A filter-trust-region method for unconstrained optimization

by N.I.M. Gould[1], C. Sainvitu[2] and Ph. L. Toint[2]

Report 04/03 February 5, 2004

[1] Computational Science and Engineering Department
Rutherford Appleton Laboratory,
Chilton, Oxfordshire, England
Email: gould@rl.ac.uk

[2] Department of Mathematics,
University of Namur,
61, rue de Bruxelles, B-5000 Namur, Belgium,
Email: caroline.sainvitu@fundp.ac.be
philippe.toint@fundp.ac.be

A filter-trust-region method for unconstrained optimization

Nick I. M. Gould, Caroline Sainvitu and
Philippe L. Toint

5 February 2004

Abstract

A new filter-trust-region algorithm for solving unconstrained nonlinear optimization problems is introduced. Based on the filter technique introduced by Fletcher and Leyffer, it extends an existing technique of Gould, Leyffer and Toint (SIAM J. Optim., to appear 2004) for nonlinear equations and nonlinear least-squares to the fully general unconstrained optimization problem. The new algorithm is shown to be globally convergent to at least one second-order critical point, and numerical experiments indicate that it is very competitive with more classical trust-region algorithms.

1 Introduction

Since filter methods have been introduced for constrained nonlinear optimization by Fletcher and Leyffer [5], they have enjoyed considerable interest in their original domain of application [1, 4, 6, 7, 16, 17]. More recently, they have been extended by Gould, Leyffer and Toint [8, 12] to the nonlinear feasibility problem (including nonlinear equations and nonlinear least-squares), which is to minimize the norm of the violations of a set of (possibly nonlinear and/or nonconvex) constraints. Since nonlinear least-squares can be seen as a specialized case of unconstrained optimization, it is natural to consider the further extension of the filter techniques to general unconstrained optimization problems: this is the object of the present paper.

The presentation is organized as follows. Section 2 introduces the problem and the new algorithm, whose global convergence to points satisfying second-order optimality conditions is shown in Section 3.1. The results of numerical experience with the new method are discussed in Section 4 and some conclusions and perspectives are finally presented in Section 5.

2 The problem and the new algorithm

We consider the following unconstrained minimization problem

$$\min_{x \in \mathbb{R}^n} \; f(x), \tag{2.1}$$

where f is a twice continuously differentiable function of the variables $x \in \mathbb{R}^n$. An efficient technique for solving this problem is to use Newton's method, which, from a current iterate x_k, computes a trial step s_k by minimizing a model of the objective function consisting of the first three terms of its Taylor's expansion around x_k, yielding a *trial point*

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

Unfortunately, it is well-known that such an algorithm may not always be well-defined (when the Taylor's model is nonconvex), or convergent from any initial point x_0. These difficulties can be circumvented by restricting the model minimization to a *trust region* containing x_k, in a manner that is now well established (see Conn, Gould and Toint [2] for an extensive description of trust-region methods and their properties). We propose to further extend such methods by introducing a multidimensional filter technique, whose aim is to encourage convergence to first-order critical points by driving every component of the objective's gradient

$$\nabla_x f(x) \stackrel{\text{def}}{=} g(x) = (g_1(x), \ldots, g_n(x))^T$$

to zero.

2.1 Computing a trial point

Before indicating how to apply our filter technique, we start by describing how to compute the trial point $x_k^+ = x_k + s_k$ from a current iterate x_k. At each iteration, we define the model of the objective function to be

$$m_k(x_k + s) = f(x_k) + g_k^T s + \frac{1}{2} s^T H_k s,$$

where H_k is a symmetric approximation to $\nabla_{xx} f(x_k)$, and consider a *trust-region* centered at x_k

$$\mathcal{B}_k = \{x_k + s \mid \|s\| \le \Delta_k\},$$

where we believe this model to be adequate. A trial step s_k is then computed by minimizing the model (possibly only approximately). At variance with classical trust-region methods, we do not require here that

$$\|s_k\| \le \Delta_k \tag{2.2}$$

at every iteration of our algorithm. The convergence analysis that follows requires, as is common in trust-region methods [2, Chapter 6], that this step provides, at iteration k, a *sufficient decrease on the model*, which is to say that

$$m_k(x_k) - m_k(x_k + s_k) \ge \kappa_{\text{mdc}} \max \left[\|g_k\| \min\left[\frac{\|g_k\|}{\beta_k}, \Delta_k\right], |\tau_k| \min[\tau_k^2, \Delta_k^2] \right] \tag{2.3}$$

where $\kappa_{\rm mdc}$ is a constant in $(0,1)$, β_k is a positive upper bound on the norm of the Hessian of the model m_k, i.e.

$$\beta_k \stackrel{\rm def}{=} 1 + \max_{x\in\mathcal{B}_k} \|\nabla_{xx} m_k(x_k)\|$$

and $\tau_k = \min[0, \lambda_{\min}[H_k]]$. Although this condition seems technical, there are efficient numerical methods to compute s_k that guarantee that it holds (see [9, 13], or, more generally, [2, Chapter 7]). Typical trust-region algorithms then evaluate the objective function at the trial point and accept x_k^+ as the new iterate if the reduction achieved in the objective function is at least a fraction of that predicted by the model. The trust-region radius Δ_k is also possibly enlarged if this is the case, or it is reduced if the achieved reduction is too small.

2.2 The multidimensional filter

We now consider using a filter mechanism to potentially accept x_k^+ as the new iterate more often. The notion of filter is based on that of *dominance*: for our problem, we say that a point x_1 *dominates* a point x_2 whenever

$$|g_i(x_1)| \leq |g_i(x_2)| \quad \text{for all} \quad i = 1, \ldots, n.$$

Thus, if iterate x_1 dominates iterate x_2 and if we focus our attention on convergence to first-order critical points only, the latter is of no real interest to us since x_1 is at least as good as x_2 for each of the components of the gradient. All we need to do is to remember iterates that are not dominated by other iterates by using a structure called a *filter*. We define a *multidimensional filter* $\mathcal{F}$ as a list of n-tuples of the form $(g_{k,1}, \ldots, g_{k,n})$, where $g_{k,i} \stackrel{\rm def}{=} g_i(x_k)$, such that, if g_k and g_ℓ belong to $\mathcal{F}$, then

$$|g_{k,j}| < |g_{\ell,j}| \quad \text{for at least one} \quad j \in \{1, \ldots, n\}. \tag{2.4}$$

Filter methods propose to accept a new trial iterate x_k^+ if it is not dominated by any other iterate in the filter.

