RAL-TR-2003-004

A Multidimensional Filter Algorithm for Nonlinear Equations and Nonlinear Least Squares

Nicholas I. M. Gould[1,2,3], Sven Leyffer[4,5] and Philippe L. Toint[6,7]

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.

---

[1] Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, England, EU. Email: n.gould@rl.ac.uk

[2] Current reports available from "http://www.numerical.rl.ac.uk/reports/reports.html".

[3] This work was supported in part by the EPSRC grant GR/R46641

[4] Mathematics and Computer Science Division, Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, IL 60439, USA. Email: leyffer@mcs.anl.gov

[5] Current reports available from "http://www-unix.mcs.anl.gov/~leyffer/".

[6] Department of Mathematics, Facultés Universitaires ND de la Paix, 61, rue de Bruxelles, B-5000 Namur, Belgium, EU. Email : philippe.toint@fundp.ac.be

[7] Current reports available from "http://www.fundp.ac.be/~phtoint/pht/publications.html".

Computational Science and Engineering Department
Atlas Centre
Rutherford Appleton Laboratory
Oxfordshire OX11 0QX

February 12, 2003.

1 Introduction

We analyse 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 vectors 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 j-th 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 if the subsets $\mathcal{I}_j$ 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 (2002). 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 Fletcher and Leyffer, 2002, Fletcher, Leyffer and Toint, 1998, Fletcher, Leyffer and Toint, 2002*b*, Gould and Toint, 2001, Fletcher, Gould, Leyffer, Toint and Wächter, 2002*a*, Gould and Toint, 2003*a*, Wächter and Biegler, 2001, Gonzaga, Karas and Vanti, 2002) 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 fact that (1.1) can itself be handled by a filter algorithm, which, as will be seen below, does not explicitly depend on any 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 (2003), but their approach is very different from the one proposed here. It indeed relies on reducing the violation of a subset of equations while keeping the other violations within bounds, and then adaptively redefining these subsets. By contrast, our approach takes all violations into account together and does not require the revision of any subset during the algorithm.

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 Newton's or the Gauss-Newton method (or one on 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, 2000) 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 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, and this task 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

In order 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 multi-criteria 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 in order to provide an efficient algorithmic tool. In particular, we do not wish to accept a new point x_k^+ if $\theta_k^+ \stackrel{\text{def}}{=} \theta(x_k^+)$ is arbitrarily close to being dominated by another point already in the filter. To avoid this situation, we slighly 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 $\gamma_\theta \in (0, 1)$ is a small positive constant, and where $\delta(\cdot, \cdot)$ is a one of the following:

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

2.1.1 Adding a new point to the filter

In order to avoid cycling, and assuming the trial point is acceptable in the sense of (2.1), we may wish to add it the 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 value θ_ℓ to be *strongly dominated* in the sense that

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

If this happens, we simplify later comparisons by removing θ_ℓ from the filter. (Note that, in the case where $\delta(\|\theta_\ell\|, \|\theta_k^+\|) = \|\theta_k^+\|$, we may weaken the above condition to remove θ_ℓ if simply $\theta_{j,l} > \theta_{j,q}$.)

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*

$$\mathcal{B}_k = \{x_k + s \mid \|s\| \leq \Delta_k\}.$$

where we believe this model to be reasonnably accurate. 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

$$m_k(x_k) - m_k(x_k + s_k) \geq \kappa_{\rm mdc} \|g_k\| \min \left[\frac{\|g_k\|}{\beta_k}, \Delta_k \right], \tag{2.2}$$

where $g_k = \nabla m_k(x_k)$ and β_k is a positive upper bound on the norm of the Hessian of m_k. This is a natural requirement in the context of trust-region algorithms (see Conn et al., 2000, p. 131, for instance), and there are efficient algorithms for achieving (2.2) in both small- and large-scale cases.

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

$$\|s_k\| \leq \Delta_k \tag{2.3}$$

at every iteration to obtain our most important convergence result. In particular, this means that, whenever (2.3) is not enforced and $m = n$, we might use, the step $s_k = -J_k^{-1} c_k$, where J_k is the Jabobian of $c(x)$ at x_k (assumed here to be full rank). This step that leads to the root of the first-order model for $c(x)$, which is also the global minimizer of

$$m_k^{\rm GN}(x_k + s) = \tfrac{1}{2} \sum_{i=1}^{p} \|c_{\mathcal{I}_i}(x_k) + J_{\mathcal{I}_i}(x_k) s\|^2, \tag{2.4}$$

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 (2.3) is not enforced, or one could choose in this case to minimize the second-order Taylor series for $f(x)$

$$m_k^{\rm N}(x_k + s) = m_k^{\rm GN}(x_k + s) + \tfrac{1}{2} \sum_{i=1}^{p} \sum_{j \in \mathcal{I}_i} c_j(x_k) \langle s, \nabla^2 c_j(x_k) s \rangle, \tag{2.5}$$

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

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.

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 $0 < \gamma_0 \leq \gamma_1 < 1 \leq \gamma_2$, $\gamma_\theta \in (0,1)$, $0 < \eta_1 < \eta_2 < 1$. 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 (2.2) and that also satisfies (2.3) 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 (2.3) fails.
- If x_k^+ is not acceptable for the current filter:
  If (2.3) holds and $\rho_k \geq \eta_1$, 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 $\|s_k\| \leq \Delta_k$, update the trust-region radius by choosing

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

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

This algorithm only provides 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 (2.2) and remain in the trust-region when required, but the literature provides several choices of direct or iterative techniques, see Moré and Sorensen (1983), Gould, Lucidi, Roma and Toint (1999) or Chapter 7 of Conn et al. (2000) 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 $\rho_k \geq \eta_1$.

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 $g_k = \nabla m_k(x_k) = \nabla f(x_k)$.

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

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

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

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

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 Chapter 6 and 16 of Conn et al., 2000).

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 AS1-AS2 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

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

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

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

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

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

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 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 (2.3) 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 Conn et al. (2000) 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 $x_k = x_*$ for all sufficiently large k, 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 (2.3) fails. These iterations have no effect, and may be ignored, as we argued above.
- The second are iterations where x_k^+ is not acceptable for the filter, but (2.3) holds. These are perfectly standard trust-region iterations.
- The third kind is when x_k^+ is acceptable for the filter, but θ_k^+ is not added to it because $\rho_k \geq \eta_1$ and (2.3) 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 Conn et al., 2000, Theorem 6.4.6) that the limit inferior in Lemma 3.3 can be replaced by a true limit provided

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

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.

