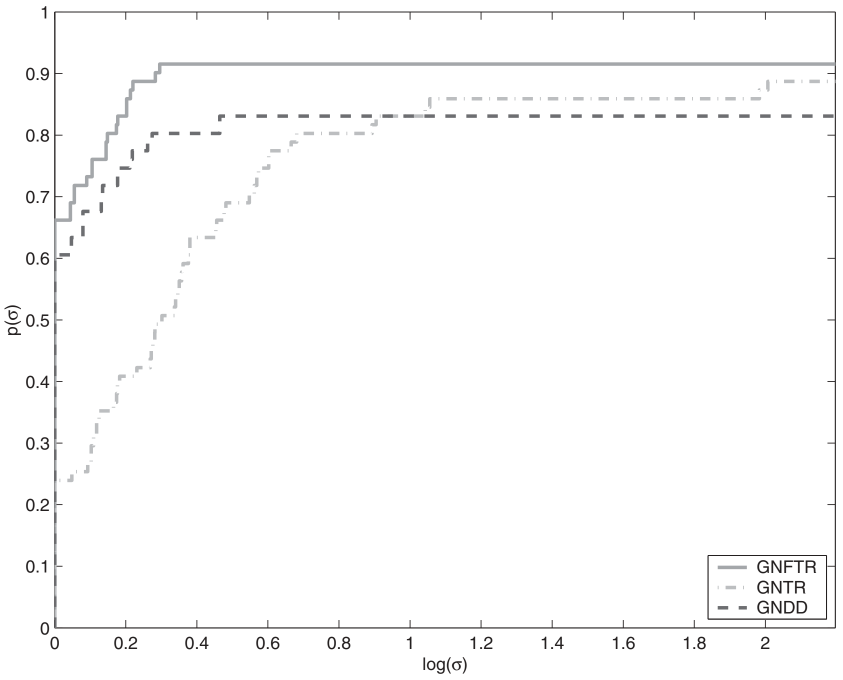

Fig. 4.1. *CPU time performance profile for GNFTR, GNTR, and GNDD.*

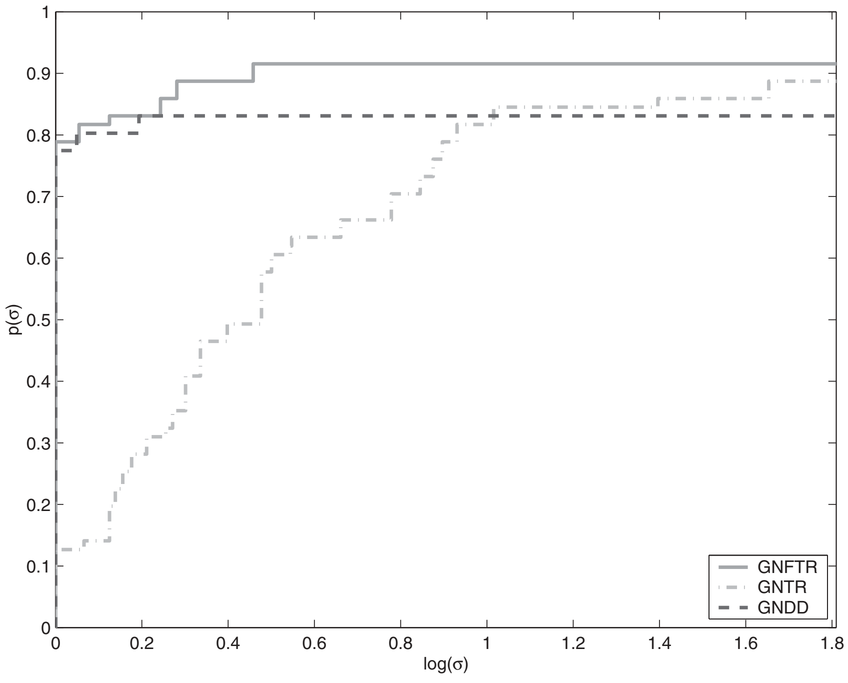

Fig. 4.2. *Iteration number performance profile for GNFTR, GNTR, and GNDD.*

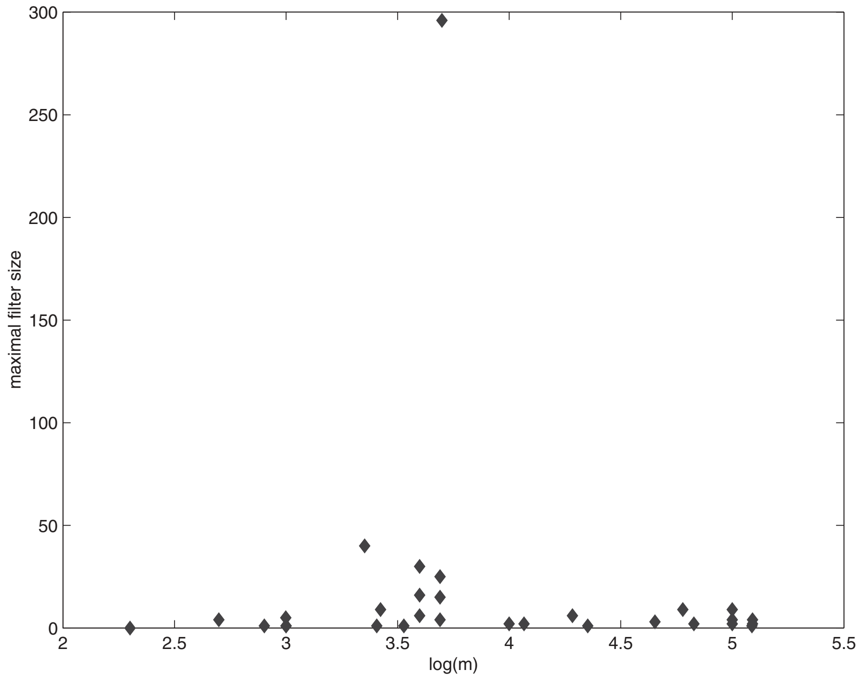

FIG. 4.3. *The maximal number of entries in the filter as a function of* $\log(m)$.

4.3. Model choice and subproblem accuracy. We next consider the choice of the Gauss–Newton model (2.9) and compare it to the full Newton model given by (2.10), in each case using default (see (4.1)) and full machine accuracy for the solution of the linear subproblem. An important feature of (2.10) is that, in contrast with (2.9), it need no longer be convex. While this is of no consequence in the pure trust-region context because the trust-region boundary prevents arbitrary long steps toward unbounded minima of (2.10), this causes a problem for Algorithm 2.1. Indeed, negative curvature of the model may be discovered when unrestricted steps are computed. This might result in unnecessarily long steps, which have then to be cut back, generating inefficient oscillations in the step length. We have thus added to Algorithm 2.1 the feature that unrestricted steps for which negative curvature is detected are automatically recomputed with `RESTRICT` set without even attempting to compute the constraints values at the trial point. This can be done at minimal cost by using the reentry facility in the GLTR package, which exploits previously computed Krylov spaces (see Gould et al. [12] for details). We thus introduce three new variants of Algorithm 2.1:

- NFTR is identical to GNFTR except that the model (2.10) is used instead of (2.9) and unrestricted steps containing negative curvature are restricted before attempting to compute the constraint values at the trial point.
- GNFTR-full is identical to GNFTR except that (4.1) is replaced by