However, we do not wish to accept a new point x_k^+ if one of the components of $g(x_k^+)$ is arbitrarily close to being dominated by another point already in the filter. In order to avoid this situation, we slightly strengthen our acceptability test and we say that a new trial point x_k^+ is *acceptable for the filter* $\mathcal{F}$ if and only if

$$\forall g_l \in \mathcal{F} \qquad \exists\, j \in \{1, \ldots, n\} \;:\; \quad |g_j(x_k^+)| \leq |g_{j,l}| - \gamma_g \|g_l\|, \tag{2.5}$$

where $\gamma_g \in (0, 1/\sqrt{n})$ is a small positive constant. If an iterate x_k is acceptable in the sense of (2.5), we may wish to add it to the filter and remove from it every $g_\ell \in \mathcal{F}$ such that $|g_{j,\ell}| > |g_{j,k}|$ for all $j \in \{1, \ldots, n\}$.

If the mechanism described so far is adequate for convex problems (where a zero gradient is both necessary and sufficient for second-order criticality), it may be unsuitable for nonconvex ones. Indeed it might prevent progress away from a saddle point, in which case an increase in the gradient components is acceptable. We therefore modify the filter mechanism to ensure that the filter is reset to the empty set after each iteration giving sufficient descent on the objective function at which the model m_k was detected to be nonconvex, and set an upper bound on the acceptable objective function values to ensure that the obtained decrease is permanent.

We are now able to combine these ideas into an algorithm, whose main objective is to let the filter play the major role in ensuring global convergence within "convex basins", falling back on the usual trust-region method only if things do not go well or if negative curvature is encountered.

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. The constants $\gamma_g \in (0, 1/\sqrt{n})$, η_1, η_2, γ_1, γ_2 and γ_3 are also given and satisfy

$$0 < \eta_1 \leq \eta_2 < 1 \quad \text{and} \quad 0 < \gamma_1 \leq \gamma_2 < 1 \leq \gamma_3. \tag{2.6}$$

Compute $f(x_0)$ and $g(x_0)$, set $k = 0$. Initialize the filter $\mathcal{F}$ to the empty set and choose $f_{\sup} \geq f(x_0)$. Define two flags `RESTRICT` and `NONCONVEX`, the former to be unset.

Step 1: Determine a trial step.

Compute a finite step s_k that "sufficiently reduces" the model m_k, i.e. that satisfies (2.3) and that also satisfies $\|s_k\| \leq \Delta_k$ if `RESTRICT` is set or if m_k is nonconvex. In the latter case, set `NONCONVEX`; otherwise unset it. Compute the trial point $x_k^+ = x_k + s_k$.

Step 2: Compute $f(x_k^+)$ and define the following ratio

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

If $f(x_k^+) > f_{\sup}$, set $x_{k+1} = x_k$, set `RESTRICT` and go to Step 4.

Step 3: Test to accept the trial step.

- Compute $g_k^+ = g(x_k^+)$.
- If x_k^+ is acceptable for the filter $\mathcal{F}$ and `NONCONVEX` is unset:
  Set $x_{k+1} = x_k^+$, unset `RESTRICT` and add g_k^+ to the filter $\mathcal{F}$ if either $\rho_k < \eta_1$ or $\|s_k\| > \Delta_k$.
- If x_k^+ is not acceptable for the filter $\mathcal{F}$ or `NONCONVEX` is set:
  If $\rho_k \geq \eta_1$ and $\|s_k\| \leq \Delta_k$, then

  set $x_{k+1} = x_k^+$, unset `RESTRICT` and if `NONCONVEX` is set, set $f_{\sup} = f(x_{k+1})$ and reinitialize the filter $\mathcal{F}$ to the empty set;

  else set $x_{k+1} = x_k$ and set `RESTRICT`.

Step 4: Update the trust-region radius.

If $\|s_k\| \leq \Delta_k$, update the trust-region radius by choosing

$$\Delta_{k+1} \in \begin{cases} [\gamma_1 \Delta_k, \gamma_2 \Delta_k] & \text{if} \quad \rho_k < \eta_1, \\ [\gamma_2 \Delta_k, \Delta_k] & \text{if} \quad \rho_k \in [\eta_1, \eta_2), \\ [\Delta_k, \gamma_3 \Delta_k] & \text{if} \quad \rho_k \geq \eta_2; \end{cases} \tag{2.7}$$

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

Note that, as stated, our algorithm lacks a formal stopping criterion. In practice, one would obviously stop the calculation if $\|g_k\|$ falls below some user-defined tolerance and `NONCONVEX` is unset, or if some fixed maximum number of iterations is exceeded. Also note that our conditions on the step might impose to recompute s_k within the trust region if negative curvature was discovered for the model only after computing a step beyond the trust-region boundary. Fortunately, this is typically a very cheap calculation and can be achieved by backtracking [14] or by other suitable restriction techniques [9].

3 Global convergence

Global convergence properties of Algorithm 2.1 will be proved under the following assumptions.

A1 f is twice continuously differentiable on $\mathbb{R}^n$.

A2 The iterates x_k remain in a closed, bounded domain of $\mathbb{R}^n$.

A3 For all k, the model m_k is twice differentiable on $\mathbb{R}^n$ and has a uniformly bounded Hessian.

A4 For all k, $m_k(x_k) = f(x_k)$ and $g_k = \nabla_x m_k(x_k) = \nabla_x f(x_k)$.

Note that A1, A2 and A3 together imply that there exist constants $\kappa_{\rm l}$, $\kappa_{\rm u} \geq \kappa_{\rm l}$, $\kappa_{\rm ufh} \geq 1$ and $\kappa_{\rm umh} \geq 1$ such that

$$f(x_k) \in [\kappa_{\rm l}, \kappa_{\rm u}], \quad \|\nabla_{xx} f(x_k)\| \leq \kappa_{\rm ufh} \quad \text{and} \quad \|H_k\| \leq \kappa_{\rm umh} - 1 \tag{3.8}$$

for all k. Combining this with the definition of β_k, we have that

$$\beta_k \leq \kappa_{\rm umh} \tag{3.9}$$

for all k and all x in the convex hull of $\{x_k\}$. For the purpose of our analysis, we shall consider

$$\mathcal{S} = \{k \mid x_{k+1} = x_k + s_k\},$$

the set of *successful iterations*,

$$\mathcal{A} = \{k \mid x_k^+ \text{ is added to the filter } \},$$

the set of *filter iterations*,

$$\mathcal{D} = \{k \mid \rho_k \geq \eta_1\},$$

the set of *sufficient descent iterations*, and

$$\mathcal{N} = \{k \mid \texttt{NONCONVEX} \text{ is set } \},$$

the set of *nonconvex iterations*. Observe that $\mathcal{A} \subseteq \mathcal{S}$ and

$$\mathcal{S} \cap \mathcal{N} = \mathcal{D} \cap \mathcal{N}. \tag{3.10}$$

We conclude this section by stating a crucial property of the algorithm.

