FILTRANE, A FORTRAN 95 FILTER-TRUST-REGION PACKAGE FOR SOLVING NONLINEAR FEASIBILITY PROBLEMS

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

Report 03/17 September 2, 2003

[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: philippe.toint@fundp.ac.be

FILTRANE, a Fortran 95 filter-trust-region package for solving nonlinear feasibility problems

Nick Gould and Philippe L. Toint

2 September 2003

Abstract

This paper describes the algorithm and performance of FILTRANE, a new Fortran 95 packages for finding vectors satisfying general sets of nonlinear equations and/or inequalities. A number of algorithmic variants are discussed and extensively compared on a set of CUTEr test problems, indicating that the default variant is both reliable and efficient. This discussion provides a first experimental study of the parameters inherent to filter algorithms.

1 Introduction

The purpose of this paper is to present FILTRANE, a new Fortran 95 package in the GALAHAD library (see Gould, Orban and Toint, 2003*c*) for solving the general smooth feasibility problem, that is the problem to find a vector $x \in \mathbb{R}^n$ such that

$$c_{\mathcal{E}}(x) = 0, \tag{1.1}$$

and

$$c_{\mathcal{I}}(x) \geq 0, \tag{1.2}$$

where $c_{\mathcal{E}}(x)$ and $c_{\mathcal{I}}(x)$ are smooth functions from $\mathbb{R}^n$ into $\mathbb{R}^m$ and $\mathbb{R}^q$, respectively. If such a point cannot be found, it is desired to find a local minimizer of the constraint violations. We choose here to consider the Eucliden norm of these violations, that is to find a local minimizer of the function

$$\min_x \tfrac{1}{2}\|\theta(x)\|^2, \tag{1.3}$$

where we define

$$\theta(x) \stackrel{\text{def}}{=} \begin{pmatrix} c_{\mathcal{E}}(x) \\ [c_{\mathcal{I}}(x)]_- \end{pmatrix} \in \mathbb{R}^p, \tag{1.4}$$

with $\|\cdot\|$ denoting the Euclidean norm, with $p = m + q$ and $[c_{\mathcal{I}}(x)]_- = \min[0, c_{\mathcal{I}}(x)]$, the minimum being taken componentwise. An important special case of this problem is when $q = 0$, which gives systems of smooth nonlinear equations. The problem under consideration is therefore not only fairly general, but also practically important because a large number of applications can be cast in this form. Moreover, solving the feasibility problem may also occur as a subproblem in practically more complicated contexts, such as the the “restoration” phase in the solution of the nonlinear programming problem using filter methods (see Fletcher and Leyffer, 2002, Fletcher and Leyffer, 1998, Fletcher, Leyffer and Toint, 2002*b*, Gonzaga, Karas and Vanti, 2002 or Fletcher, Gould, Leyffer, Toint and Wächter, 2002*a*, amongst others).

The method of choice for solving (1.1)–(1.2) or (1.3) is Newton's method, because of its fast convergence properties. However, as is well-known, Newton's method must be safeguarded to ensure that it converges to a solution even from starting points that are far from the solution, a feature that is not automatic otherwise. Various safeguarding techniques are known, including the use of linesearches (see Ortega and Rheinboldt, 1970, Dennis and Schnabel, 1983, Toint, 1986, Toint, 1987, . . .) or trust regions (see Moré and Sorensen, 1984, Nocedal, 1984, or Chapter 16 of Conn, Gould and Toint, 2000). More recently, Gould, Leyffer and Toint (2003*a*) have proposed a method that combines the basic trust-region mechanism with filter techniques: not only did they prove global convergence for the algorithm, but they also reported very encouraging initial numerical experience. Our current objective is to describe the FILTRANE package that results from this research, and to investigate some of its properties and features in substantially more detail.

The paper is organized as follows. Section 2 presents the filter algorithm and some of its variants whose performance we wish to study. Section 3 discusses the numerical results obtained with the package and its variants, compares them whenever possible, and analyzes the sensitivity of their performance as one varies some of its important algorithmic parameters. Some conclusions are finally drawn in section 4.

2 The filter algorithm and its algorithmic options

2.1 The objective function, its models and the step

In order to subsume both (1.1)–(1.2) and (1.3) in a single description, we consider an algorithm which aims at minimizing

$$f(x) = \tfrac{1}{2}\|\theta(x)\|^2.$$

For simplicity of exposition, we start by assuming that the problem only contains nonlinear equations ($q = 0$). In this case, we may build two distinct local quadratic models of $f(x)$ in the neighbourhood of a given iterate x_k. The first is the Gauss-Newton model, and is given by

$$m_k^{\rm GN}(x_k + s) = \tfrac{1}{2}\sum_{i=1}^{m} \|c_{\mathcal{E}_i}(x_k) + J_{\mathcal{E}_i}(x_k)s\|^2, \tag{2.1}$$

where $J_{\mathcal{E}_i}(x_k)$ is the Jacobian of $c_{\mathcal{E}}(x)$ at x_k. The second is the full second-order Newton model

$$m_k^{\mathrm{N}}(x_k+s) = m_k^{\mathrm{GN}}(x_k+s) + \tfrac{1}{2}\sum_{i=1}^{p}\sum_{j\in\mathcal{E}_i} c_j(x_k)\langle s, \nabla^2 c_j(x_k)s\rangle, \tag{2.2}$$

(with $\mathcal{E}_i = \{1,\ldots,m\}$) which includes an additional term involving the curvature of the equality constraints.

In FILTRANE, we have chosen to compute the step s_k by minimizing one of these models in some region surrounding the current iterate x_k, defined by the constraint

$$\|s_k\|_k \le \tau_k \Delta_k, \tag{2.3}$$

where Δ_k is a trust-region radius which is updated in the usual trust-region manner (see Chapters 6 and 17 of Conn et al., 2000, for instance), and where $\tau_k \ge 1$ is a real parameter which is adjusted from iteration to iteration. The effect of this parameter is allow for steps that potentially extend much beyond the limit of the trust region itself, in the case where convergence seems satisfactory. The precise mechanism for determining τ_k will be discussed in more detail below. The $\|\cdot\|_k$ norm appearing in (2.3) is the preconditioned Euclidean norm, that is

$$\|s\|_k^2 = \langle s, P_k^{-1}s\rangle,$$

where P_k is a symmetric positive-definite preconditioning matrix that is used at the k-th iteration. The solution of the subproblem of minimizing $m_k^{\mathrm{GN}}(x_k+s)$ or $m_k^{\mathrm{N}}(x_k+s)$ subject to (2.3) is computed approximately using the Generalized Lanczos Trust-Region (GLTR) method of Gould, Lucidi, Roma and Toint (1999) as implemented in the GLTR module of GALAHAD (see Gould et al., 2003*c*). This procedure guarantees the familiar Cauchy point condition

$$m_k(x_k) - m_k(x_k+s_k) \ge \kappa_{\mathrm{mdc}}\|g_k\| \min\left[\frac{\|g_k\|}{\beta_k}, \Delta_k\right], \tag{2.4}$$

where m_k is either m_k^{GN} or m_k^{N}, $g_k = \nabla m_k(x_k)$, κ_{mdc} is a constant in $(0,1)$, and β_k is a positive upper bound on the norm of the Hessian of m_k.

Besides using $P_k = I$ (i.e. no preconditioning at all), FILTRANE can also be instructed to use a diagonal preconditioning that is obtained by extracting the diagonal of the matrix $H_k \stackrel{\mathrm{def}}{=} J_{\mathcal{E}_i}(x_k)J_{\mathcal{E}_i}(x_k)^T$, or a banded preconditioning matrix of semi-bandwidth 5 obtained by extracting the corresponding part of H_k and modifying it if necessary to ensure its positive definiteness (see Conn, Gould and Toint, 1992 for details of that procedure). The package also allows the use of a user-defined preconditioning via its reverse communication interface.

Note that the subproblem is convex whenever the Gauss-Newton model (2.1) is used, but that this is not necessarily the case if Newton's model (2.2) is used, since the matrices $\nabla^2 c_j(x_k)$ may be indefinite. In this last case, using $\tau_k > 1$ in (2.3) is inappropriate, and we impose that $\tau_k = 1$. Unfortunately, non-convexity of the model is only discovered in the course of its mimimization: when this happens for $\tau_k > 1$, we then simply use the re-entry feature of the GLTR module, which computes, at modest cost, the minimum of the model on the Krylov space explored so far for a shorther value of the radius ($\tau_k = 1$).