$$\|\nabla m^{\mathrm{GN}}(x_k + s)\| \le \sqrt{\epsilon_M}\|\nabla m^{\mathrm{GN}}(x_k)\|. \tag{4.3}$$

- NFTR-full is identical to NFTR except that (4.1) is replaced by (4.3).

Comparing the reliability of GNFTR, GNFTR-full, NFTR, and NFTR-full, we note that, as above, all four variants stall on `ARGLBLE` and `ARGLCLE`. NFTR has not converged in 1000 iterations on `DRCAVTY3`, and this is also the case for GNFTR-full on `ARTIF` and `DRCAVTY3` and for NFTR-full on `DRCAVTY2`. The package was terminated after 1 hour of CPU time for GNFTR-full on `DRCAVTY2`, `MSQRTA`, and `MSQRTB`, for NFTR-full on `BDVALUES`, `EIGENC`, `MSQRTA`, `MSQRTB`, `DRCAVTY3`, `SEMICN2U`, and `YATP2SQ`. The variants using the full Newton model therefore seem less reliable than the Gauss–Newton ones on our test set. Furthermore, they are less efficient, as is apparent in the performance profiles of Figures 4.4 and 4.5. But these conclusions must be tempered by the observation that many of our test problems are actually such that $\|c(x_*)\| = 0$, therefore implicitly favoring the Gauss–Newton model, as it is well known that the full Newton model is typically more efficient when the solution of the problem occurs at a point x_* for which $\|c(x_*)\|$ is strictly positive.

Occasionally, however, there is considerable benefit from the Newton model. The most extreme example in this direction is that of the `ARWHDNE` problem, which is also the problem for which $\|c(x_*)\|$ is largest. It is detailed in Table 4.2.

TABLE 4.2
Detailed performance of the GNFTR, GNFTR-full, NFTR, and NFTR-full variants on problem `ARWHDNE`.

	CPU(s)	iter.	CG iter.
GNFTR	0.2	178	240
GNFTR-full	0.2	179	273
NFTR	0.1	6	11
NFTR-full	0.1	6	12

Adaptive procedures to choose between the Gauss–Newton and the Newton model have been studied (see Dennis, Gay, and Welsch [5], Toint [22], or Lukšan [19], for instance), but their application to our framework is beyond the scope of this paper and is postponed to [16].