Lemma 3.5 Suppose that A1–A4 and (3.6) hold. Then $\lim_{k\to\infty} \|\nabla f(x_k)\| = 0$. 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

We now look at the practical behaviour of the new method when applied to test problems of the CUTEr collection (see Bongartz, Conn, Gould and Toint, 1995, and Gould, Orban and Toint, 2002*a*). Table 4.1 on page 11 indicates the names and dimensions of the test problems considered. The columns "res. size" and "ign. bnds" in this table indicate whether or not the residual at the solution is zero and whether the original problem statement includes bounds that are ignored here. As is apparent from this table, the test set includes a fair mix of problems sizes and over/under-determined cases. All experiments reported in this section were run with the current version of the Fortran 95 FILTRANE package (see Gould and Toint, 2003*b*) on a Dell Latitude C840 portable computer (1.6 MHz, 1Gbyte of RAM) under the Fujitsu frt Fortran compiler with default optimization.

In what follows, we compare several variants of the algorithm discussed above for reliability and efficiency. All variants discussed use the choice

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

Efficiency comparisons are made using the performance profiles introduced by Dolan and Moré (2001). Suppose that a given 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 that the smaller this statistic the better the algorithm 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

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

where $s_j^* = \min_{i \in \mathcal{A}} s_{ij}$. 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 $\lim_{\sigma \to \infty} p_i(\sigma)$ 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 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.

FTR : This variant is Algorithm 2.1 where, at each iteration, the trial point is computed by approximately minimizing $m^{\rm GN}(x_k+s)$ using the Generalized Lanczos Trust-Region

Problem	n	n free	m	res. size	ign. bnds	Problem	n	n free	m	res. size	ign. bnds
AIRCRFTA	8	5	5	0		HEART6	6	6	6	0	
ARGAUSS	3	3	15	0		HEART8	8	8	8	0	
ARGLALE	200	200	400	> 0		HYDCAR6	29	29	29	0	
ARGLBLE	200	200	400	> 0		HYDCAR20	99	99	99	0	
ARGLCLE	200	200	399	> 0		INTEGREQ	502	500	500	0	
ARGTRIG	200	200	200	0		METHANB8	31	31	31	0	
ARTIF	5002	5000	5000	0		METHANL8	31	31	31	0	
ARWHDNE	500	500	998	> 0		MSQRTA	1024	1024	1024	0	
BDVALUE	102	100	100	0		MSQRTB	1024	1024	1024	0	
BRATU2D	5184	4900	4000	0		NONMSQNE	49	49	49	> 0	
BRATU2DT	5184	4900	4000	0		NYSTROM5	18	15	20	0	
BRATU3D	4913	3375	3375	0		PFIT1	2	2	3	0	yes
BROTDN3D	5000	5000	5000	0		PFIT2	2	2	3	0	yes
BROTDNBD	5000	5000	5000	0		PFIT3	2	2	3	0	yes
BROWNALE	200	200	199	0		PFIT4	2	2	3	0	yes
CBRATU2D	3200	2888	2888	0		POROUS1	4096	3844	3884	0	
CBRATU3D	3456	2000	2000	0		POROUS2	4096	3844	3884	0	
CHANDHEQ	100	100	100	0		POWELLBS	2	2	2	0	
CHANNEL	9600	9598	9598	0		POWELLSQ	2	2	2	0	
CHNRSBNE	50	50	98	0		QR3D	610	610	610	0	yes
CLUSTER	2	2	2	0		RECIPE	3	3	2	0	
CUBENE	2	2	2	0		RES	20	20	14	0	
DECONVNE	61	61	40	0		RSNBRNE	2	2	2	0	
DRCAVTY1	1225	961	961	0		SEMICON2	5002	5000	5000	0	yes
EIGENA	110	110	110	0		SINVALNE	2	2	2	0	
EIGENB	110	110	110	0		VANDERM1	100	100	199	> 0	
EIGENC	462	462	462	0		VANDERM2	100	100	199	> 0	
GROWTH	3	3	12	> 0		WOODSNE	4000	4000	3001	> 0	
HATFLDF	3	3	3	0		YFITNE	3	3	17	0	
HATFLDG	25	25	25	0							

Table 4.1: Test problem characteristics

algorithm of Gould et al. (1999) as implemented in the GALAHAD library (Gould et al., 2002*a*). This procedure is terminated at the first s for which

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

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 addtion,

the condition

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

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

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

or if the number of iterations exceeds 1000, or if the computing time exceeds one hour. Other details of the implementation are discussed in Gould and Toint (2003*b*) and do not matter in our comparison.

TR: This variant is identical to FTR, 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 Conn et al., 2000). Its implementation is comparable in reliability and efficiency to LANCELOT B of GALAHAD.

FTRf : Identical to FTR, except that (4.1) is replaced by the more stringent

$$\|\nabla m^{\rm GN}(x_k + s)\| \leq \sqrt{\epsilon_M}\|\nabla m^{\rm GN}(x_k)\|, \tag{4.3}$$

effectively requiring a full-accuracy subproblem solution.

TRf: Identical to TR, except that (4.1) is replaced by (4.3).

All four variants are reasonably reliable, even if they all failed (while still producing a very good approximation of the solution) on problems `ARBLBLE` and `ARGLCLE`, two very ill-conditioned linear least-squares problems. TRf required more than 1000 iterations for `HYDCAR20`. FTR fails on `PFIT2` and `PFIT3` because one of the bounds on the variables (that are ignored in our test) was violated, causing a negative argument for a logarithm. This error is therefore attributable to our choice of test problems and not to the algorithm itself. FTRf ran out of time on `DRCAVTY1` and `POROUS1` and stalled on `POROUS2`.

The relative efficiency of these methods are illustrated by the performance profile in Figures 4.1 to 4.3. The overall performance of the filter variants is impressive. These profiles show that FTR is the clear winner in both CPU-time and number of conjugate-gradient iterations (these two measures being obviously strongly correlated): the FTR filter variant appears to be within a factor two of the best for approximately 90% of the test problems (it is fastest on 76%). The dominance of the FTR and FTRf variants is even stronger if one considers the number of iterations (which, in our setting, is equivalent to the number of constraints evaluations and globally proportional to the number of Jacobian evaluations). The efficiency gains introduced by the new filter techniques thus appear to be very significant.