2.2 The filter-trust-region mechanism

Once the step s_k is computed, we may define the *trial point* to be

$$x_k^+ = x_k + s_k \tag{2.5}$$

and consider the question of deciding whether or not it is acceptable as our next iterate x_{k+1}. In order to define our filter, we first say that a point x_1 *dominates* a point x_2 whenever

$$|\theta_i(x_1)| \leq |\theta_i(x_2)| \quad \text{for all} \quad i \in \mathcal{E}_i.$$

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 i. 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 m-tuples of the form $(\theta_{1,k}|, \ldots, |\theta_{m,k}|)$ such that, for $k \neq \ell$,

$$|\theta_{i,k}| < |\theta_{i,\ell}| \quad \text{for at least one} \quad i \in \mathcal{E}_i.$$

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

$$\forall \theta_\ell \in \mathcal{F} \quad \exists i \in \mathcal{E}_i \quad |\theta_i(x_k^+)| < \Big[|\theta_{i,\ell}| - \gamma_\theta \delta(\|\theta_\ell\|, \|\theta_k^+\|) \Big]_+ \tag{2.6}$$

where $\gamma_\theta \in (0, 1/\sqrt{m})$ is a small positive constant, $[w]_+ = \max[0, w]$, and where $\delta(\cdot,\cdot)$ is one of the following:

$$\delta(\|\theta_\ell\|, \|\theta_k^+\|) = \|\theta_\ell\|, \tag{2.7}$$

$$\delta(\|\theta_\ell\|, \|\theta_k^+\|) = \|\theta_k^+\|, \tag{2.8}$$

or

$$\delta(\|\theta_\ell\|, \|\theta_k^+\|) = \min(\|\theta_\ell\|, \|\theta_k^+\|). \tag{2.9}$$

The upper bound of $1/\sqrt{m}$ on γ_θ ensures that the right-hand side of (2.6) is always positive for some j for the choices (2.7) and (2.9), and thus that points acceptable for the filter always exist in these cases. Note that such points must exist if (2.8) is considered provided $\|\theta_\ell\| > 0$, but a small value for γ_θ clearly makes it more likely that (2.6) holds for a given θ_k^+.

In order to avoid cycling, and assuming the trial point is acceptable in the sense of (2.6), we may wish to add it the to the filter, so as to avoid other iterates that are worse, that is we perform the simple operation

$$\mathcal{F} \leftarrow \mathcal{F} \cup \{\theta_k\}.$$

This may however cause an existing filter value θ_ℓ to be *strongly dominated* in the sense that

$$\exists\, \theta_q \in \mathcal{F} \quad \forall j \in \{1, \ldots, p\} \quad |\theta_{j,\ell}| \geq |\theta_{j,q}| - \gamma_\theta \|\theta_\ell\|. \tag{2.10}$$

If this happens, we simplify later comparisons by removing θ_ℓ from the filter. (Note that $\theta_{j,\ell} > \theta_{j,q}$ is sufficient in this last condition if we restrict our choice to $\delta(\|\theta_\ell\|, \|\theta_k^+\|) = \|\theta_k^+\|$.)

If the trial point is not acceptable for the filter, it may nevertheless be acceptable for the usual trust-region mechanism. This requires that $\|s_k\| \leq \Delta_k$ and that ρ_k is sufficiently positive, where ρ_k is the familiar ratio of achieved to predicted reduction defined by

$$\rho_k = \frac{f(x_k) - f(x_k^+)}{m_k(x_k) - m_k(x_k^+)}. \tag{2.11}$$

Our algorithm therefore combines the filter and trust-region acceptability criteria to allow a potentially larger set of trial points to be accepted.

2.3 Penalty for inequalities

Inequality constraints are treated in way entirely similar to that used for equalities: as already mentioned in (1.4) we define θ to measure the violation of the inequality constraints. This causes the ℓ_2-penalty function (1.3) to have discontinuous second derivatives on the boundary of the set of vectors satisfying the inequality constraints, which creates some problems that we discuss below.

2.4 An outline of the algorithm and some further details

2.4.1 The algorithmic framework

We are now ready to outline the FILTRANE algorithm using the ideas developed above. This outline is presented as Algorithm 2.1, page 6. This outline leaves a number of points to be clarified, which is the object of the remainder of this section.

2.4.2 An adaptive model strategy

The first issue that we examine is how the model m_k is chosen. As we discussed above, two natural choices are the Gauss-Newton and full Newton models given by (2.1) and (2.2), respectively. The initial experiments reported in Gould et al. (2003*a*) indicate that the first is very often preferable, but that the latter sometimes brings significant efficiency gains, in particular in the case where $\|\theta\|$ is significantly different from zero

Algorithm 2.1: Outline of the 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 $0 < \gamma_0 \leq \gamma_1 < 1 \leq \gamma_2$, $\gamma_\theta \in (0, 1/\sqrt{p})$, $0 < \eta_1 < \eta_2 < 1$. Compute $c_0 = c(x_0)$ and θ_0. Set $k = 0$, $\mathcal{F} = \emptyset$, and select $\tau_0 \geq 1$.

Step 1: Test for termination.

If either θ_k or $\|\nabla f(x_k)\|$ is sufficiently small, stop.

Step 2: Choose a model and a norm.

Choose a norm $\|\cdot\|_k$ for (2.3). Set m_k to be either m_k^{GN} or m_k^{N}.

Step 3: Determine a trial step.

Compute a step s_k that satisfies (2.3) and (2.4), using the GLTR algorithm. If the model is found to be nonconvex and $\tau_k > 1$, reenter the GLTR algorithm with $\tau_k = 1$. Compute the trial point $x_k^+ = x_k + s_k$.

Step 4: Evaluate the residual at the trial step.

Compute $c(x_k^+)$ and $\theta_k^+ = \theta(x_k^+)$. Define ρ_k according to (2.11).

Step 5: Test to accept the trial step.

- If x_k^+ is acceptable for the current filter:
  Set $x_{k+1} = x_k^+$, select $\tau_{k+1} \geq 1$ and add θ_k^+ to $\mathcal{F}$ if either $\rho_k < \eta_1$ or $\|s_k\| > \Delta_k$.
- If x_k^+ is not acceptable for the current filter:
  If $\|s_k\| \leq \Delta_k$ and $\rho_k \geq \eta_1$, set $x_{k+1} = x_k^+$ and select $\tau_{k+1} \geq 1$. Else, set $x_{k+1} = x_k$ and $\tau_{k+1} = 1$.

Step 6: 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.

at the solution. The default version of FILTRANE therefore includes an adaptive model choice that attempt to exploit the best of these two models.

A first choice strategy is to start with the Gauss-Newton model, but to evaluate ρ_k at each iteration, not only for the model currently in use (Gauss-Newton, initially), but also for the model not being used. Thus we obtain ρ_k^{GN} and ρ_k^{N}. Each iteration for which

$$|\rho_k^{\text{GN}} - 1| \leq |\rho_k^{\text{N}} - 1| \tag{2.12}$$

casts a vote in favor of the Gauss-Newton model, while a vote in favor of the Newton model is recorded otherwise. After n_v iterations, the model credited with a majority of the corresponding n_v votes is used for the next n_v iterations. The parameter n_v represents the "inertia" of the model choice mechanism and prevents a rapid model change. The choice $n_v = 5$ is made by default.

FILTRANE also provides an optional alternative strategy, which differs from the default only in that the condition (2.12) is replaced by

$$\rho_k^{\mathrm{GN}} \geq \rho_k^{\mathrm{N}}. \tag{2.13}$$

2.4.3 Filter management

We now turn to issues related to the way in which the filter technique is implemented.