4.4. Equation grouping. We finally examine the impact of varying p, the dimension of the filter space, by defining sets of equations as explained in the introduction. We have chosen to compare our initial GNFTR variant (in which every equation has its own subset) with three different alternative strategies for grouping equations in k disjoint subsets (groups), with $k = n/10$, 10, and 1. In our experiments, we simply assigned the ith equation to the group

$$s(i) = \begin{cases} \bmod(i,k) & \text{if } \bmod(i,k) > 0, \\ k & \text{otherwise.} \end{cases}$$

Observe that if all equations are merged into a single group, (2.1) and (2.3) reduce to

$$\|c(x_k^+)\| < \frac{1}{1+\gamma_\theta}\|c(x_{\ell(k)})\|,$$

where $\ell(k)$ is the last entry added to the filter before or at iteration k. It follows from the design of the algorithm that every successful iteration between $\ell(k)$ and k must produce a successful trust-region step, since otherwise the corresponding trial step would have been added to the filter in Step 4. This method may thus be seen as a particular nonmonotone filter-trust-region strategy with variable memory, in the spirit of the proposals by Ke and Han [18], Xiao and Chu [26], and Toint [23].

Not surprising, all variants fail on the ill-conditioned problems `ARGLBLE` and `ARGLCLE`. The computation was terminated after 1000 iterations on `ARWHDNE` for the

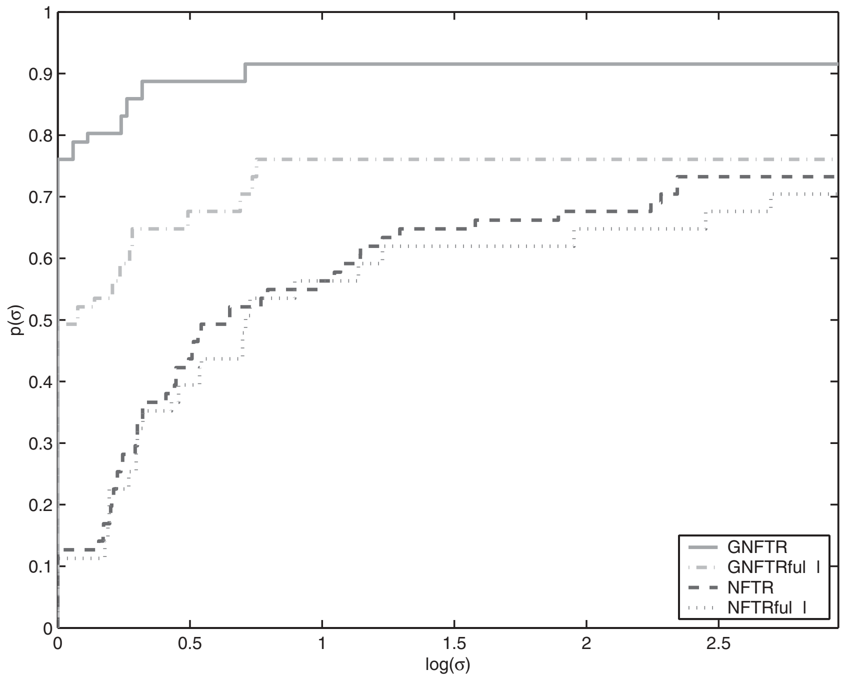

FIG. 4.4. *CPU time performance profile for GNFTR, NFTR, GNFTR-full, and NFTR-full.*