Lemma 3.1 *We have that, for all $k \geq 0$,*

$$f(x_0) - f(x_{k+1}) \geq \sum_{\substack{j=0 \\ j \in \mathcal{S} \cap \mathcal{N}}}^{k} [f(x_j) - f(x_{j+1})]. \tag{3.11}$$

Proof. Denoting $\mathcal{S} \cap \mathcal{N} = \{k_i\}$, we observe that the mechanism of the algorithm ensures that

$$f(x_{k_{i+1}}) \leq f(x_\ell) < f(x_{k_i})$$

for all i and all $k_i + 1 \leq \ell \leq k_{i+1}$. This directly implies the desired inequality. □

3.1 Convergence to Critical Points

We first prove the convergence of our algorithm to first-order critical points.
Our first step is to prove that, as long as a first-order critical point is not approached, we do not have infinitely many successful nonconvex iterations in the course of the algorithm. We start by recalling three results from [2] in order to show that the trust-region radius is bounded away from zero in this case.

Lemma 3.2 *Suppose that A1-A4 hold and that $\|s_k\| \leq \Delta_k$. Then we have that*

$$|f(x_k + s_k) - m_k(x_k + s_k)| \leq \kappa_{\text{ubh}} \Delta_k^2, \tag{3.12}$$

where $x_k + s_k \in \mathcal{B}_k$ and

$$\kappa_{\text{ubh}} \stackrel{\text{def}}{=} \max[\kappa_{\text{ufh}}, \kappa_{\text{umh}}]. \tag{3.13}$$

The proof is identical to that of Theorem 6.4.1 in [2] but we now need to make the additional assumption that $\|s_k\| \leq \Delta_k$ explicit (instead of being implicit, in this reference, in the definition of a trust-region step).

We now show that the trust-region radius must increase if the current iterate is not first-order critical and the trust-region radius is small enough.

Lemma 3.3 *Suppose that A1-A4 hold and that $\|s_k\| \le \Delta_k$. Suppose furthermore that $g_k \neq 0$ and that*

$$\Delta_k \le \frac{\kappa_{mdc}\|g_k\|(1-\eta_2)}{\kappa_{ubh}}. \tag{3.14}$$

Then iteration $\rho_k \ge \eta_2$ and

$$\Delta_{k+1} \ge \Delta_k. \tag{3.15}$$

The proof is the same as Theorem 6.4.2 in [2] when $\|s_k\| \le \Delta_k$, while (3.15) follows from Step 4 when $\|s_k\| > \Delta_k$ as then $\Delta_{k+1} = \Delta_k$. As a consequence, we obtain that the radius cannot become too small as long as a first-order critical point is not approached.

Lemma 3.4 *Suppose that A1-A4 hold and that there exists a constant $\kappa_{lbg} > 0$ such that $\|g_k\| \ge \kappa_{lbg}$ for all k. Then there is a constant $\kappa_{lbd} > 0$ such that*

$$\Delta_k \ge \kappa_{lbd} \tag{3.16}$$

for all k.

Proof. Assume that iteration k is the first such that

$$\Delta_{k+1} \le \frac{\gamma_1 \kappa_{\rm mdc}\, \kappa_{\rm lbg}(1-\eta_2)}{\kappa_{\rm ubh}}. \tag{3.17}$$

This means that the trust-region radius has been decreased at iteration k, which in turn implies, from the condition in Step 4 of the algorithm, that $\|s_k\| \le \Delta_k$. We also have that $\gamma_1\Delta_k \le \Delta_{k+1}$, and hence that

$$\Delta_k \le \frac{\kappa_{\rm mdc}\, \kappa_{\rm lbg}(1-\eta_2)}{\kappa_{\rm ubh}}.$$

Our assumption on the norm of the gradient then implies that (3.14) holds. This and the fact that $\|s_k\| \le \Delta_k$ thus give that (3.15) is satisfied. But this contradicts the fact that iteration k is the first such that (3.17) holds, and our initial assumption is therefore impossible. This yields the desired conclusion with

$$\kappa_{\rm lbd} = \frac{\gamma_1 \kappa_{\rm mdc}\, \kappa_{\rm lbg}(1-\eta_2)}{\kappa_{\rm ubh}}.$$

□

We now prove the crucial result that the number of successful nonconvex iterations must be finite unless a first-order critical point is approached.

Theorem 3.5 *Suppose that A1-A4 hold and that there exists a constant $\kappa_{lbg} > 0$ such that $\|g_k\| \ge \kappa_{lbg}$ for all k. Then there can only be finitely many successful nonconvex iterations in the course of the algorithm, i.e. $|\mathcal{S} \cap \mathcal{N}| < +\infty$.*

Proof. Suppose, for the purpose of obtaining a contradiction, that there are infinitely many successful nonconvex iterations, which we index by $\mathcal{S} \cap \mathcal{N} = \{k_i\}$. It follows from (3.10) that the algorithm also guarantees that $\rho_k \geq \eta_1$ for all iterations in $\mathcal{S} \cap \mathcal{N}$, which in turn implies, with (2.3), that, for $k \in \mathcal{S} \cap \mathcal{N}$,

$$\begin{aligned} f(x_k) - f(x_{k+1}) &\geq \eta_1 [m_k(x_k) - m_k(x_k + s_k)] \\ &\geq \eta_1 \, \kappa_{\mathrm{mdc}} \|g_k\| \min \left[\frac{\|g_k\|}{\beta_k}, \Delta_k \right] \\ &\geq \eta_1 \, \kappa_{\mathrm{mdc}} \, \kappa_{\mathrm{lbg}} \min \left[\frac{\kappa_{\mathrm{lbg}}}{\kappa_{\mathrm{umh}}}, \kappa_{\mathrm{lbd}} \right], \end{aligned}$$

where we have used the Lemma 3.4, (3.9) and our lower bound on the gradient norm to obtain the last inequality. Combining now this bound with (3.11), we deduce that

$$f(x_0) - f(x_{k+1}) \geq \sum_{\substack{j=0 \\ j \in \mathcal{S} \cap \mathcal{N}}}^{k} [f(x_j) - f(x_{j+1})] \geq \varsigma_k \, \eta_1 \, \kappa_{\mathrm{mdc}} \, \kappa_{\mathrm{lbg}} \min \left[\frac{\kappa_{\mathrm{lbg}}}{\kappa_{\mathrm{umh}}}, \kappa_{\mathrm{lbd}} \right],$$

where $\varsigma_k = |\{1, \ldots, k\} \cap \mathcal{S} \cap \mathcal{N}|$. As we have supposed that there are infinitely many successful nonconvex iterations, we have that

$$\lim_{k \to \infty} \varsigma_k = +\infty,$$

and $[f(x_0) - f(x_{k+1})]$ is unbounded above, which contradicts the fact that the objective function is bounded below, as stated in (3.8). Our initial assumption must then be false, and the set $\mathcal{S} \cap \mathcal{N}$ of successful nonconvex iterations must be finite. $\square$

We now establish the criticality of the limit point of the sequence of iterates when there are only finitely many successful iterations.