2.4.3.1 Pre-filtering Since condition (2.6) has to be tested, at each iteration, for the trial point and each filter entry, we may wish to make this test reasonnably efficient. We therefore maintain a list of entries currently in the filter, arranged by order of increasing Euclidean norm. When a new $\theta(x_k^+)$ is tested for filter acceptability, the tests are performed by comparing it to the successive filter entries in that list. If, for some ℓ,

$$\|\theta(x_k^+)\| < \|\theta_\ell\| - \gamma_\theta \sqrt{p}\, \delta(\theta_\ell, \|\theta_k^+\|).$$

then the current filter point lies inside the largest sphere that is tangent (up to the margin) to the ℓ-th filter entry in the list, and (2.6) must hold. Moreover, since the list is organized by increasing values of $\|\theta_\ell\|$, the same must be true of all remaining filter entries in the list, and x_k^+ may be declared acceptable for the filter without further testing.

2.4.3.2 The value of τ_k One of the advantages of the filter algorithm presented above is the possibility of taking steps whose norm exceeds the trust-region radius, without affecting the convergence properties of the method. In practice, this is most useful in the first few iterations, while imposing some limitation on $\|s_k\|$ turns out to be a reasonable stabilization scheme later. We thus have chosen to set $\tau_0 = 10^{20}$ and impose $\tau_k \in [1, \tau_0]$ initially, while we strengthen this condition to $\tau_k \in [1, \tau_{\max}]$, with $\tau_{\max} = 1000$, as soon as it has been reset at least once. The actual value of τ_k varies gradually in this interval: it is doubled at each iteration where $\rho_k \geq \eta_2$ until it reaches its upper bound, but is halved (with a lower bound of 1) if the new point is acceptable for the filter, but $\rho_k < \eta_1$. This somewhat involved compromise appears to balance performance and reliability reasonably.

2.4.3.3 Unsigned filter entries As was suggested in Gould et al. (2003*a*), we may also extend our filter definition by considering $\theta(x_k)$ instead of $|\theta(x_k)|$. In this case, the acceptability condition (2.6) becomes

$$\forall \theta \in \mathcal{F} \quad \exists j \in \{1, \ldots, m\} \text{ such that}$$

$$\begin{aligned} \text{either} \quad & 0 \leq \theta_j(x_k^+) < \left[\theta_{j,\ell} - \gamma_\theta \delta(\|\theta_\ell\|, \|\theta_k^+\|)\right]_+ \\ \text{or} \quad & 0 > \theta_j(x_k^+) > \left[\theta_{j,\ell} + \gamma_\theta \delta(\|\theta_\ell\|, \|\theta_k^+\|)\right]_- \end{aligned} \tag{2.14}$$

(Condition (2.10) can be adapted in the same manner.) This makes the trial point potentially more often acceptable, as this condition is obviously weaker than (2.6). We refer to this extension as using *unsigned filter entries* and discuss its impact in Section 3.4. Note that it does not affect the prefiltering technique just discussed.

2.4.3.4 The filter margin The default choice in FILTRANE are

$$\delta(\|\theta_\ell\|, \|\theta_k^+\|) = \|\theta_\ell\| \text{ and } \gamma_\theta = \min\left[\epsilon_\theta, \frac{1}{2\sqrt{p}}\right],$$

where $\epsilon_\theta = 0.001$ by default, but the two other choices for δ can be specified by the user. The effect of an alternative choice of δ or ϵ_θ is discussed in Section 3.4.

2.4.3.5 Grouping and balancing the violations Gould et al. (2003*a*) pointed out that the constraint could be grouped in (potentially overlapping) subsets for which constraint violation would then be measured as a whole, using the Euclidean norm of the vector containing all the violations of the constraints in the group. FILTRANE provides a mechanism to specify these subsets, either automatically or according to a user's preference. Furthermore, the automatic grouping can be chosen as to approximately balance between the groups the aggregate violations at the starting point. When grouping is used, the dimension of the "filter space" falls from $m+q$ to the number of groups. Note that unsigned filter entries are not available for groups containing more than one constraint.

2.4.4 Preconditioning and stopping

At each iteration, the subproblem solution may be preconditioned by a positive matrix M_k (which amounts to specifying the trust-region norm $\|\cdot\|_k = \sqrt{\langle\cdot, M_k \cdot\rangle}$, see Conn et al., 2000, Section 6.7). Besides using no preconditioner at all (i.e. using $M_k = I$ and the Euclidean norm to define the trust region), FILTRANE also provides the choice of diagonal preconditioning, or preconditioning using the a band submatrix of adjustable semi-bandwidth that is extracted from the problem's Hessian. Both the diagonal and the banded submatrix are modified to make them positive definite if necessary (see Gould et al., 2003*c*).

FILTRANE is successfully terminated as soon as

$$\|\theta(x_k)\|_\infty \leq \epsilon_T \text{ or } \|\nabla_x f(x_k)\|_{[k]} \leq \epsilon_G \sqrt{n},$$

where the default values are $\epsilon_T = \epsilon_G = 10^{-6}$, and where $\|\cdot\|_{[k]} = \sqrt{\langle\cdot, M_k^{-1}\cdot\rangle}$ is the dual norm of $\|\cdot\|_k$ (see Conn et al., 2000, Section 2.3.1). Observe that this choice makes the stopping criterion dependent on preconditioning, which is justified by the observation that termination is indeed best decided for the scaled problem. On the other hand, this prevents directly comparing variants using different preconditioners.

2.4.5 Subproblem accuracy

The subproblem of minimizing m_k subject to (2.3) can be solved more or less exactly. In FILTRANE, the default setting is to stop the conjugate-gradient/Lanczos process as soon as

$$\|\nabla_x m_k(x_k+s)\| \leq \min\left[\epsilon_{GLTR}, \max\left[\|\nabla_x m_k(x_k)\|^{\epsilon_R}, \sqrt{\epsilon_M}\right]\right] \|\nabla_x m_k(x_k)\|, \qquad (2.15)$$

or

$$\|\nabla_x m_k(x_k+s)\| \leq \min\left[\sqrt{\epsilon_M}, \tfrac{1}{2}\epsilon_G\sqrt{n}\right] \qquad (2.16)$$

where, by default, $\epsilon_{GLTR} = 0.01$ and $\epsilon_R = 1$, and where ϵ_M is the machine precision. The effect of loosening or tightening this requirement is discussed in Section 3.3.

2.4.6 Other issues

In an attempt to make trial points acceptable as often as possible without compromising the global convergence properties of the algorithm, Gould et al. (2003*a*) also suggested that x_k^+ may be deemed acceptable whenever the reduction in the objective function is at least as large as some fraction of its value. This possibility is offered as an option in FILTRANE: if it is activated by the user, the trial point is then accepted if

$$\|\theta_k\| - \|\theta_k^+\| \geq 0.1 \min\left[1, \|\theta_k\|^{\epsilon_W}\right]. \qquad (2.17)$$

The effect of using this option is discussed in Section 3.5.

Finally, some constants related to trust-region management remain to be defined. Inspired by Conn et al. (2000), Section 17.1, our implementation uses the constants

$$\gamma_0 = 0.0625, \quad \gamma_1 = 0.25, \quad \gamma_2 = 2, \quad \eta_1 = 0.01, \quad \eta_2 = 0.9, \quad \Delta_0 = 1,$$

FILTRANE is written as a standard Fortran 90 module, integrated in the GALAHAD library (see Gould et al., 2003*c*). The user interface uses reverse communication, i.e. returns control to the user whenever user-defined preconditioner must be applied or function/derivative information is requested.

3 Numerical experience

We now turn to the discussion of the numerical experience with FILTRANE, with particular emphasis on the advantages and drawbacks of its various algorithmic options. To conduct these experiments, we selected 122 significant problems from the CUTEr collection of test problems (see Gould, Orban and Toint, 2003*b*). Table A.1 reports the names and characteristics of these problems. In these tables, the column heading $n_{\rm fr}$ indicates the number of free variables, $n_{\rm b}$ the number of variables that are bounded on one side (above or below), $n_{\rm r}$ the number of variables that are bounded both from above and below (often called "range" variables), and $n_{\rm fx}$ the number of fixed variables (problem parameters). The selection provides a variety of cases including small and large problems, linear and nonlinear, equality and/or inequality constrained.

All experiments reported in this section were run on a Dell Latitude C840 portable computer (1.6 MHz, 1Gbyte of RAM) under the Fujitsu frt Fortran compiler with default optimization. All attempts to solve the test problems were limited to a maximum of 1000 iterations or 1 hour of CPU time.