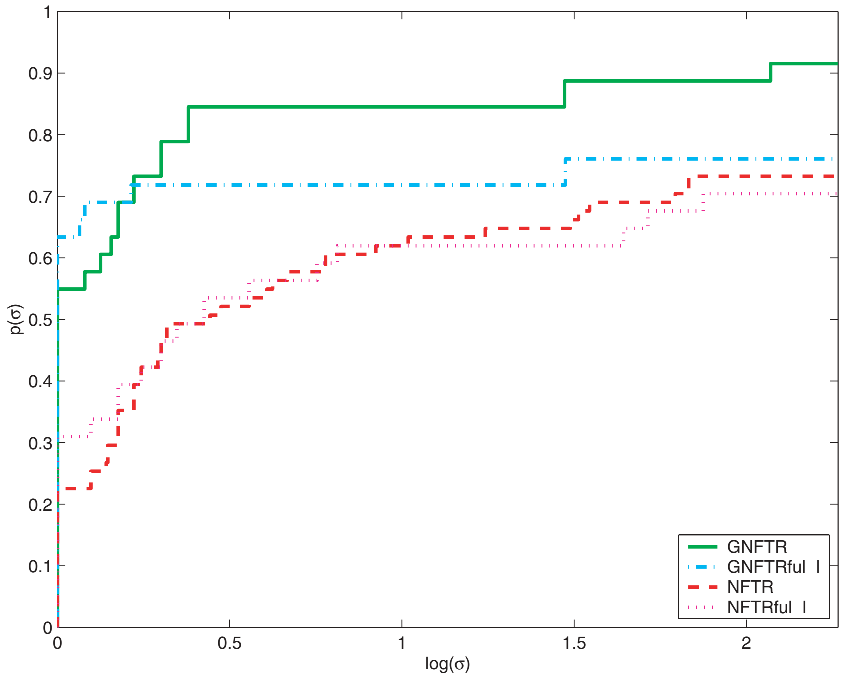

FIG. 4.5. *Iterations performance profile for GNFTR, NFTR, GNFTR-full, and NFTR-full.*

case of 10 groups, while it was terminated after 1 hour of CPU time on DRCAVTY3 for the cases of 1, 10, and $n/10$ groups.

The performance profiles in Figures 4.6 and 4.7 show that the variants differ, but to a lesser degree than in the comparisons discussed above.

For each of the three criteria, we note that a higher number of groups (a higher value of p) seems to result in better performance. This effect is mostly due to the fact that the set of unacceptable values in a higher dimensional filter space is typically smaller than for a low filter dimension, which then allows more iterates to be accepted.

It is interesting to note, in Figure 4.8, that the size of the filter tends to decrease with the number of groups, mostly because more filter entries are removed from the filter (using (2.5)) when there are fewer groups and thus fewer filter dimensions. Thus the memory requirements for storing the filter entries decrease even faster than simply implied by the reduction in the dimension of each filter entry.

We finally observe that the performance of GNFTR with a single group remains excellent, which indicates that a unidimensional filter may provide an attractive alternative to other nonmonotone trust-region methods.

We also attempted to partition the equations in different groups such that the $\theta_i(x_0)$ are balanced in size, but this strategy did not produce results significatively different from those obtained without balancing.

4.5. Comparison with NITSOL and LANCELOT-B. Finally, we attempted to compare GNFTR with two existing packages for solving nonlinear equations. We considered NITSOL (see Pernice and Walker [21]) and LANCELOT-B (see Conn, Gould, and Toint [3] and Gould, Orban, and Toint [14]), a program available in the GALAHAD library.

NITSOL uses a truncated GMRES iterative method to solve Newton's equations

$$J(x_k)s_k + c_k = 0,$$

followed by a backtracking linesearch to ensure global convergence. LANCELOT-B is a nonmonotone trust-region method. (See [23] for a description of the algorithm.) Although principally intended for the solution of nonlinear optimization problems, it can nevertheless be used to solve nonlinear systems of equations. In that case, it uses a truncated conjugate gradient method on the full Newton model (2.10), which makes it very similar in spirit to NFTR.

Comparing different software packages is always difficult. To make the exercise as fair as possible, we modified the default stopping criterion of NITSOL to reflect the factor $\sqrt{n}$ used in our new code (see (4.2)) and we set the maximum number of iterations to 1000 and the maximum allowed number of GMRES iterations to 100,000. We believe that this makes NITSOL's performance very comparable to that of our new method. However, because NITSOL requires that problems be square (i.e., $m = n$), we removed problems ARGLALE, ARGLBLE, ARGLCLE, ARWHDNE, SPMSQRT, and WOODSNE from the comparisons. We also set the maximum number of iterations to 1000 for LANCELOT-B, but we could not ensure a strictly comparable stopping criterion, as LANCELOT-B terminates when each component of the gradient is below the convergence threshold. This is slightly more restrictive than (4.2), which requires that their average is below that threshold.

NITSOL reported that no further progress was possible and stopped away from the solution for problems BROYDNBD, EIGENB, EIGENC, MSQRTA, MSQRTB, POROUS1, POROUS2, DRCAVTY3, EIGENAU, and YATP2SQ. LANCELOT-B reported the same diagnostic for POROUS1 and POROUS2, although it reduced the norm of the constraints