Theorem 3.6 *Suppose that A1-A4 and (2.3) hold and that there are only finitely many successful iterations, i.e.* $|\mathcal{S}| < +\infty$*. Then* $x_k = x^*$ *for all sufficiently large* k*, and* x^* *is first-order critical.*

Proof. Let k_0 be the index of the last successful iterate. Then $x^* = x_{k_0+1} = x_{k_0+j}$ and

$$\rho_{k_0+j} < \eta_1 \text{ for all } j > 0. \tag{3.18}$$

Now observe that `RESTRICT` is set by the algorithm in the course of every unsuccessful iteration. This flag must thus be set at the beginning of every iteration of index $k_0 + j + 1$ for $j > 0$. As a consequence, $\|s_{k_0+j+2}\| \leq \Delta_{k_0+j+2}$ for all $j > 0$. This, (3.18) and the mechanism of Step 4 of the algorithm then imply that

$$\lim_{k \to \infty} \Delta_k = 0. \tag{3.19}$$

Assume now, for the purpose of establishing a contradiction, that $\|g_{k_0+1}\| \geq \varepsilon$ for some $\varepsilon > 0$. Then Lemma 3.4 implies that (3.19) is impossible and we deduce that

$$\|g_{k_0+j}\| = 0$$

for all $j > 0$. $\square$

Having proved the desired convergence property for the case where $\mathcal{S}$ is finite, we restrict our attention, for the rest of this section, to the case where there are infinitely many successful iterations, i.e. $|\mathcal{S}| = +\infty$. We first investigate what happens if infinitely many values are added to the filter in the course of the algorithm.

Theorem 3.7 *Suppose that A1-A4 hold and that* $|\mathcal{A}| = |\mathcal{S}| = +\infty$. *Then*

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{3.20}$$

Proof. Assume, for the purpose of obtaining a contradiction, that, for all k large enough,

$$\|g_k\| \geq \kappa_{\rm lbg} \tag{3.21}$$

for some $\kappa_{\rm lbg} > 0$. This bound and Theorem 3.5 then imply that $|\mathcal{S} \cap \mathcal{N}|$ is finite and therefore that the filter is no longer reset to the empty set for k sufficiently large. Moreover, since our assumptions imply that $\{\|g_{k_{i+1}}\|\}$ is bounded above and below, there must exist a subsequence $\{k_\ell\} \subseteq \{k_{i+1}\}$ where $\{k_i\} = \mathcal{A}$ such that

$$\lim_{\ell\to\infty} g_{k_\ell} = g_\infty \qquad \text{with} \quad \|g_\infty\| \geq \kappa_{\rm lbg}. \tag{3.22}$$

By definition of $\{k_\ell\}$, x_{k_ℓ} is acceptable for the filter for every ℓ, which implies, since the filter is not reset for ℓ large enough, that, for each ℓ sufficiently large, there exists an index $j \in \{1, \ldots, n\}$ such that

$$|g_{k_\ell,j}| - |g_{k_{\ell-1},j}| < -\gamma_g \|g_{k_{\ell-1}}\|. \tag{3.23}$$

But (3.21) implies that $\|g_{k_{\ell-1}}\| \geq \kappa_{\rm lbg}$ for all ℓ sufficiently large. Hence we deduce from (3.23) that

$$|g_{k_\ell,j}| - |g_{k_{\ell-1},j}| < -\gamma_g \kappa_{\rm lbg}$$

for all ℓ sufficiently large. But the left-hand side of this inequality tends to zero when ℓ tends to infinity because of (3.22), yielding the desired contradiction. Hence (3.20) holds. $\square$

Consider now the case where the number of iterates added to the filter in the course of the algorithm is finite.

Theorem 3.8 *Suppose that A1-A4 hold and that* $|\mathcal{S}| = +\infty$ *but* $|\mathcal{A}| < +\infty$. *Then (3.20) holds.*

Proof. Assume, again for the purpose of obtaining a contradiction, that (3.21) holds for all k large enough and for some $\kappa_{\rm lbg} > 0$. The finiteness of $|\mathcal{A}|$ then implies that $\rho_k \geq \eta_1$ and that $\|s_k\| \leq \Delta_k$ for all $k \in \mathcal{S}$ sufficiently large. If we define $\bar{\varsigma}_{p,k} = |\{p, \ldots, k\} \cap \mathcal{S}|$, we then obtain that

$$f(x_p) - f(x_{k+1}) = \sum_{\substack{j=p \\ j\in\mathcal{S}}}^{k} [f(x_j) - f(x_{j+1})] \geq \bar{\varsigma}_{p,k}\, \eta_1\, \kappa_{\rm mdc}\, \kappa_{\rm lbg} \min\left[\frac{\kappa_{\rm lbg}}{\kappa_{\rm umh}}, \kappa_{\rm lbd}\right],$$

for p and k sufficiently large, where, as above, we used (2.3), (3.9) and (3.21) to derive the inequality. But $\bar{\varsigma}_{p,k}$ tends to infinity with k for a fixed p sufficiently large since $|\mathcal{S}|$ is infinite, and we again derive a contradiction from the fact that $f(x_{k+1})$ then becomes unbounded below. The limit (3.20) then follows. $\square$

By the two last theorems, we have that at least one of the limit points of the sequence of iterates generated by the algorithm satisfies the first-order necessary condition. As the following example shows, this cannot be improved without modifying the algorithm.

Example 3.1 Consider the objective function

$$f(x) = x^3(3x-4),$$

which has a (degenerate[(1)]) critical point at $x = 0$, and its global minimizer at $x = 1$. We will show that it is possible for Algorithm 2.1 to construct iterates for which $x_{2k} = -\frac{1}{2}^k$ and $x_{2k+1} = \frac{5}{4}$ for $k = 0, 1, 2, \ldots$; clearly there are two limit points, $x_*^{\mathrm{L}} = 0$ and $x_*^{\mathrm{R}} = \frac{5}{4}$, but only the first is critical.