In what follows, we compare several variants of FILTRANE for reliability and efficiency. Remarkably, all test problems except SEMICON1, CHEMRCTB, FLOSP2HM and FLOSP2TM could be solved (within the prescribed iteration and time constraints) by the default variant of FILTRANE or one of its (diagonal or 5-banded) internal preconditioned variants, which indicates good global reliability of the package. The first two failures were caused by arithmetic errors in the comutation of the objective function and the last two by the lack of a suitable preconditioner. Furthermore, detailed analysis showed that some variants were very close to the problem solution despite them reporting that no further progress could be made(1). These occurences are counted as successful in our discussion.

Efficiency comparisons are made after the problems for which different local solutions were found by different variants are removed(2). They use the performance profiles introduced by Dolan and Moré (2002). 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 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 variant 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 variant i was the most effective (according to statistics s_{ij}), $p_i(2)$ gives the fraction of the number for which variant 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 variant succeeded. We consider such a profile to be a very effective means of comparing the relative merits of our algorithmic variants, but have limited the range of the horizontal axis to 10, *de facto* identifying (in the figure) a performance beyond 10 times worse than the best with failure. 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%.

(1)This occured the unpreconditioned variants on problems ARGLBLE and ARGLCLE.

(2)This occured for problem PFIT2, and, for some variants, for problems ARTIF and GROWTH.

3.1 Filter vs. pure trust-region algorithms

We first examine the impact of using the multidimensional filter technique in addition to the trust-region mechanism, by comparing the default version of FILTRANE described above with a variant where the trust-region constraint is enforced at every step and the filter mechanism is not used for deciding on the acceptability of the trial point as a new iterate. The resulting algorithm then conforms to the usual monotone trust-region framework (see Chapter 6 of Conn et al., 2000).

The first observation is that the default FILTRANE is more reliable than the pure trust-region variant, as its solves 110 of the 123 test problems (within the prescribed CPU and iteration limits) while the pure trust-region variant solves 101.

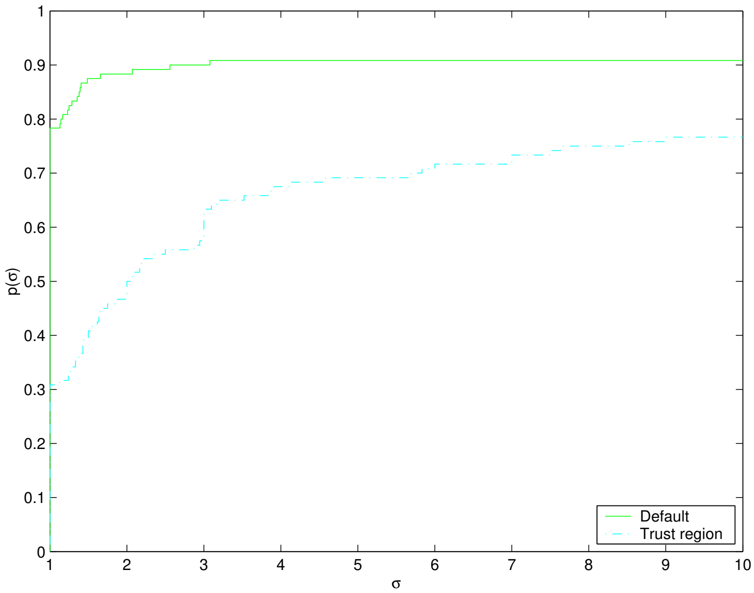

Figure 3.1: Iteration performance profile for the default FILTRANE variant (including filter) and the pure trust-region variant (no filter)

Figures 3.1 and 3.2 also illustrate that the default FILTRANE is typically considerably more efficient on the problems that could be solved by both variants, both in iterations and CPU time. This comparison therefore confirms the findings of Gould et al. (2003*a*).

3.2 Model choice and inertia

We next compare the default FILTRANE with two variants that use the Gauss-Newton model (for the first) or the full Newton model (for the second) at every iteration. In terms of realiability, the default and pure Gauss-Newton variants are best (109/122) while the pure Newton variant is substantially behind (87/122). If we now consider efficiency, the performance profiles presented in Figures 3.3 and 3.4 indicate that the adaptive default strategy is about as efficient as the Gauss-Newton strategy and considerably better than the pure Newton.

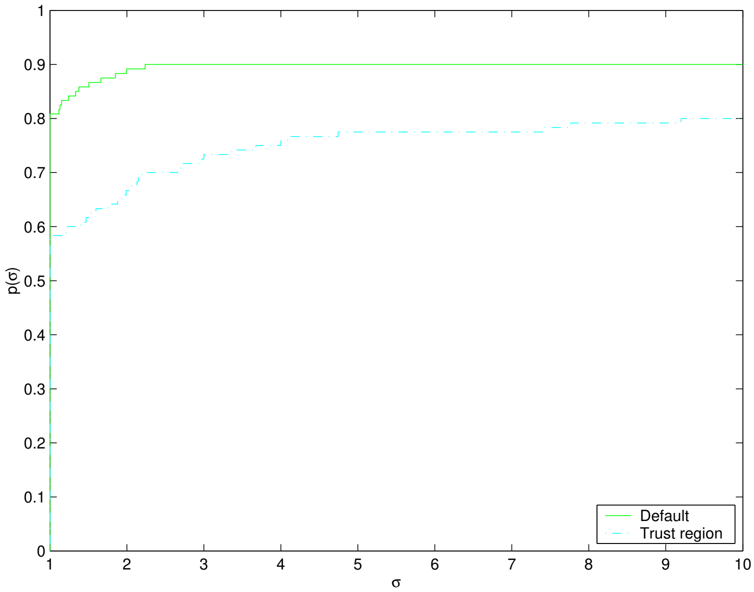

Figure 3.2: CPU time performance profile for the default FILTRANE variant (including filter) and the pure trust-region variant (no filter)

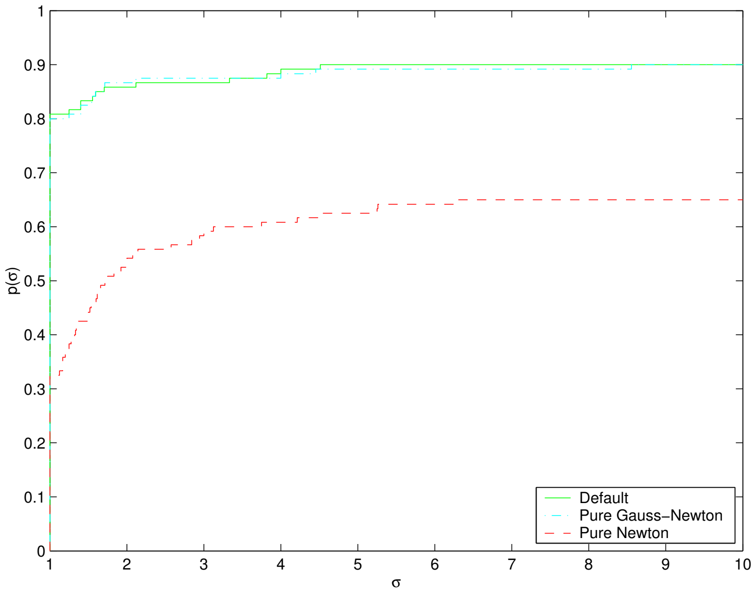

Figure 3.3: Iteration performance profile for the default FILTRANE variant (including adaptive model choice) and the pure Gauss-Newton and Newton variants

Although the unsatisfactory performance of the pure Newton model had already been noticed in Gould et al. (2003*a*) for the solution of sets of nonlinear equations, this characteristic appears to be reinforced when considering problems that involve inequality constraints. The discontinuity of the second derivatives of the objective function (1.3) indeed makes a prediction of objective decrease based on the full Newton model relatively unreliable. Interestingly, this phenomenon does not occur if the Gauss-