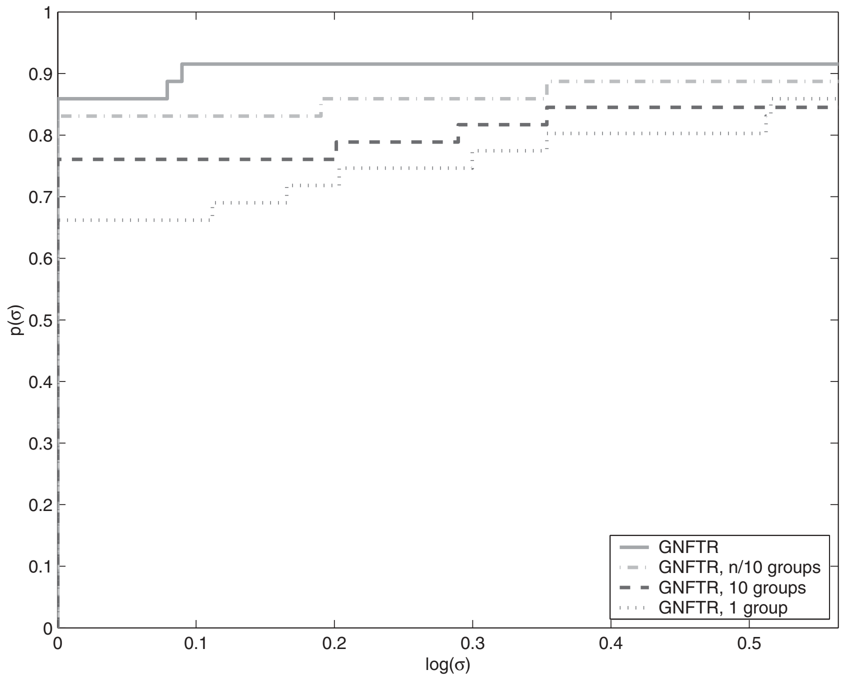

FIG. 4.6. *CPU time performance profile for GNFTR according to the number of groups.*

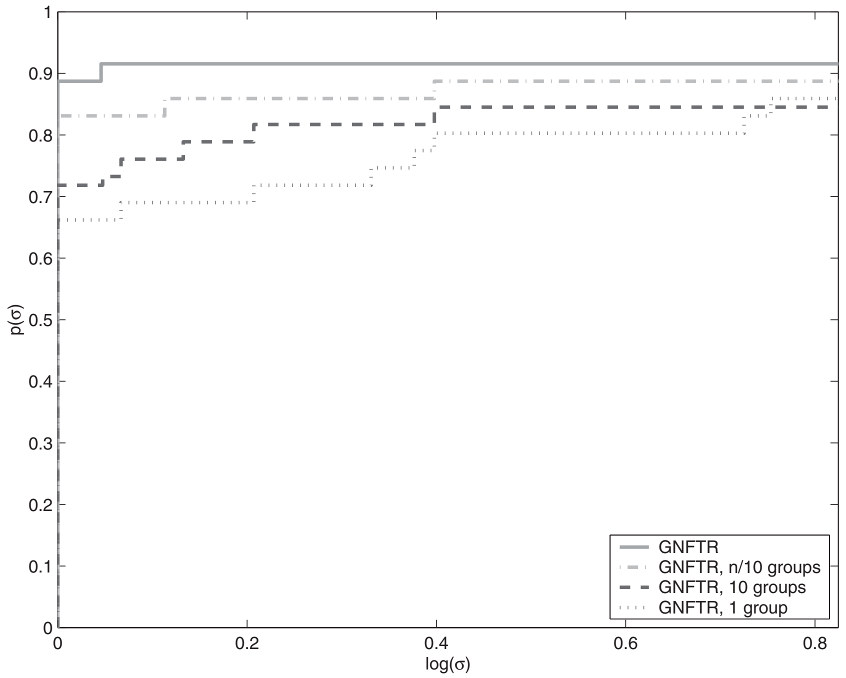

FIG. 4.7. *Iterations performance profile for GNFTR according to the number of groups.*

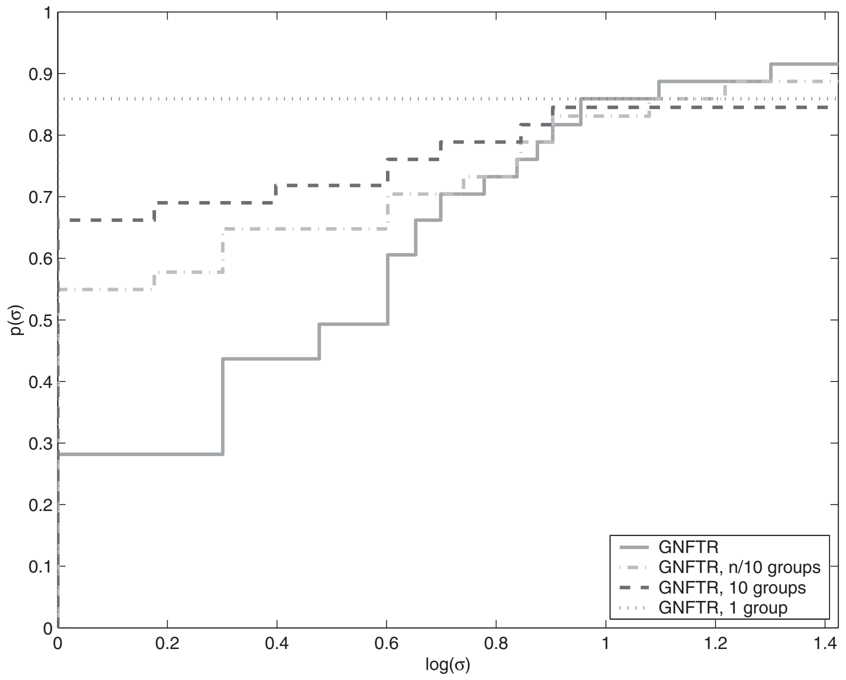

Fig. 4.8. *Maximal filter size performance profile GNFTR according to the number of groups.*