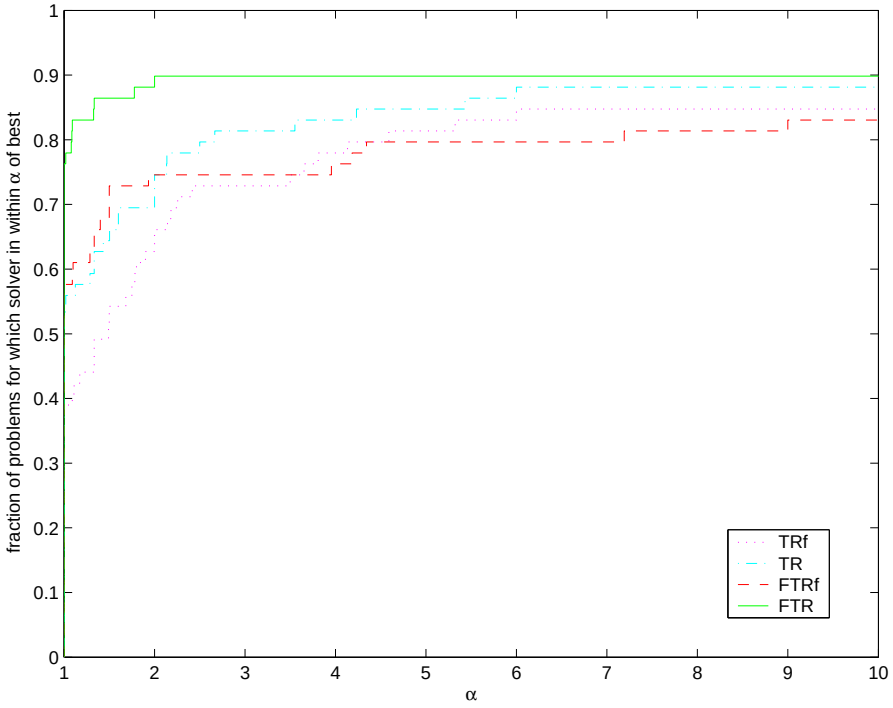

Figure 4.1: CPU time performance profile for TR, TRf, FTR and FTRf

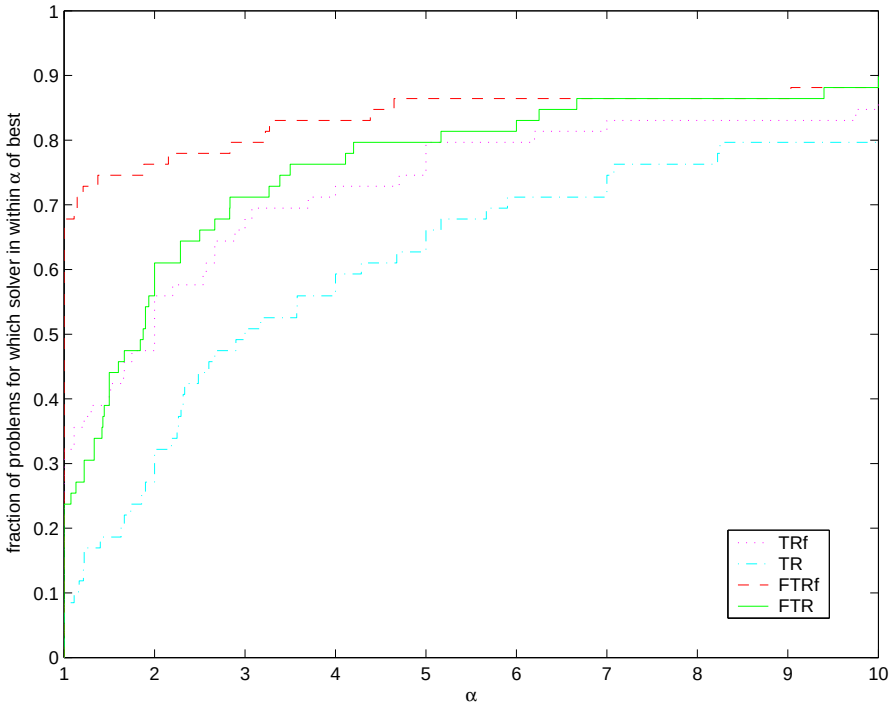

Figure 4.2: Iteration number performance profile for TR, TRf, FTR and FTRf

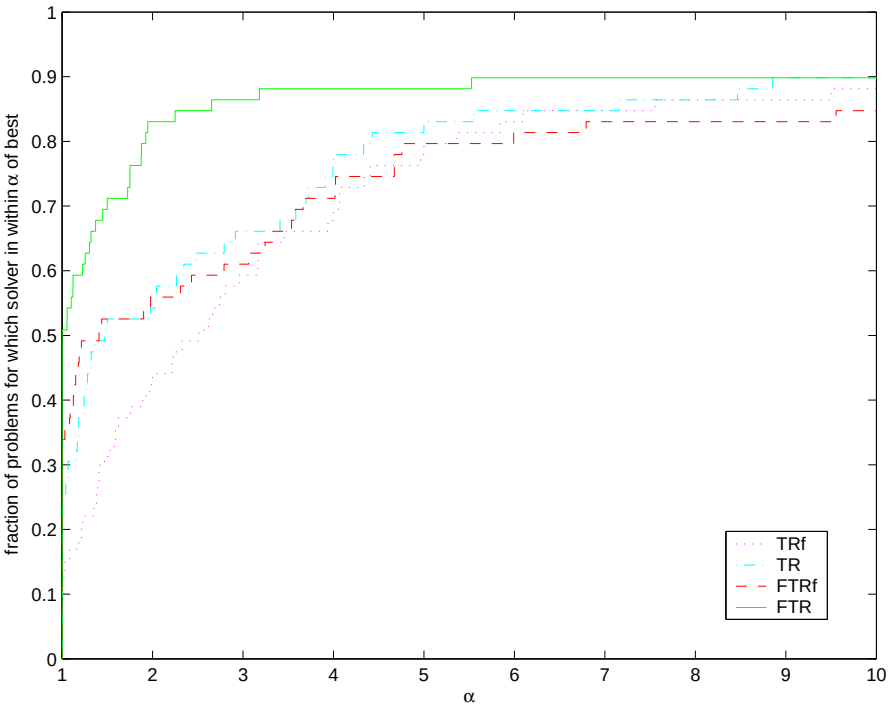

Figure 4.3: CG iterations performance profile for TR, TRf, FTR and FTRf

We now turn our attention the effect of preconditioning on these results. We introduce 8 further variants of our algorithm, that correspond to the four previously studied variants (FTR, FTRf, TR and TRf) where we now use a diagonal preconditioner (suffix -d), or a preconditioner obtained by extracting a band matrix of semi-bandwidth 5 from the Hessian $J_k^T J_k$ of the Gauss-Newton model (2.4) (suffix -b5). Thus FTR-b5 is the FTR variant with a banded preconditioner of semi-bandwidth 5 and TRf-d is the TRf variant with a diagonal preconditioner. Moreover, the tests (4.1), (4.3) and (4.2) are no longer expressed in the Euclidean norm, but in the preconditioned norm $\|\cdot\| = \sqrt{\langle \cdot, M^{-1} \cdot \rangle}$, where M is the preconditioner. Direct efficiency comparisons between the preconditioned and unpreconditioned variants are therefore inappropriate.

Of course, both reliability and efficiency of preconditioned variants strongly depend on how suitable the preconditioner is for the test problems considered. In particular, one expects the diagonal preconditioner to improve performance on diagonally dominant problems, and to be much less advantageous for problems whose Hessian has a significant off-diagonal part.

We first comment on the reliabity of the diagonally preconditioned variants. The four variants now stall on problems `ARWHDNE` and `POWELLSQ`, both of which feature strong off-diagonal Hessian entries. In addition, TRf-d ran out of time on `MSQRTA`, FTRf required more than 1000 iterations on `HEART6`, the same being true for FTR on problem `ARTIF`,

while FTR cause an exponential to overflow in evaluating the residuals.