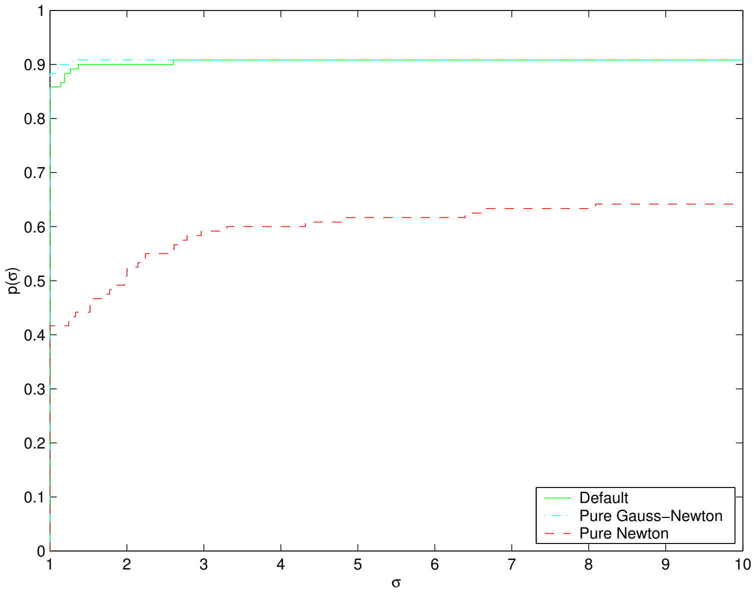

Figure 3.4: CPU time performance profile for the default FILTRANE variant (including adaptive model choice) and the pure Gauss-Newton and Newton variants

Newton model is used because the model then attempts to find a root of the linearized inequality constraints instead of a minimum of its squared violation.

As the default variant uses the adaptive model choice, it is useful to verify that the default choice of inertia ($n_v = 5$) behaves well compared to other values. We therefore tested four additional variants, with $n_v = 3, 4, 6$ and 7. A first observation is that the last 3 of these variants share the same reliability as the default (109/122), while the variant using $n_v = 3$ solves 108 problems. Their relative efficiencies are very comparables, as shown by Figures 3.5 and 3.6.

We also performed the same tests on a variant of FILTRANE that chooses its adaptive model using the "best reduction" criterion (2.13) instead of the default "best fit" (2.12). The reliability of this variant (108/122) is very nearly as good as that of the default, and its performance is comparable, although slighlty worse in terms of iterations, as shown by Figures 3.7 and 3.8.

3.3 Accuracy of the subproblem solution

Another important algorithmic parameter is the minimum reduction in the norm of the model's gradient that is required for terminating the GLTR step calculation. We therefore tested variants with $\epsilon_{GLTR} = 0.1, 0.001$ and $\sqrt{\epsilon_M}$ as well as $\epsilon_R = 0.5$ in (2.15). Some of these variants found different local minima for `GROWTH`, and this problems was therefore excluded from the comparisons reported in this paragraph. The default version (using $(\epsilon_{GLTR}, \epsilon_R) = (0.01, 1)$) proved to be the most reliable (108/121 problems solved), followed by the choices $(\epsilon_{GLTR}, \epsilon_R) = (0.01, 0.5)$ with 107/121 problems solved, $(\epsilon_{GLTR}, \epsilon_R) = (0.001, 1)$ and $(0.001, 0.5)$ with 105/121, and the choices

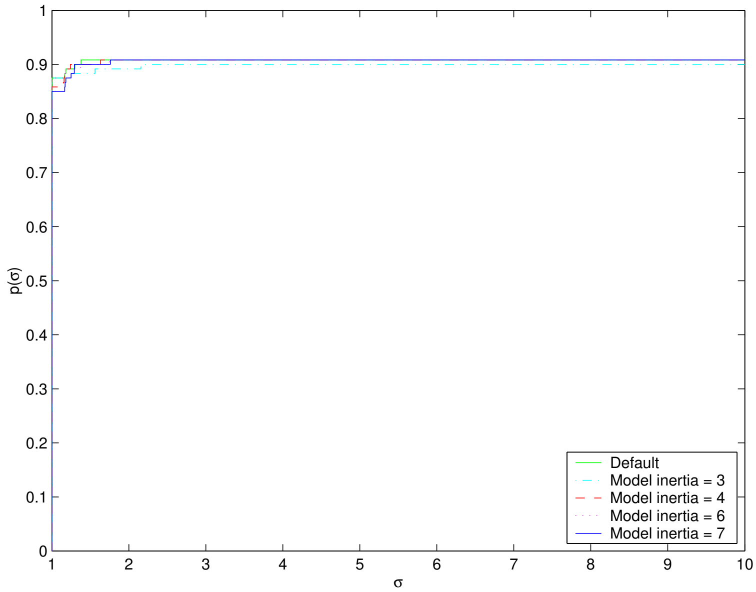

Figure 3.5: Iteration performance profile for the default FILTRANE variant ($n_v = 5$) and variants with neighbouring inertia

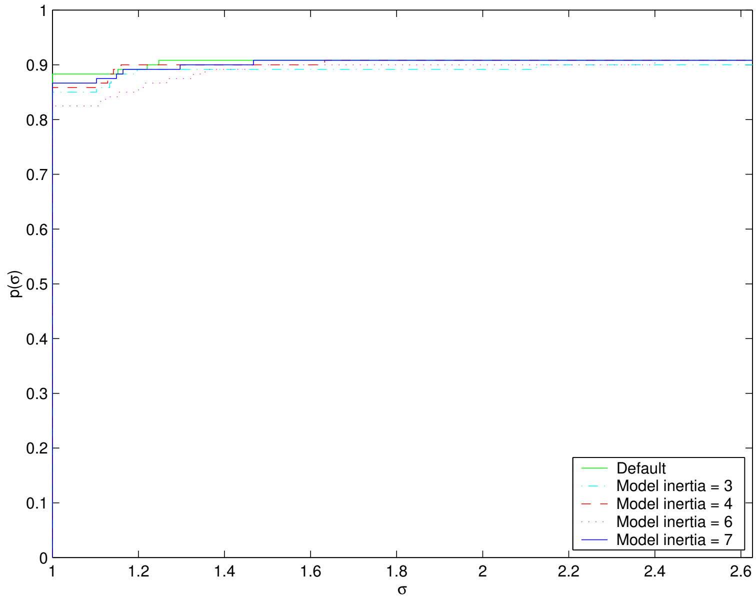

Figure 3.6: CPU time performance profile for the default FILTRANE variant ($n_v = 5$) and variants with neighbouring inertia

$(\epsilon_{GLTR}, \epsilon_R) = (0.1, 1)$ and $(0.1, 0.5)$ with 104/121. The variant using full accuracy $(\epsilon_{GLTR}, \epsilon_R) = (\sqrt{\epsilon_M}, 1))$ solved 98 problems.

The iteration and CPU time performance profiles (Figures 3.9 and 3.10) indicate that the default version and that using $\epsilon_{GLTR} = 0.001$ (and $\epsilon_R = 1$) behave similarly. Requiring full accuracy typically results in a smaller number of iterations but longer CPU time. The looser accuracy choice ($\epsilon_{GLTR} = 0.1$) appears to be globally less

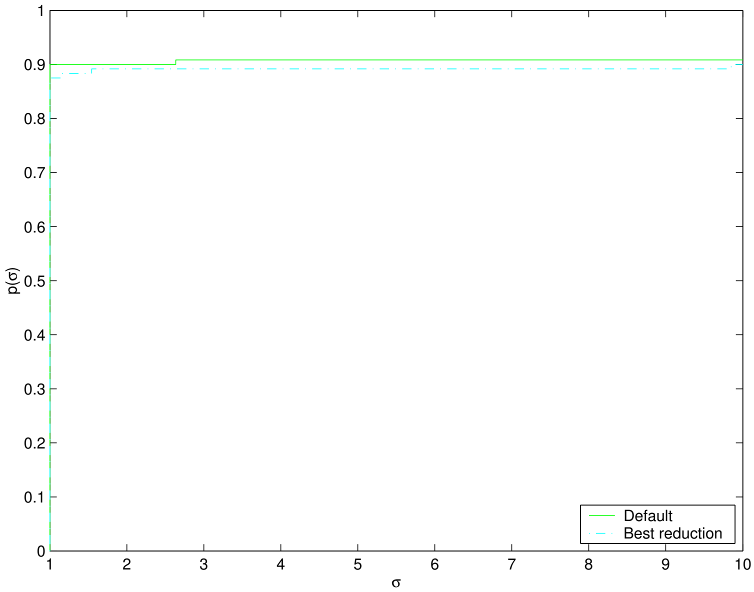

Figure 3.7: Iteration performance profile for the default "best fit" and the "best reduction" FILTRANE variants