Let $\Delta_0 > 2$, and suppose that $\gamma_g < \frac{1}{2}$ and that the trust-region updating scheme (2.7) is specifically

$$\Delta_{k+1} = \begin{cases} \frac{1}{2}\Delta_k & \text{if } \rho_k < \eta_1, \\ \Delta_k & \text{if } \eta_1 \le \rho_k < \eta_2 \text{ and} \\ 2\Delta_k & \text{if } \eta_2 \le \rho_k. \end{cases} \tag{3.24}$$

Now suppose that

$$\mathcal{F} = \{f'(x_{2k})\} \equiv \{-12(1 + \tfrac{1}{2}^k)\tfrac{1}{2}^{2k}\} \text{ and } \Delta_{2k} > 2. \tag{3.25}$$

We then show that the above iteration is possible for Algorithm 2.1, and that (3.25) will persist.

Consider first $x_{2k} = -\frac{1}{2}^k$, and the convex model

$$m_{2k}(x_{2k} + s) = f(x_{2k}) + sf'(x_{2k}) + \tfrac{1}{2}s^2 h_{2k}, \text{ where } h_{2k} = -\frac{f'(x_{2k})}{\frac{5}{4} - x_{2k}} > 0.$$

Then the unconstrained global minimizer of m_{2k} is $s_{2k} = \frac{5}{4} - x_{2k}$, and s_{2k} will sufficiently reduce the model within the trust region since $\Delta_{2k} > 2 > \frac{5}{4} + (\frac{1}{2})^k$. Moreover

$$m_{2k}(x_{2k}) - m_{2k}(x_{2k} + s_{2k}) = \tfrac{1}{2}\frac{(f'(x_{2k}))^2}{h_{2k}} = \tfrac{1}{2}(\tfrac{5}{4} - x_{2k})f'(x_{2k}) \to 0$$

while

$$f(x_{2k}) - f(x_{2k} + s_{2k}) = f(x_{2k}) - f(\tfrac{5}{4}) > f(0) - f(\tfrac{5}{4}) = \tfrac{125}{256} > 0$$

and thus

$$\rho_{2k} \ge \eta_2 \tag{3.26}$$

for large enough k. The trial point $x_{2k} + s_{2k}$ is not acceptable for the filter since its gradient is $f'(\frac{5}{4}) = \frac{75}{16} \gg f'(x_{2k})$, but it is an acceptable point because the trust region

[(1)]i.e., both its first and second derivatives vanish.

bound is inactive and because of (3.26). Thus $x_{2k+1} = x_{2k} + s_{2k} = \frac{5}{4}$, while (3.24) and (3.26) ensure that $\Delta_{2k+1} = 2\Delta_{2k}$.

Now consider $x_{2k+1} = \frac{5}{4}$, and the convex model

$$m_{2k+1}(x_{2k+1} + s) = f(x_{2k+1}) + sf'(x_{2k+1}) + \tfrac{1}{2}s^2 h_{2k+1},$$

where

$$h_{2k+1} = \frac{f'(x_{2k+1})}{x_{2k+1} + \frac{1}{2}^{k+1}} > 0.$$

As before, the unconstrained global minimizer of m_{2k+1} is $s_{2k+1} = -x_{2k+1} - \frac{1}{2}^{k+1}$, and s_{2k+1} will sufficiently reduce the model within the trust region since $\Delta_{2k+1} > 4 > \frac{5}{4} + (\frac{1}{2})^k$. Although $f(x_{2k+1}) - f(x_{2k+1} + s_{2k+1}) < 0$ and hence

$$\rho_{2k+1} < 0, \tag{3.27}$$

$x_{2k+1} + s_{2k+1} = -\frac{1}{2}^{k+1}$ is acceptable for the filter since it is easy to check that

$$|f'(x_{2k+1} + s_{2k+1})| = |f'(-\tfrac{1}{2}^{k+1})| < \tfrac{1}{2}|f'(x_{2k})|.$$

Hence $x_{2k+2} = x_{2k+1} + s_{2k+1} = -\frac{1}{2}^{k+1}$. Moreoever (3.24) and (3.27) imply that $f'(x_{2k+2})$ replaces $f'(x_{2k})$ in the filter, and that $\Delta_{2k+2} = \frac{1}{2}\Delta_{2k+1} = \Delta_{2k}$, and thus that (3.25) persists.

It is unclear how to modify the algorithm to enforce the property that all limit points are first-order critical without adversely affecting its numerical behaviour. We have considered not allowing filter iterations when the ratio between the current gradient norm and the smallest gradient norm found so far exceeds some prescribed (large) constant. While such a modification does not appear to affect the results of our numerical experiments, to date we have been unable to show that the modification yields the desired conclusion. Since we believe that the likelihood of the algorithm converging to more than a single limit point is very small (as with every trust-region method we are aware of), the issue really is of mostly theoretical interest.

We thus pursue our analysis by examining convergence to second-order critical points under the assumption that there is only one limit point. As in [2], we also assume the following.

$\boxed{\textbf{A5}}$ The matrix H_k is arbitrarily close to $\nabla_{xx} f(x_k)$ whenever a first-order critical point is approached, i.e.

$$\lim_{k\to\infty} \|\nabla_{xx} f(x_k) - H_k\| = 0 \text{ whenever } \lim_{k\to\infty} \|g_k\| = 0.$$

(Notice that $h_{2k} \to 0$, and thus that A5 holds in the above example.)
We are then able to derive the following theorem.

Theorem 3.9 *Suppose that A1-A5 hold and that the complete sequence of iterates $\{x_k\}$ converge to the unique limit point x^*. Then x^* is a second-order critical point.*

Proof. We start our proof (strongly inspired by Theorem 6.6.4 of [2]) by noting that [2, Lemma 6.5.3] is valid in our context. Observe also that our previous results imply that

$$g(x^*) = 0. \tag{3.28}$$

For the purpose of deriving a contradiction, assume now that

$$\tau_* \stackrel{\text{def}}{=} \lambda_{\min}[\nabla_{xx} f(x^*)] < 0. \tag{3.29}$$

Then, using A5 and (3.28), we deduce that there exists a k_0 such that, for $k \geq k_0$,

$$\lambda_{\min}[H_k] < \tfrac{1}{2}\tau_* < 0,$$

and, consequently, that $k \in \mathcal{N}$ and

$$\|s_k\| \leq \Delta_k \tag{3.30}$$

for $k \geq k_0$. Our sufficient decrease condition (2.3) then ensures that, for $k \geq k_0$,

$$m_k(x_k) - m_k(x_k + s_k) \geq \tfrac{1}{2}\kappa_{\text{mdc}}|\tau_*| \min[\tfrac{1}{4}\tau_*^2, \Delta_k^2]. \tag{3.31}$$