Relative efficiencies are illustrated in Figures 4.4 to 4.6. Again the filter variants FTR-d and FTRf-d dominate clearly the pure trust-region variants TR-d and TRf-d. This dominance, although slighly less marqued than in the unpreconditioned case, remains substantial.

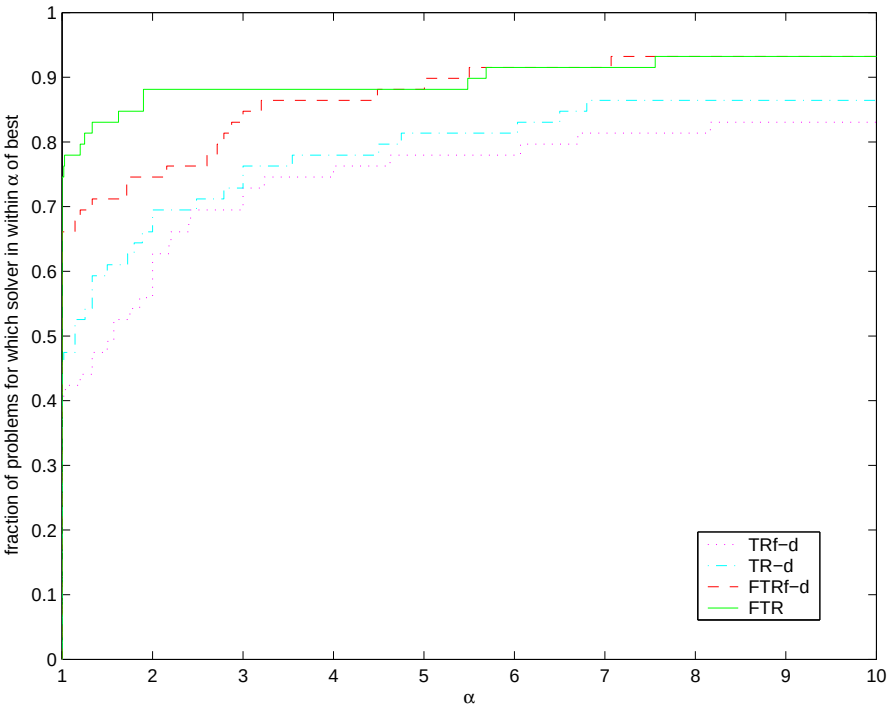

Figure 4.4: CPU time performance profile for TR-d, TRf-d, FTR-d and FTRf-d

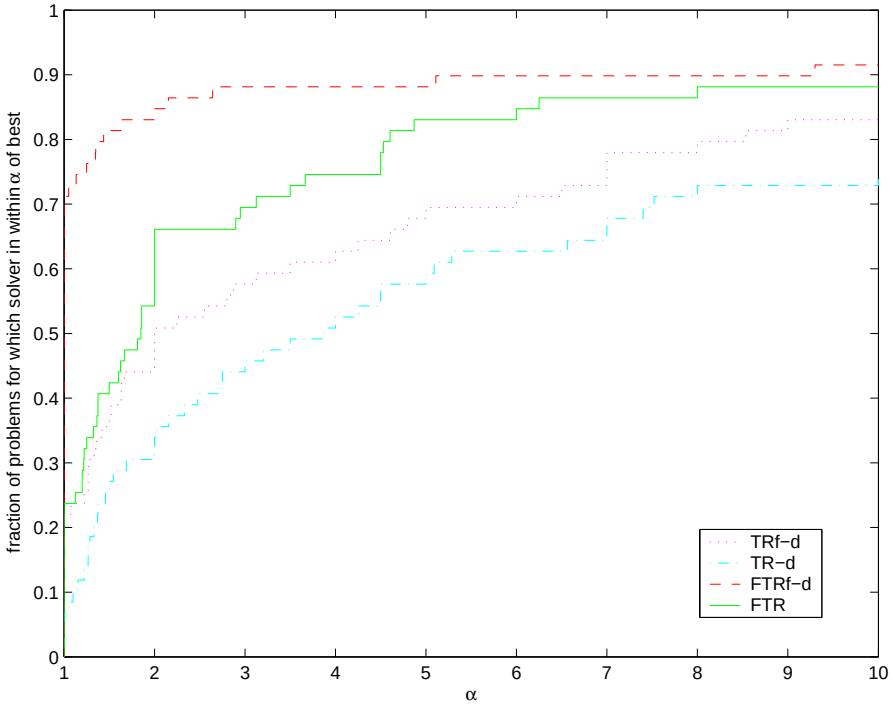

Figure 4.5: Iterations performance profile for TR-d, TRf-d, FTR-d and FTRf-d

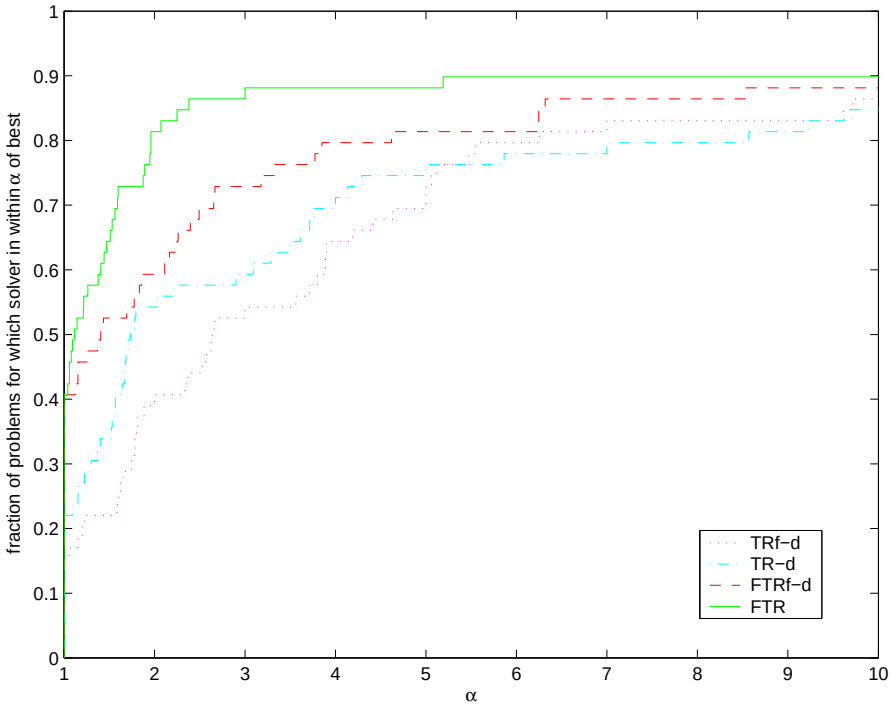

Figure 4.6: CG iterations performance profile for TR-d, TRf-d, FTR-d and FTRf-d

The choice of a banded submatrix of the model's Hessian as preconditioner does not modify the conclusions reached so far. The four new variants still stall on `ARWHDNE` but also on `NONMSQNE`. Only the pure trust-region variants TRF-b5 and TR-b5 now stall on `POWELLSQ`, while FTRf-b5 stalls on `HATFLDF` and FTR-b5 on `BROYDNBD`. All four b5 variants also fail on `SEMICON2` because the violation of the problem bounds causes an arithmetic error. As noted above, this is not a defect that can be attributed to the algorithm, but rather to the use of this test problem without its associated bound constraints. The performance profiles of Figures 4.7 to 4.9 confim the observed dominance of the filter variants over the pure trust-region ones.