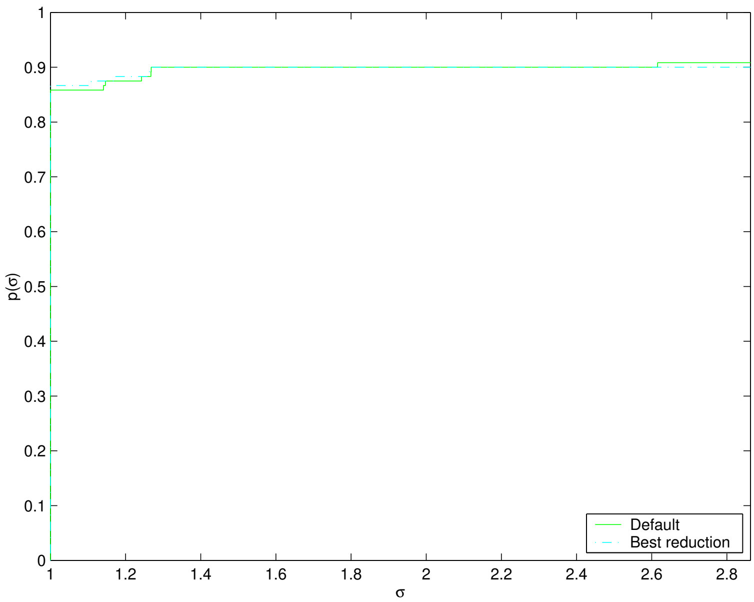

Figure 3.8: CPU time performance profile for the default "best fit" and the "best reduction" FILTRANE variants

efficient. The full-accuracy version excels in terms of iteration numbers, but pays a heavy price in computing time.

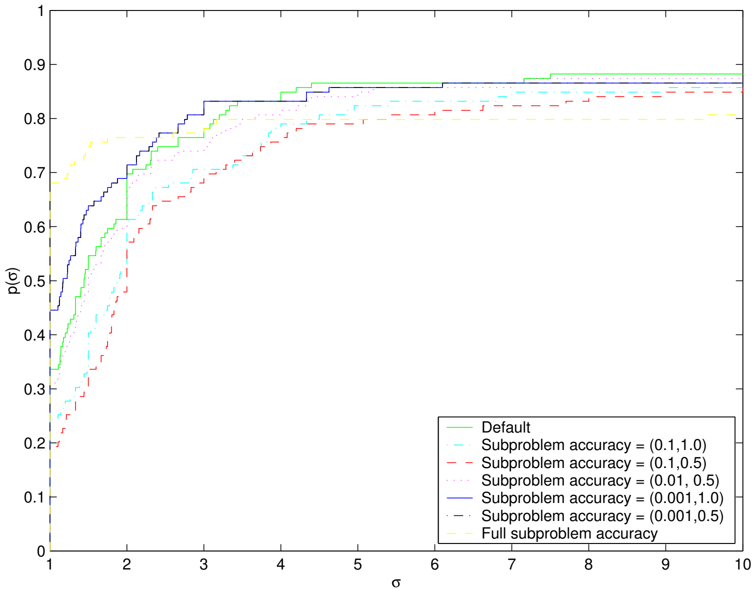

Figure 3.9: Iteration performance profile for the FILTRANE variants depending on the requested subproblem accuracy

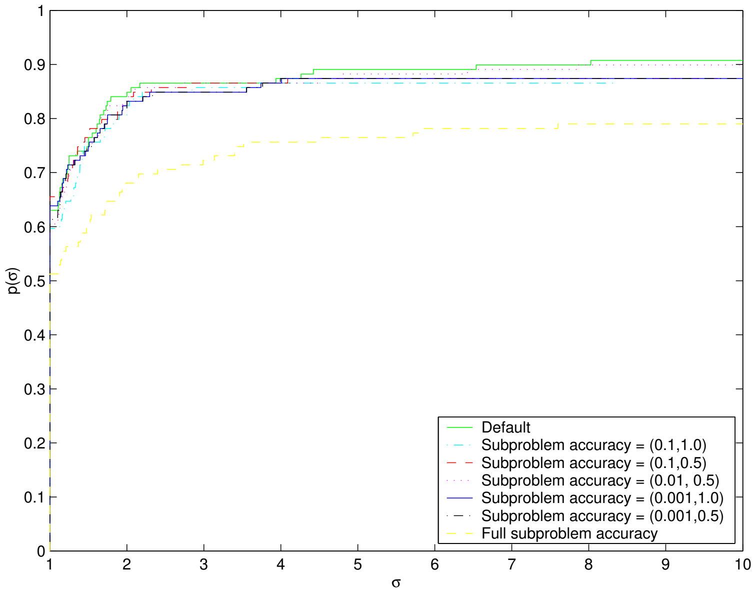

Figure 3.10: CPU time performance profile for the FILTRANE variants depending on the requested subproblem accuracy

3.4 Filter management

We now turn to the numerical appraisal of the various filter management issues, and start with the value of $\tau_{\max}$, the maximal trust-region relaxation parameter. The default value $\tau_{\max} = 1000$ again provides the best reliability together with the choice $\tau_{\max} = 10000$, but the difference is slight with the variant using $\tau_{\max} = 100$, which solves 108 of the 122 problems. The choice $\tau_{\max} = 1$, which amounts to imposing the trust-region constraint (although using the filter to accept new iterates) is less reliable (103/122). These conclusions are reinforced by the performances profiles of Figures 3.11 and 3.12.

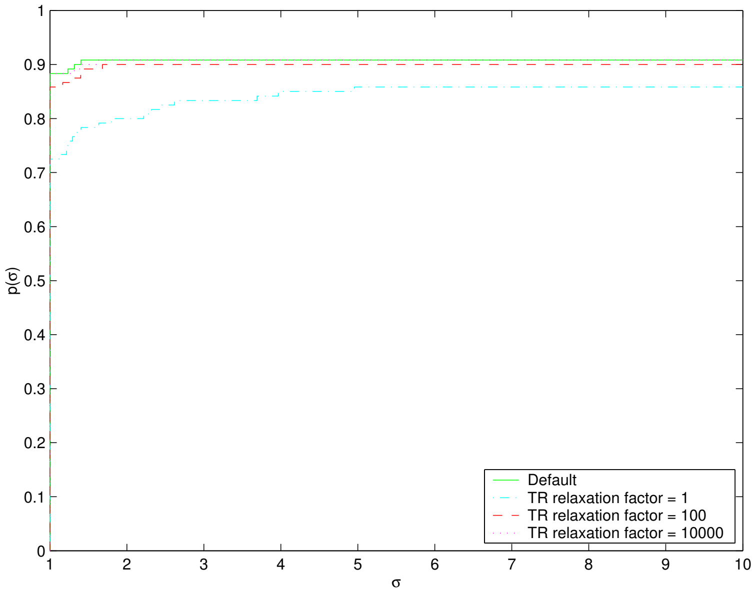

Figure 3.11: Iteration performance profile for the FILTRANE variants depending on the trust-region relaxation factor

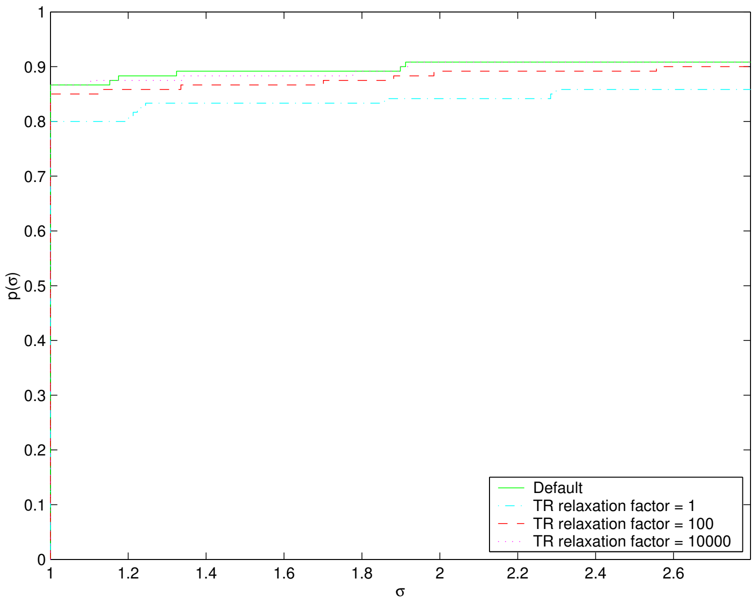

Figure 3.12: CPU time performance profile for the FILTRANE variants depending on the trust-region relaxation factor

Interestingly, enforcing the trust-region constraint ($\tau_{\max} = 1$) seems to be globally advantageous if one wishes to reduce the number of filter entries, as is shown in

Figure 3.13.