Consider now the ratio of achieved versus predicted reduction ρ_k in the case where $\Delta_k \leq \frac{1}{2}|\tau_*|$. Applying [2, Lemma 6.5.3] to the complete sequence $\{x_k\}$, we deduce from (3.30) that there must exist a $k_1 \geq k_0$ and a $\delta_1 \in (0, \frac{1}{2}|\tau_*|]$ such that

$$\rho_k \geq \eta_2 \;\text{ for all }\; k \geq k_1 \;\text{ such that }\; \Delta_k \leq \delta_1.$$

As a consequence, each iteration where these two conditions hold must be very successful and the algorithm then guarantees that $\Delta_{k+1} \geq \Delta_k$. This and the inequality $\gamma_1\delta_1 < \delta_1 \leq \frac{1}{2}|\tau_*|$ in turn imply that

$$\Delta_k \geq \min[\gamma_1\delta_1, \Delta_{k_0}] \stackrel{\text{def}}{=} \delta_2 \tag{3.32}$$

for all $k \geq k_1$. For every successful iteration $k \geq k_1$, we then obtain from (3.31) that

$$f(x_k) - f(x_{k+1}) \geq \tfrac{1}{2}\eta_1\kappa_{\text{mdc}}|\tau_*| \min[\tfrac{1}{4}\tau_*^2, \delta_2^2] > 0.$$

Remembering now that $k \in \mathcal{N}$ for $k \geq k_1$ (and thus that $|\mathcal{N}| = \infty$), we obtain from (3.11) that $|\mathcal{S} \cap \mathcal{N}|$, and hence $|\mathcal{S}|$, must be finite, which in turn implies that the trust-region radius tends to zero. But this contradicts (3.32). Hence our initial assumption (3.29) must be false and the proof is complete. □

4 Numerical experiments

We now report the results obtained by running our algorithm on the set of 160 unconstrained[(2)] problems from the CUTEr collection [10]. The names of the problems with their dimensions[(3)] are detailed in Table 4.1.

(2) We excluded problem BROYDN7D because of its multiple local minima.

(3) The number of free variables.

Problem	n	Problem	n	Problem	n
AIRCRFTB	5	DQRTIC	5000	OSBORNEA	5
ALLINITU	4	EDENSCH	10000	OSBORNEB	11
ARGLINA	200	EG2	1000	PALMER1C	8
ARGLINB	200	EIGENALS	2550	PALMER1D	7
ARGLINC	200	EIGENBLS	2550	PALMER2C	8
ARWHEAD	5000	EIGENCLS	2652	PALMER3C	8
BARD	3	ENGVAL1	10000	PALMER4C	8
BDQRTIC	5000	ENGVAL2	2	PALMER5C	6
BEALE	2	ERRINROS	50	PALMER6C	8
BIGGS3	3	EXPFIT	2	PALMER7C	8
BIGGS5	5	EXTROSNB	1000	PALMER8C	8
BIGGS6	6	FMINSRF2	5625	PARKCH	15
BOX2	2	FMINSURF	49	PENALTY1	1000
BOX3	3	FREUROTH	5000	PENALTY2	200
BRKMCC	2	GENROSE	500	PENALTY3	200
BROWNAL	200	GROWTHLS	3	POWELLSG	5000
BROWNBS	2	GULF	3	POWER	100
BROWNDEN	4	HAIRY	2	QUARTC	5000
BRYBND	5000	HATFLDD	3	RAYBENDL	2046
CHAINWOO	4000	HATFLDE	3	RAYBENDS	2046
CHNROSNB	50	HEART6LS	6	ROSENBR	2
CLIFF	2	HEART8LS	8	S308	2
CLPLATEA	10100	HELIX	3	SBRYBND	500
CLPLATEB	4970	HIELOW	3	SCHMVETT	5000
CLPLATEC	4970	HILBERTA	2	SCOSINE	5000
COSINE	10000	HILBERTB	10	SCURLY10	100
CRAGGLVY	5000	HIMMELBB	2	SCURLY20	100
CUBE	2	HIMMELBF	4	SCURLY30	100
CURLY10	10000	HIMMELBG	2	SENSORS	100
CURLY20	10000	HIMMELBH	2	SINEVAL	2
CURLY30	1000	HYDC20LS	99	SINQUAD	10000
DECONVU	61	JENSMP	2	SISSER	2
DENSCHNA	2	KOWOSB	4	SNAIL	2
DENSCHNB	2	LIARWHD	5000	SPARSINE	5000
DENSCHNC	2	LMINSURF	5329	SPARSQUR	10000
DENSCHND	3	LOGHAIRY	2	SPMSRTLS	4900
DENSCHNE	3	MANCINO	100	SROSENBR	5000
DENSCHNF	2	MARATOSB	2	SSC	4900
DIXMAANA	9000	MEXHAT	2	STRATEC	10
DIXMAANB	9000	MEYER3	3	TESTQUAD	5000
DIXMAANC	9000	MINSURF	36	TOINTGOR	50
DIXMAAND	9000	MOREBV	5000	TOINTGSS	5000
DIXMAANE	9000	MSQRTALS	1024	TOINTPSP	50
DIXMAANF	9000	MSQRTBLS	1024	TOINTQOR	50
DIXMAANG	9000	NCB20	5010	TQUARTIC	5000
DIXMAANH	9000	NCB20B	5000	TRIDIA	5000
DIXMAANI	9000	NLMSURF	5329	VARDIM	200
DIXMAANJ	9000	NONCVXU2	5000	VAREIGVL	50
DIXMAANK	9000	NONCVXUN	5000	VIBRBEAM	8
DIXMAANL	9000	NONDIA	5000	WATSON	12
DIXON3DQ	10000	NONDQUAR	5000	WOODS	10000
DJTL	2	NONMSQRT	100	YFITU	3
DQDRTIC	5000	ODC	4900	ZANGWIL2	2

Table 4.1: The test problems and their dimension

In each case, the starting point supplied with the problem was used. All tests were performed in double precision on a Dell Latitude C840 portable computer (1.6 Mhz, 1 Gbyte of RAM) under Red Hat 9.0 Linux and the Lahey Fortran compiler (version L6.10a) with default options. All attempts to solve the test problems were limited to a maximum of 1000 iterations or 1 hour of CPU time. The values $\gamma_1 = 0.625$, $\gamma_2 = 0.25$, $\gamma_3 = 2$,$\eta_1 = 0.01$, $\eta_2 = 0.9$, $\Delta_0 = 1$ and

$$\gamma_g = \min\left[0.001, \frac{1}{2\sqrt{n}}\right]$$

were used.