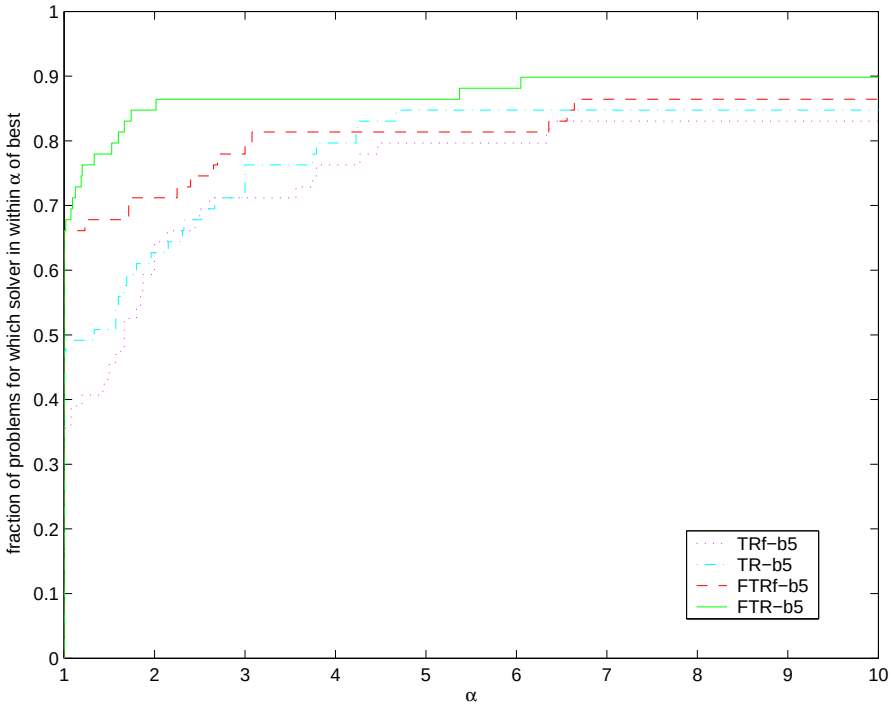

Figure 4.7: CPU time performance profile for TR-b5, TRf-b5, FTR-b5 and FTRf-b5

We now consider the choice of the Gauss-Newton model (2.4) and compare it to the full Newton model given by (2.5). As is well-known, this last model is typically more efficient when the solution of the problem occurs at a point x_* for which $\|c(x_*)\|$ is strictly positive. Another important feature of (2.5) is that, in contrast with (2.4), 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 towards unbounded minima of (2.5), 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 unecessarily long steps, which have then to be cut back, generating inefficient oscillations in the step length.

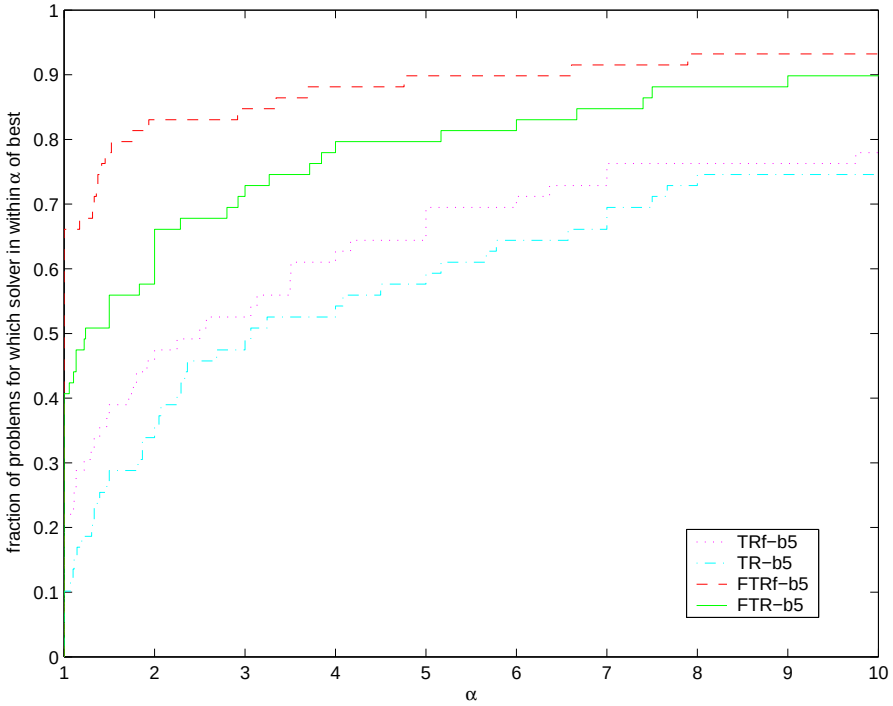

Figure 4.8: Iterations performance profile for TR-b5, TRf-b5, FTR-b5 and FTRf-b5

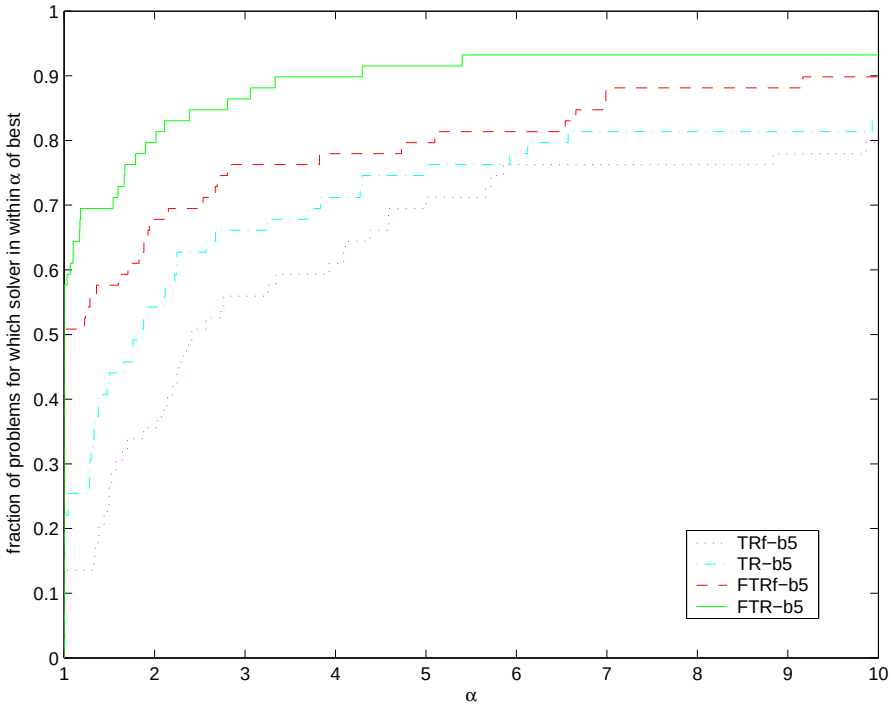

Figure 4.9: CG iterations performance profile for TR-b5, TRf-b5, FTR-b5 and FTRf-b5

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 re-entry facility in the GLTR package, that exploits previously computed Krylov spaces (see Gould, Orban and Toint, 2002*b*, for details). We thus introduce two new variants of Algorithm 2.1:

FTRN: is identical to FTR, except that the model (2.5) is used instead of (2.4) and that unrestricted steps containing negative curvature are restricted before attempting to compute the constraint values at the trial point;

TRN: is derived from TR in the same manner.