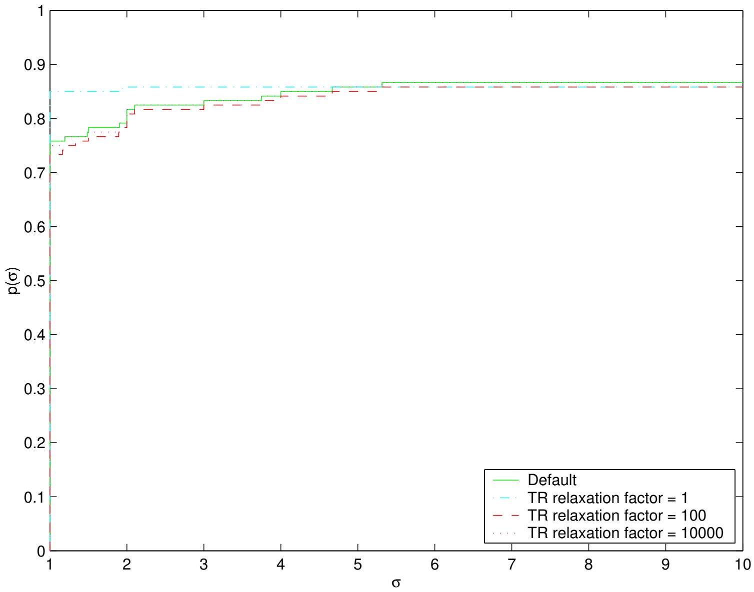

Figure 3.13: Filter size performance profile for the FILTRANE variants depending on the trust-region relaxation factor

We next consider the numerical effect of extending the definition of filter entries by allowing them to be unsigned (see (2.14)). Our experiments show a slightly increased reliability (109/122 versus 106/122) with a modest gain in efficiency, both in iterations and time, as indicated in Figures 3.14 and 3.15. Figure 3.16 shows that this gain is obtained at the cost of including more entries in the filter, as can be expected.

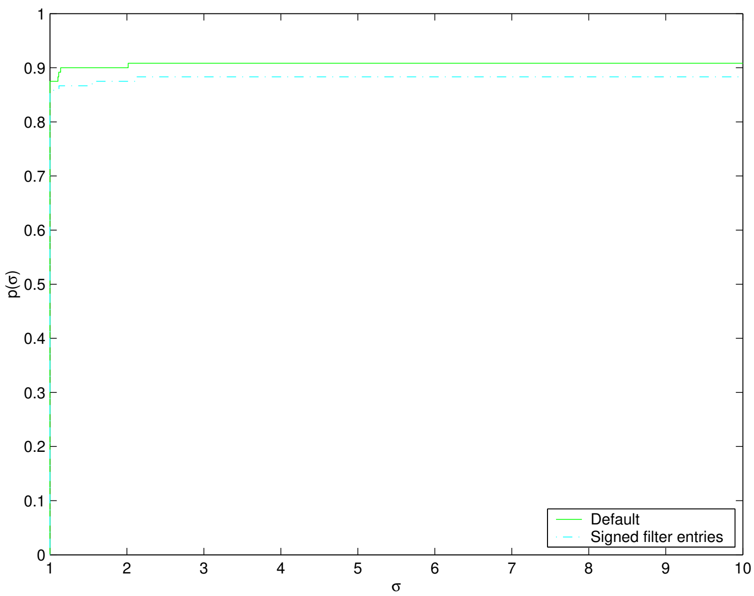

Figure 3.14: Iteration performance profile for the FILTRANE variants using signed or unsigned filter entries

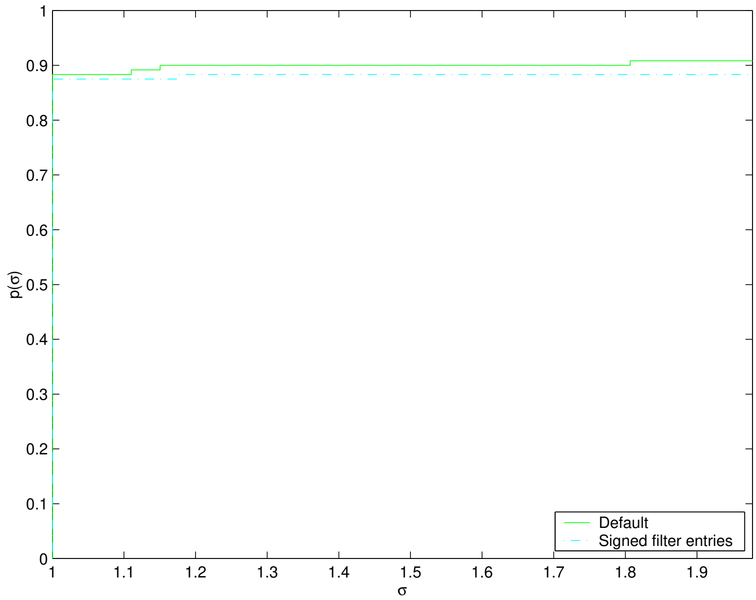

Figure 3.15: CPU time performance profile for the FILTRANE variants using signed or unsigned filter entries

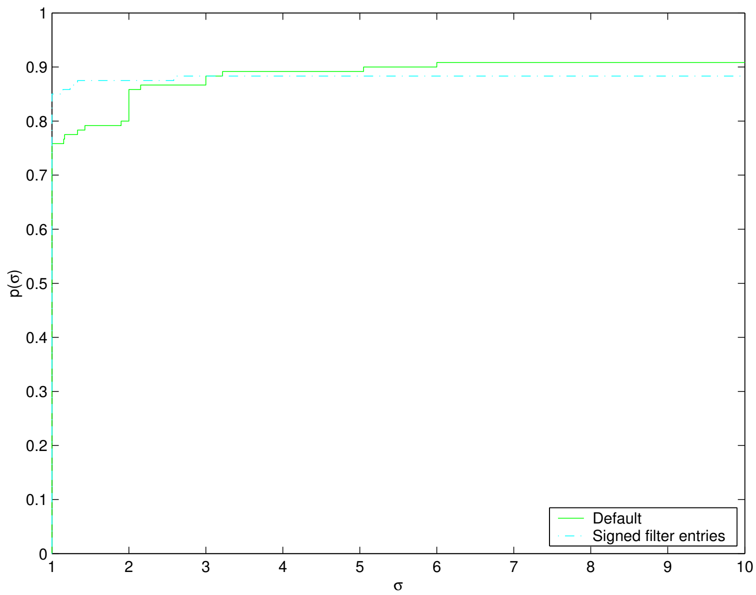

Figure 3.16: Filter size performance profile for the FILTRANE variants using signed or unsigned filter entries

In our default variant, we selected the choice (2.7), which can be seen as using the filter entry violation to prescribe the filter margin size. We nevertheless tested variants using (2.8) (using the current violation instead), and (2.9) (using the smallest of these two violations). The appear to be slightly less reliable (107/122), and, in Figures 3.17, 3.18 and 3.19, to be marginally less efficient.