Two particular variants were tested. The first (called default) is the algorithm as described above, where, exact first and second derivatives are used and where, at each iteration, the trial point is computed by approximately minimizing $m_k(x_k + s)$ using the Generalized Lanczos Trust-Region algorithm of [9] (without preconditioning) as implemented in the GALAHAD library [11]. This procedure is terminated at the first s for which

$$\|\nabla m_k(x_k + s)\| \leq \min\left[0.1, \sqrt{\max(\epsilon_M, \|\nabla m_k(x_k)\|)}\right] \|\nabla m_k(x_k)\|, \tag{4.33}$$

where ϵ_M is the machine precision. In addition, we choose

$$f_{\rm sup} = \min(10^6 f(x_0), f(x_0) + 1000)$$

at Step 0 of the algorithm. Based on practical experience [12], we also impose that $\|s_k\| \leq 1000\Delta_k$ at all iterations following the first one at which a restricted step was taken. Finally, the algorithm stops if

$$\|\nabla f(x_k)\| \leq 10^{-6}\sqrt{n}. \tag{4.34}$$

The second algorithmic variant is the pure trust-region version, that is the same algorithm with the exception that no trial point is ever accepted in the filter.

On the 160 problems, the default version successfully solved 144 and the pure trust-region 143. Failure always occurs because the maximal iteration count is reached before convergence is declared, with the exception of the trust-region variant failing on MEYER3 because the problem is judged to be too ill-conditioned. The filter variant is thus just as reliable(4) as the trust-region version.

Figures 4.1, 4.2 and 4.3 give the performance profiles for the two variants for iterations, cpu-time and the total amount of conjugate-gradient iterations, respectively. Performance profiles give, for every $\sigma \geq 1$, the proportion $p(\sigma)$ of test problems on which each considered algorithmic variant has a performance within a factor σ of the best (see [3] for a more complete discussion). When comparing CPU times, we also take into account inaccuracies in timing by considering run-times as indistinguishable if they differ by less than 1 second or less than 5%.

(4)The two variants consistently fail on CHAINWOO, HYDC20LS, LMINSURF, LOGHAIRY, MEYER3, NLMSURF, NONCVXU2, NONCVXUN and SCURLY10.