Comparing the reliablity of TRN, TR, FTRN and FTR, we note that, as above, all four variants stall on `ARGLBLE` and `ARGLCLE`. TRN requires more than 1000 iterations to solve `HEART6`, the same being true for FTRN on `DRCAVTY1` and `NYSTROM5`. This last variant also stalls on `POWELLSQ`.

Full Newton variants are also less efficient than the Gauss-Newton ones on our test set, as is apparent in the performance profiles of Figures 4.10 to 4.12. But these conclusions must be tempered by the observation that many of our test problems are actually such that

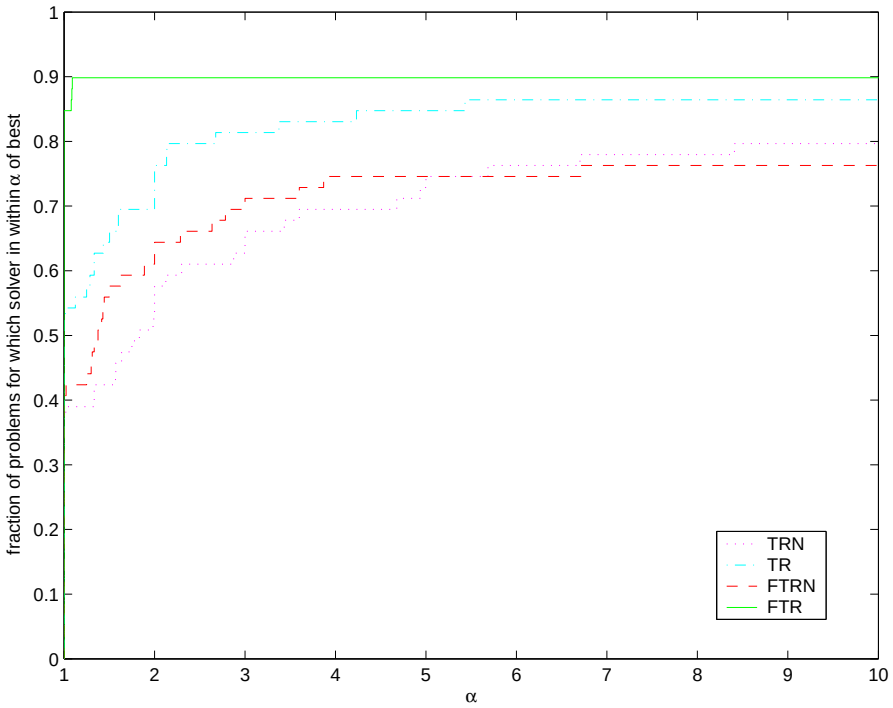

Figure 4.10: CPU time performance profile for TRN, TR, FTRN and FTR

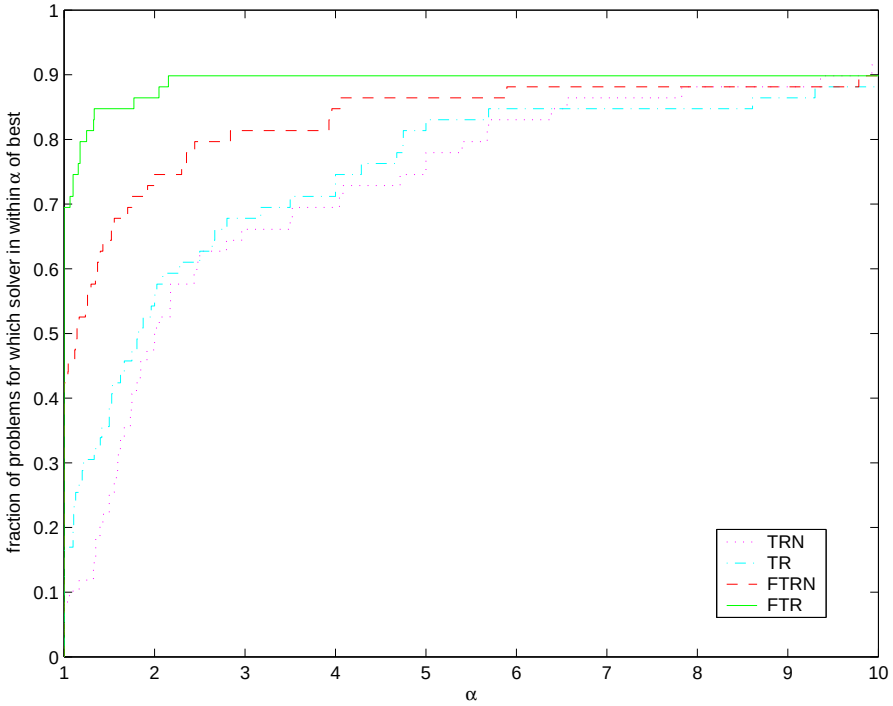

Figure 4.11: Iterations performance profile for TRN, TR, FTRN and FTR

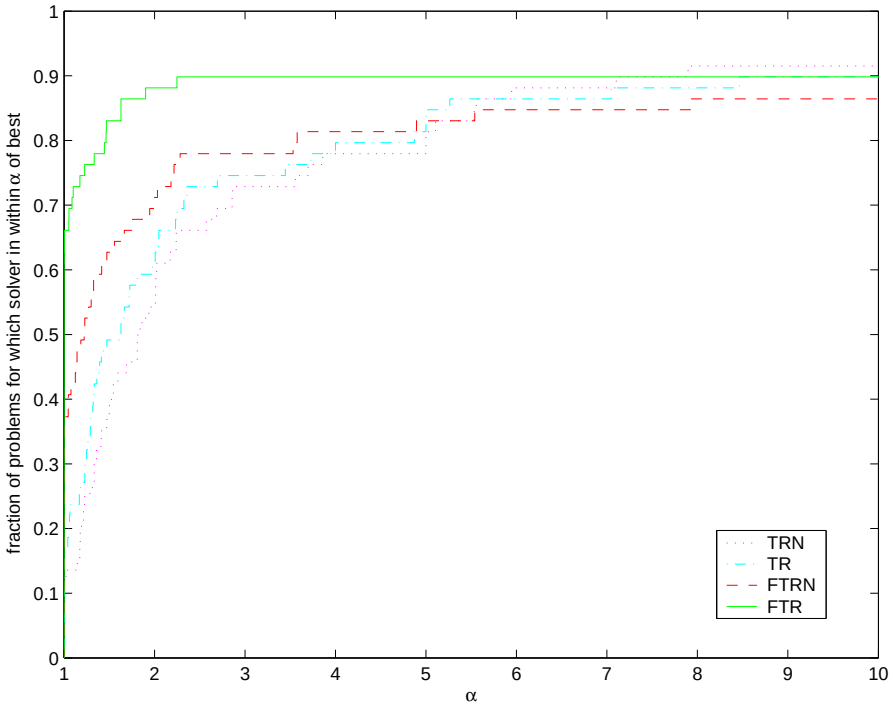

Figure 4.12: CG iterations performance profile for TRN, TR, FTRN and FTR

$\|c(x_*)\| = 0$, therefore implicitly favouring the Gauss-Newton model.

Occasionaly, though, 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.

	CPU(s)	iter.	CG iter.
TRN	0.1	7	12
TR	0.2	134	136
FTRN	0.1	6	9
FTR	0.3	148	252

Table 4.2: Detailed performance of the TRN, TR, FTRN and FTR variants on problem `ARWHDNE`

Adaptive procedures to choose between the Gauss-Newton and the Newton model have been studied (see Dennis, Gay and Welsh, 1981, Toint, 1987, or Lukšan, 1996, for instance), but their application to our framework is beyond the scope of this paper and is postponed to Gould and Toint (2003*b*).