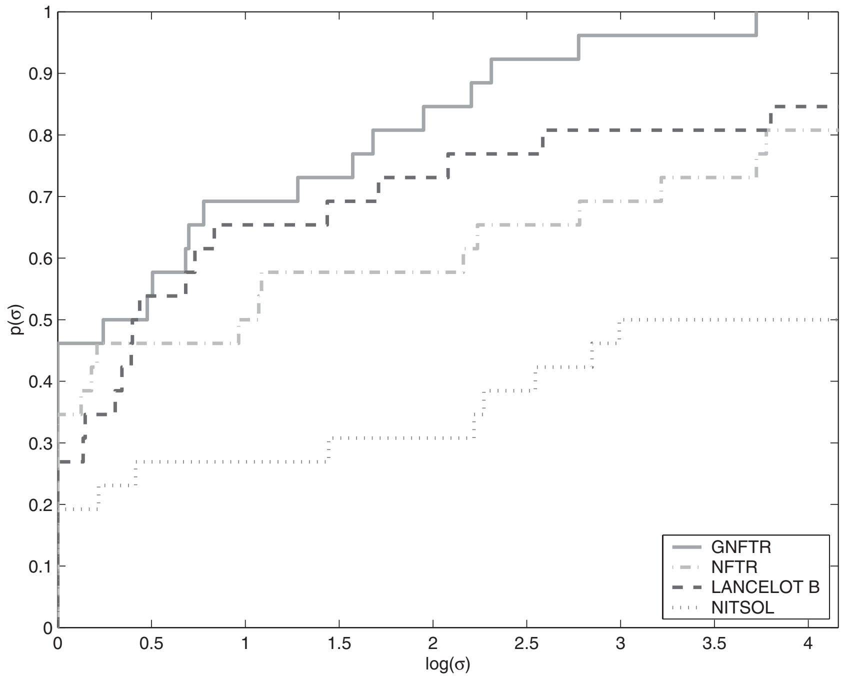

Fig. 4.9. *CPU time performance profile for GNFTR,* LANCELOT-B*, and NITSOL.*

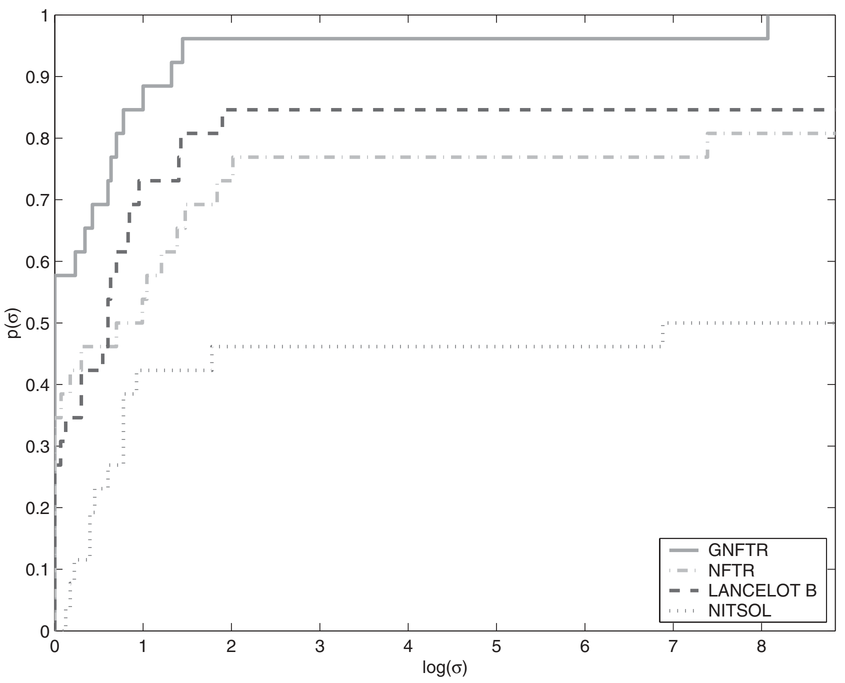

FIG. 4.10. *Iterations performance profile for GNFTR,* LANCELOT-B*, and NITSOL.*

violation to approximately 10^{-16}. The runs were terminated after 1 hour of CPU time on problems ARTIF, BRATU2D, and BRATU2DT for NITSOL and on problems ARTIF, BDVALUES, DRCAVTY1, DRCAVTY2, DRCAVTY3, and YATP2SQ for LANCELOT-B.