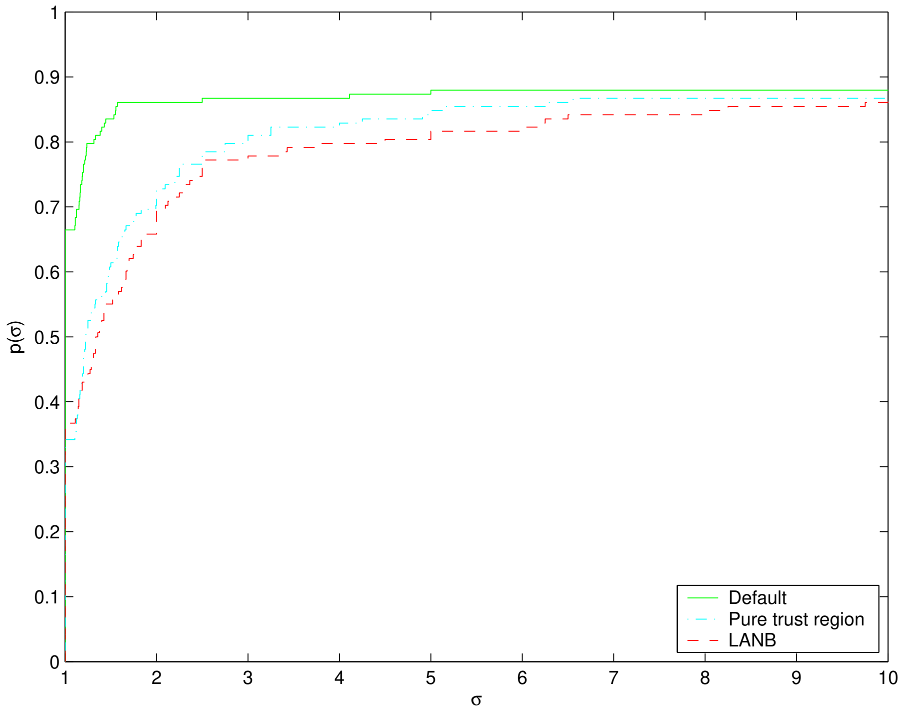

Figure 4.1: Iterations performance profiles for the two variants and LANCELOT B

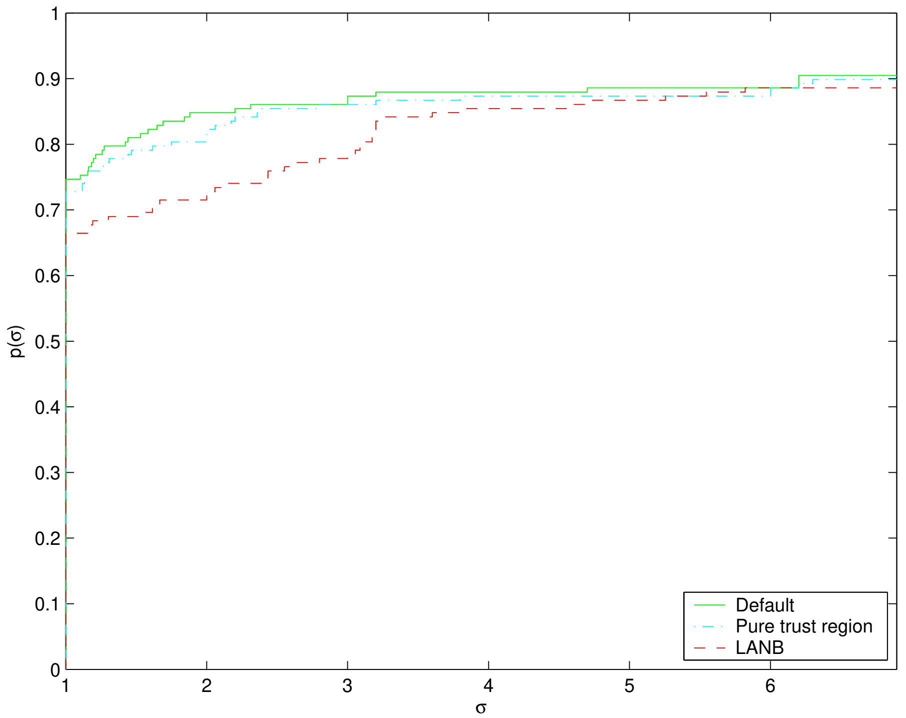

Figure 4.2: CPU performance profiles for the two variants and LANCELOT B

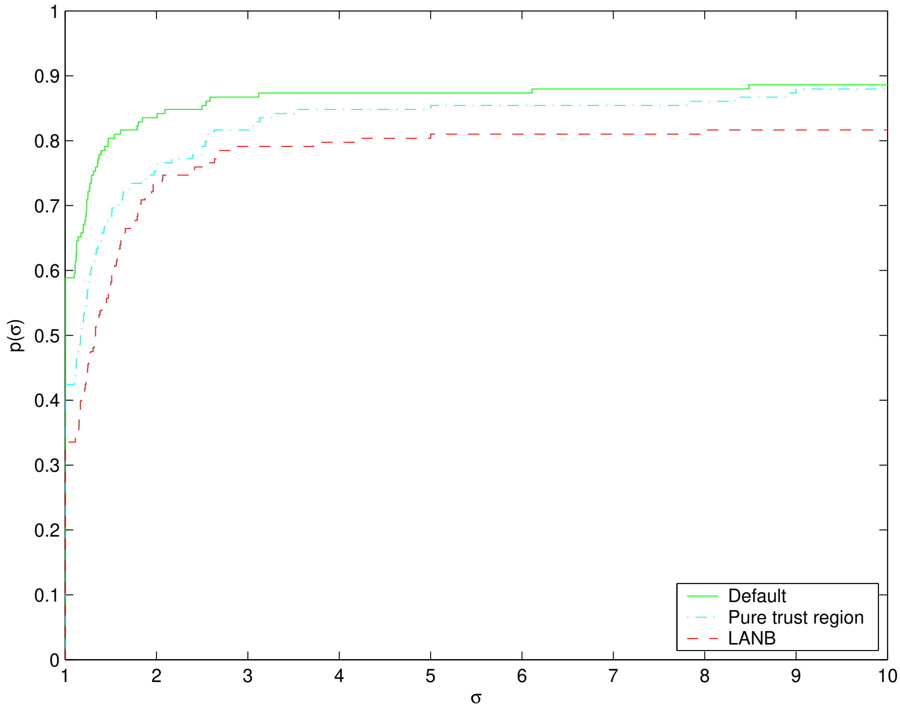

Figure 4.3: CG iterations performance profiles for the two variants and LANCELOT B

It is not difficult to see in these figures that the filter variant is significantly more efficient than the pure trust-region method in terms of the number of iterations (which is identical to the number of function and gradient evaluations). Its advantage is smaller but significant in terms of conjugate-gradients iterations, but is offset by the additional cost of the filter management operations. As a result, both variants are essentially comparable in terms of cpu-time efficiency, with a very slight advantage for the default method.

The profiles also include a comparison with LANCELOT-B, one of the GALAHAD codes [11]. This is a non-monotone trust-region algorithm (see [15] or [2, Section 10.1]), which we used unpreconditioned with $\Delta_0 = 1$ and with its other settings at their default values. Again this method, which successfully solves 141 out of 160 problems, appears to be consistently inferior to the new filter algorithm.

5 Conclusion

We have presented a filter algorithm for unconstrained optimization and have shown, under standard assumptions, that it produces at least a first-order critical point, irrespective of the chosen starting point. Under mild additional conditions, we also proved, convergence of the complete sequence of iterates can only occur to a second-order critical point. Preliminary numerical experience on the set of unconstrained test problems from the CUTEr collection indicate that, although there is little to gain in CPU-time when comparing the new algorithm to a pure trust-region method, significant gains in the number of iterations and function/gradient evaluations can be achieved.

References

[1] C. M. Chin and R. Fletcher. Convergence properties of SLP-filter algorithms that takes EQP steps. *Mathematical Programming, Series A*, 96(1):161–177, 2003.

[2] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-Region Methods*. Number 01 in MPS-SIAM Series on Optimization. SIAM, Philadelphia, USA, 2000.

[3] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

[4] R. Fletcher, N. I. M. Gould, S. Leyffer, Ph. L. Toint, and A. Wächter. Global convergence of trust-region SQP-filter algorithms for nonlinear programming. *SIAM Journal on Optimization*, 13(3):635–659, 2002.

[5] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, 91(2):239–269, 2002.

[6] R. Fletcher, S. Leyffer, and Ph. L. Toint. On the global convergence of a filter-SQP algorithm. *SIAM Journal on Optimization*, 13(1):44–59, 2002.

[7] C. C. Gonzaga, E. Karas, and M. Vanti. A globally convergent filter method for nonlinear programming. *SIAM Journal on Optimization*, 13(3):646–669, 2003.

[8] N. I. M. Gould, S. Leyffer, and Ph. L. Toint. A multidimensional filter algorithm for nonlinear equations and nonlinear least-squares. *SIAM Journal on Optimization*, (to appear), 2004.

[9] N. I. M. Gould, S. Lucidi, M. Roma, and Ph. L. Toint. Solving the trust-region subproblem using the Lanczos method. *SIAM Journal on Optimization*, 9(2):504–525, 1999.

[10] N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEr, a contrained and unconstrained testing environment, revisited. *Transactions of the ACM on Mathematical Software*, 29(4):373–394, 2003.

[11] N. I. M. Gould, D. Orban, and Ph. L. Toint. GALAHAD—a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization. *Transactions of the ACM on Mathematical Software*, 29(4):353–372, 2003.

[12] 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. Technical Report 03/15, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2003.

[13] J. J. Moré and D. C. Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4(3):553–572, 1983.

[14] J. Nocedal and Y. Yuan. Combining trust region and line search techniques. In Y. Yuan, editor, *Advances in Nonlinear Programming*, pages 153–176, Dordrecht, The Netherlands, 1998. Kluwer Academic Publishers.

[15] Ph. L. Toint. A non-monotone trust-region algorithm for nonlinear optimization subject to convex constraints. *Mathematical Programming*, 77(1):69–94, 1997.

[16] M. Ulbrich, S. Ulbrich, and L. N. Vicente. A globally convergent primal-dual interior point filter method for nonconvex nonlinear programming. *Mathematical Programming, Series A*, (to appear), 2004.

[17] A. Wächter and L. T. Biegler. Global and local convergence of line search filter methods for nonlinear programming. Technical Report CAPD B-01-09, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, USA, 2001.