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 FTR variant (in which every equation has its own subset) with 5 different alternative stategies for grouping equations in k disjoint subsets: they are refered to as FTRG(k), with $k = n/2$, $n/10$, 10, 2 and 1. (Note that, in this notation, FTR = FTRG(n).) The variant FTRG(1) may be seen as a particular non-monotone filter-trust-region strategy, in the spirit of the proposals by Ke and Han (1995), Xiao and Chu (1995) or Toint (1997). In our experiments, we simply assigned the i-th equation to the subset

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

Not surprisingly, all FTRG variants and TR fail on the ill-conditioned problems `ARGLBLE`, `ARGLCLE`. Failure on the `PFIT*` problems is more random, since it depends on violating the problem neglected bounds, TR being the only variant without such failure due its insistance on restricted steps. FTRG($n/10$) stalls on `POROUS2`.

The performance profiles in Figures 4.13 to 4.15 show that the variants differ, but to a much 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. The best variants appear to be FTR = FTRG(n)

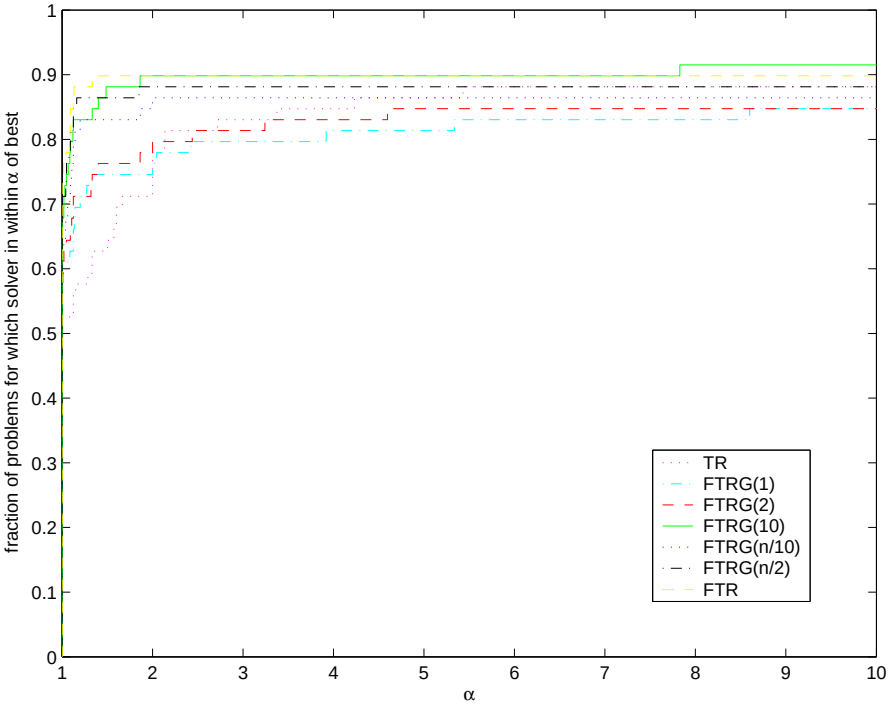

Figure 4.13: CPU time performance profile for FTR, FTRG(1), FTRG(2), FTRG(10), FTRG($n/10$) and FTRG($n/2$)

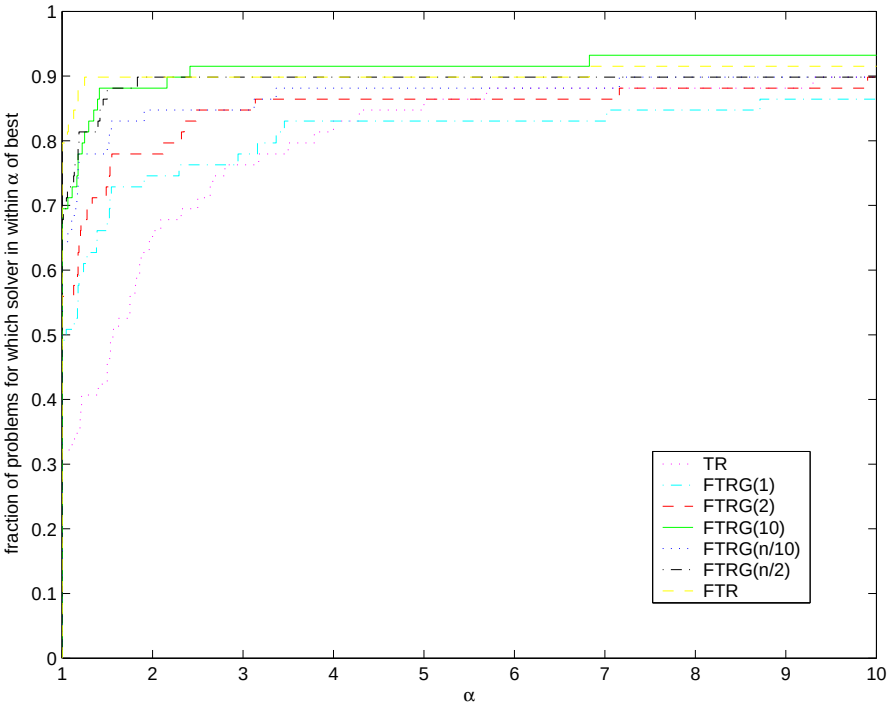

Figure 4.14: Iterations performance profile for FTR, FTRG(1), FTRG(2), FTRG(10), FTRG($n/10$) and FTRG($n/2$)