Figures 4.9 and 4.10 give the CPU time and iteration performance profiles for the comparison. GNFTR clearly dominates LANCELOT-B and NITSOL on our test set, both in reliability and efficiency. The comparison between NFTR and LANCELOT-B is less clear, the former dominating for small values of σ and the latter for larger ones.

5. Further comments and perspectives.

We have presented a new algorithm for nonlinear equations and nonlinear least squares that blends filter and trust-region ideas. We have proved, under reasonable assumptions, that every limit point of the sequence of iterates must be a first-order stationary point of the Euclidean norm merit function. From the preliminary numerical experience presented, it appears that its efficiency is remarkably good, both in computing time and in the number of iterations and function calls. Its reliability is globally satisfying, but further work on the algorithmic heuristics is expected to bring improvements. Of course, these conclusions depend on the test problem set used, and the authors are well aware that only continued numerical investigation can confirm the suggested gains in the longer term. A Fortran 95 module, called FILTRANE [16], has just been introduced into the GALAHAD library of optimization algorithms [14] and should facilitate a wider use of the filter-trust-region algorithm.

While the analysis discussed above is highly encouraging, it represents only the first steps in the area of research that is opened by the combination of the classical trust-region method and multidimensional filters. Numerous extensions and further developments are possible, in both theory and practice. A first idea would be to consider the values of the $c(x_k)$ themselves for inclusion in the filter, instead of their absolute values, whenever each set $\mathcal{I}_j$ contains only a single index. The filter is then no longer restrained to the positive orthant and possibly more iterates can be accepted. Interestingly, the extension of our theory to this case is straightforward. Its efficiency is under investigation. Another possibility for allowing even more iterates to be potentially acceptable is to introduce nonmonotone filter techniques similar to those discussed in Gould and Toint [15] or by modifying Algorithm 2.1 to accept x_k^+ at the next iterate if it produces a decrease in the merit function which can be judged sufficient irrespective of the step size. We could, for example, choose to accept x_k^+ whenever

$$f(x_k) - f(x_k^+) \geq \min\left(\epsilon, \kappa[f(x_k)]^\alpha\right)$$

for some positive ϵ, κ, and α. The convergence theory presented above directly extends to cover this additional condition, but its practical use remains to be explored in detail. It may also be interesting to consider norms other than the Euclidean to construct the merit function $f(x)$, the ℓ_1 and ℓ_∞ norms being obvious candidates. In this case, the subproblems consist of (approximately) solving linear programs, which, although feasible in principle, requires further research to define suitable subproblem truncation procedures.

The possibility of handling linear or nonlinear inequalities, in conjunction with equalities or alone, is also a natural development. Its theoretical aspects directly result from the present paper, since it is enough to redefine

$$\theta_j(x) = \| \, [c_{\mathcal{I}_j}(x)]_+ \, \|,$$

where the symbol $[c_k(x)]_+$ simply measures the violation of the kth constraint at x. The analysis of section 3 applies without any modification to this more general case. Practical aspects of this development are described in [16].

Finally, several strategies come to mind for coping with the (so far hypothetical) case where the memory requirements for the storage of filter entries are excessive. The first, obviously, is to use out-of-core storage. The second is to impose an upper bound on the number of filter entries and to fall back on the pure trust-region algorithm if this maximum is reached. A more sophisticated technique would be to reduce the filter space dimension by decreasing the number of equation subsets. For instance, if equations of index $i \in \mathcal{M}$ are to be merged into a new subset, the components of each filter entry whose index is $\mathcal{M}$ at iteration k, i.e., $\{\theta_{i,k}\}_{i\in\mathcal{M}}$, can be replaced by the single component

$$\sqrt{\sum_{i\in\mathcal{M}} \theta_{i,k}^2}.$$

The discussion above then suggests that the reduction in memory requirement could exceed $|\mathcal{F}| \times (|\mathcal{M}| - 1)$, as the merging process may create strongly dominated filter entries, which can then be removed. It also suggests that the deterioration in performance would remain moderate. One might also consider selectively removing filter entries that are not strongly dominated, but this last idea needs to be investigated in more detail as it is not compatible with our current proof of global convergence.

An adequate definition of equation subsets remains an interesting open question. One could, for instance, base this definition on some domain decomposition if the problem at hand arises from discretization, or one could assign different subsets to residuals of coupled equations, if the problem has this form. Many other strategies are possible and further research is necessary to identify the better ones.

We conclude by noting that the ideas presented in this paper immediately suggest other uses for the same techniques, like the introduction of multidimensional filters in nonlinear programming (where current state-of-the-art methods use only a two-dimensional filter space) or filter techniques for the solution of nonlinear equilibrium problems (where the complementarity residual is a separate filter entry).

Acknowledgments. The authors are extremely grateful to Michael Ulbrich, Luis N. Vicente, Dominique Orban, and an anonymous referee for their useful comments on an earlier version of this paper.

REFERENCES

[1] I. Bongartz, A. R. Conn, N. I. M. Gould, and Ph. L. Toint, CUTE*: Constrained and unconstrained testing environment*, ACM Trans. Math. Software, 21 (1995), pp. 123–160.

[2] C. M. Chin and R. Fletcher, *Convergence properties of SLP-filter algorithms that takes EQP steps*, Math. Program., 96 (2003), pp. 161–177.

[3] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, LANCELOT*: A Fortran Package for Large-Scale Nonlinear Optimization (Release A)*, Springer Ser. Comput. Math. 17, Springer-Verlag, Heidelberg, Berlin, New York, 1992.

[4] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *Trust Region Methods*, MPS/SIAM Ser. Optim. 1, SIAM, Philadelphia, 2000.

[5] J. E. Dennis, D. M. Gay, and R. E. Welsh, *An adaptive nonlinear least squares algorithm*, ACM Trans. Math. Software, 7 (1981), pp. 348–368.

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

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

[8] R. Fletcher and S. Leyffer, *Nonlinear programming without a penalty function*, Math. Program., 91 (2002), pp. 239–269.

[9] R. Fletcher and S. Leyffer, *Filter-type algorithms for solving systems of algebraic equations and inequalities*, in High Performance Algorithms and Software in Nonlinear Optimization, G. Di Pillo and A. Murli, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 2003, pp. 259–278.

[10] R. Fletcher, S. Leyffer, and Ph. L. Toint, *On the global convergence of a filter—SQP algorithm*, SIAM J. Optim., 13 (2002), pp. 44–59.

[11] C. C. Gonzaga, E. Karas, and M. Vanti, *A globally convergent filter method for nonlinear programming*, SIAM J. Optim., 14 (2003), pp. 646–669.

[12] N. I. M. Gould, S. Lucidi, M. Roma, and Ph. L. Toint, *Solving the trust-region subproblem using the Lanczos method*, SIAM J. Optim., 9 (1999), pp. 504–525.

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

[14] N. I. M. Gould, D. Orban, and Ph. L. Toint, GALAHAD*—a library of thread-safe Fortran* 90 *packages for large-scale nonlinear optimization*, ACM Trans. Math. Software, 29 (2003), pp. 353–372.

[15] N. I. M. Gould and Ph. L. Toint, *Global Convergence of a Non-monotone Trust-Region Filter Algorithm for Nonlinear Programming*, Tech. Report 02/15, Department of Mathematics, University of Namur, Namur, Belgium, 2002.

[16] N. I. M. Gould and Ph. L. Toint, FILTRANE*: A Fortran* 95 *Filter-Trust-Region Package for Solving Systems of Nonlinear Equalities, Nonlinear Inequalities and Nonlinear Least-Squares Problems*, Tech. Report 03/15, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2003.

[17] N. I. M. Gould and Ph. L. Toint, *Global convergence of a hybrid trust-region SQP-filter algorithm for general nonlinear programming*, in System Modeling and Optimization XX, E. Sachs and R. Tichatschke, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 2003, pp. 23–54.

[18] X. Ke and J. Han, *A class of nonmonotone trust region algorithms for constrained optimizations*, Chinese Sci. Bull., 40 (1995), pp. 1321–1324.

[19] L. Lukšan, *Hybrid methods for large sparse nonlinear least-squares*, J. Optim. Theory Appl., 89 (1996), pp. 575–595.

[20] J. J. Moré and D. C. Sorensen, *Computing a trust region step*, SIAM J. Sci. Statist. Comput., 4 (1983), pp. 553–572.

[21] M. Pernice and H. F. Walker, *NITSOL: A Newton iterative solver for nonlinear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 302–318.

[22] Ph. L. Toint, *On large scale nonlinear least squares calculations*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 416–435.

[23] Ph. L. Toint, *A non-monotone trust-region algorithm for nonlinear optimization subject to convex constraints*, Math. Program., 77 (1997), pp. 69–94.

[24] M. Ulbrich, S. Ulbrich, and L. N. Vicente, *A globally convergent primal-dual interior point filter method for nonconvex nonlinear programming*, Math. Program. Ser. A, 100 (2004), pp. 379–410.

[25] A. Wächter and L. T. Biegler, *Global and Local Convergence of Line Search Filter Methods for Nonlinear Programming*, Tech. Report CAPD B-01-09, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, 2001.

[26] Y. Xiao and E. K. W. Chu, *Nonmonotone Trust Region Methods*, Tech. Report 95/17, Monash University, Clayton, Australia, 1995.