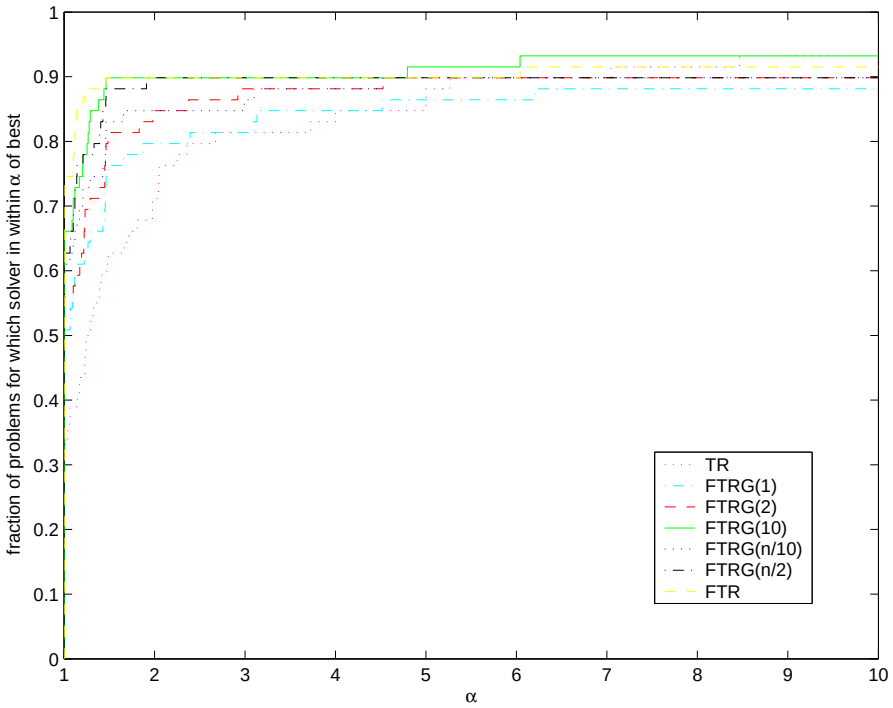

Figure 4.15: CG iterations performance profile for FTR, FTRG(1), FTRG(2), FTRG(10), FTRG($n/10$) and FTRG($n/2$)

and FTRG(10), the latter one being especially interesting if one measures CPU time. This not surprising since p directly conditions the amount of necessary storage for each value in the filter and the amount of work for verifying that a new θ_k^+ value is acceptable for the filter or not. We also observe the good performance of FTRG(1) compared to TR, which indicates that a unidimensional filter may provide an attractive alternative to other non-monotone 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 only resulted in marginal differences that will not be discussed here.

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 it must converge to 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 (Gould and Toint, 2003*b*), is currently under developement for the GALAHAD library of optimization algorithms (see Gould et al., 2002*b*) and should facilitate a wider use of the filter-trust-region algorithm.

While the analysis discussed above is highly encouraging, it only represents the first steps in the area of research that is opened by the combination of classical trust-region method and multidimensional filters. Numerous extensions and further developments are possible, both in 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 $\mathcal{I}_j$ only contains 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 straighforward. Its efficiency is currently under investigation. Another possibility for allowing even more iterates to be potentially acceptable is to introduce non-monotone filter techniques similar to those discussed in Gould and Toint (2003*a*), 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 other norms than the Euclidean one to construct the merit function $f(x)$, the ℓ_1 and ℓ_∞ norms being obvious candidates. In this case, the subproblems consists in (approximately) solving linear programs, which, although feasible in principle, requires further research to define suitable subproblem truncation procedures.

The possibility to handle linear or nonlinear inequalities, in conjunction with equalities or alone, also consists a natural development. Its theoretical aspects directly results 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 k-th constraint at x. The analysis of Section 3 applies without any modification to this more general case. Practical aspects of this development will be described in Gould and Toint (2003*b*).

We conclude by noting that the ideas presented in this paper immediately suggest other uses of the same techniques, like the introduction of multidimensional filters in nonlinear programming (where current state-of-the-art methods only use a two-dimensional filter

space), or filter techniques for the solution nonlinear equilibrium problems (where the complementarity residual is a separate filter entry).

References

I. Bongartz, A. R. Conn, N. I. M. Gould, and Ph. L. Toint. CUTE: Constrained and Unconstrained Testing Environment. *ACM Transactions on Mathematical Software*, **21**(1), 123–160, 1995.

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.

J. E. Dennis, D. M. Gay, and R. E. Welsh. An adaptive nonlinear least-squares algorithm. *ACM Transactions on Mathematical Software*, **7**(3), 348–368, 1981.

E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. Technical Report ANL/MCS-P861-1200, Mathematics and Computer Science, Argonne National Laboratory, Argonne, Illinois, USA, 2001.

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

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

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*a*.

R. Fletcher, S. Leyffer, and Ph. L. Toint. On the global convergence of a SLP-filter algorithm. Technical Report 98/13, Department of Mathematics, University of Namur, Namur, Belgium, 1998.

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*b*.

C. C. Gonzaga, E. Karas, and M. Vanti. A globally convergent filter method for nonlinear programming. Technical report, Department of Mathematics, Federal University of Santa Catarina, Florianopolis, Brasil, 2002.

N. I. M. Gould and Ph. L. Toint. Global convergence of a hybrid trust-region SQP-filter algorithm for general nonlinear programming. Technical Report 01/07, Department of Mathematics, University of Namur, Namur, Belgium, 2001.

N. I. M. Gould and Ph. L. Toint. Global convergence of a non-monotone trust-region filter algorithm for nonlinear programming. Technical Report TR-2003-003, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2003*a*.

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 (in preparation), Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2003*b*.

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.

N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEr, a contrained and unconstrained testing environment, revisited. Technical Report 02/07, Department of Mathematics, University of Namur, Namur, Belgium, 2002*a*.

N. I. M. Gould, D. Orban, and Ph. L. Toint. GALAHAD—a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization. Technical Report TR-2002-014, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2002*b*.

X. Ke and J. Han. A class of nonmonotone trust region algorithms for constrained optimizations. *Chinese Science Bulletin*, **40**(16), 1321–1324, 1995.

L. Lukšan. Hybrid methods for large sparse nonlinear least-squares. *Journal of Optimization Theory and Applications*, **89**(3), 575–595, 1996.

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

Ph. L. Toint. On large scale nonlinear least-squares calculations. *SIAM Journal on Scientific and Statistical Computing*, **8**(3), 416–435, 1987.

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

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.

Y. Xiao and E. K. W. Chu. Nonmonotone trust region methods. Technical Report 95/17, Monash University, Clayton, Australia, 1995.