We complete our investigation of filter management by considering the impact of

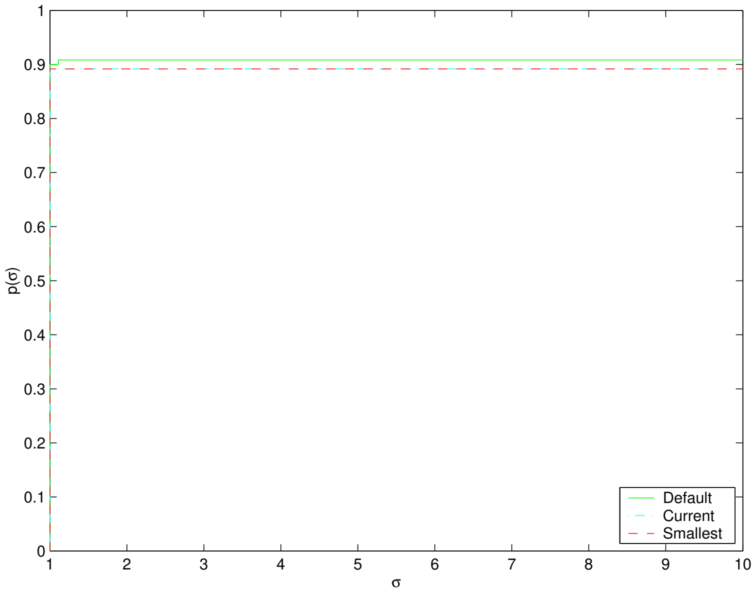

Figure 3.17: Iteration performance profile for the FILTRANE variants according to the choice of the violation that prescribes the margin size

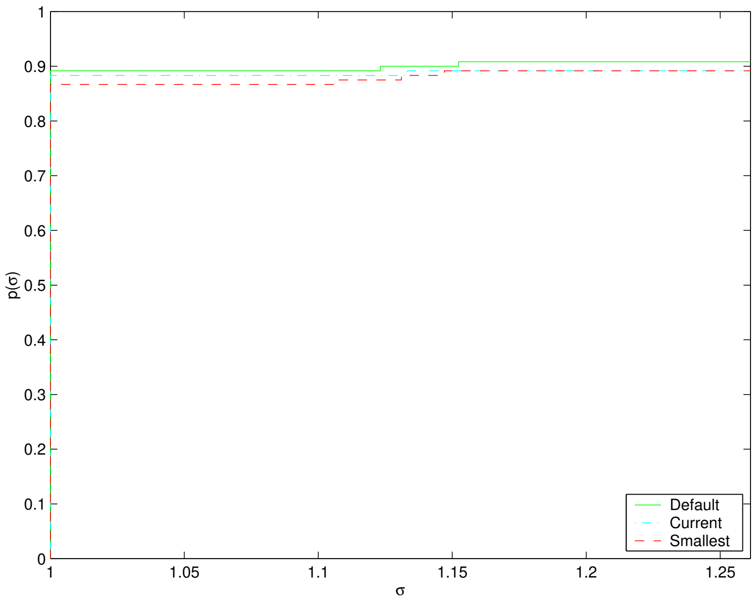

Figure 3.18: CPU time performance profile for the FILTRANE variants according to the choice of the violation that prescribes the margin size

the choice of the filter margin constant ϵ_θ. The default variant uses $\epsilon_\theta = 0.001$, but we also tested variants with $\epsilon_\theta = 0.1, 0.01$ and 0.0001. The default choice once more provides the best reliability, but all other choices still solved 108/122 problems. The excellent performance of the default version is illustrated in Figures 3.20 and 3.21, but the differences between variants remain small.

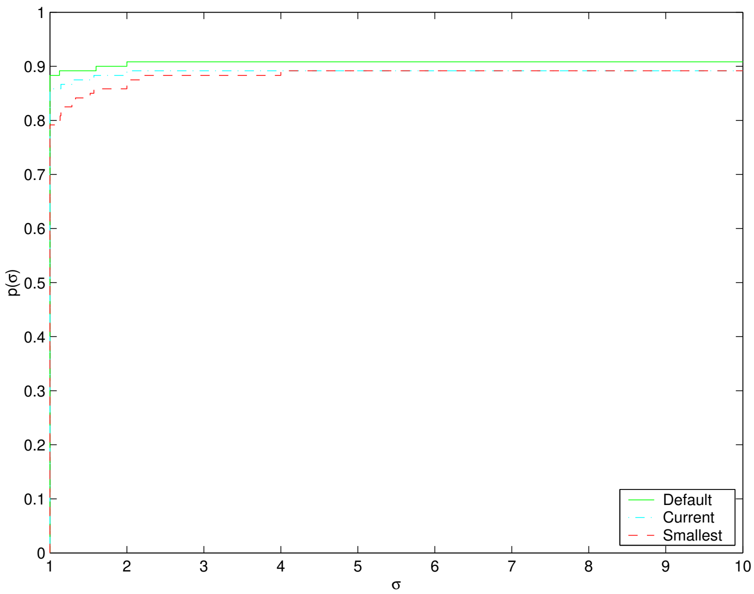

Figure 3.19: Filter size performance profile for the FILTRANE variants according to the choice of the violation that prescribes the margin size

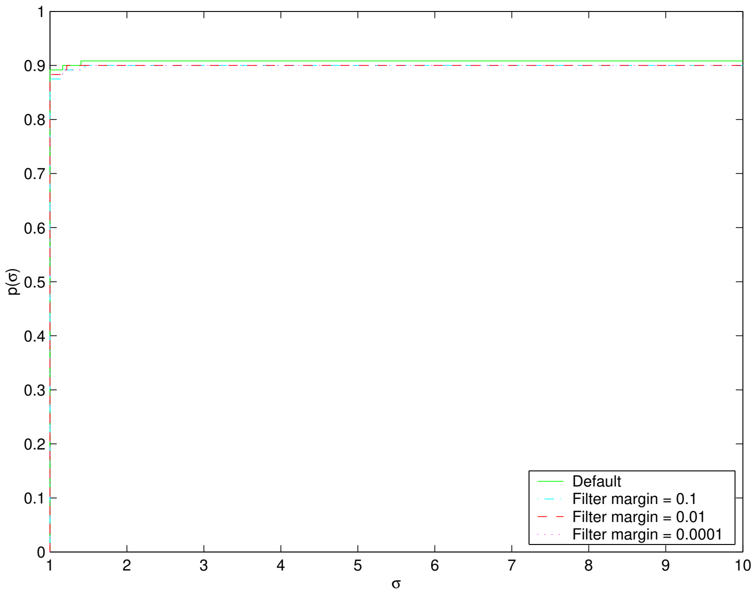

Figure 3.20: Iteration performance profile for the FILTRANE variants depending on the filter margin constant

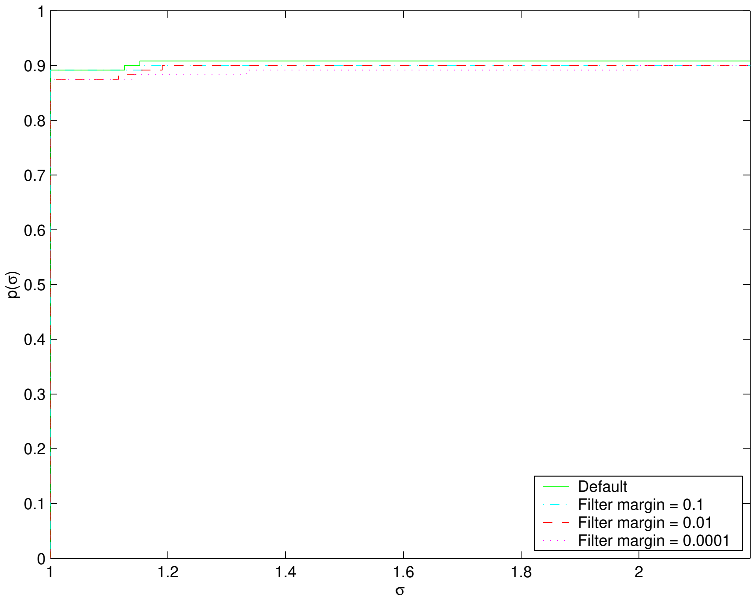

Figure 3.21: CPU time performance profile for the FILTRANE variants depending on the filter margin constant

3.5 Weak acceptance criterion

The weak acceptance rule (2.17) does not appear to bring any improvement since the default variant compares favourably with the variants that use it with $\epsilon_W = 1$, 2 or 3, both in reliability (109/122 versus 107, 106 and 108, respectively) and efficiency (see Figures 3.22 and 3.23).

3.6 Preconditioning

Although clearly crucial in practice, the question does not lend itself to much discussion here, since we have mentioned the fact that different preconditioning-dependent stopping criteria make efficiency comparisons hard to interpret. We may compare the reliability scores of the default (unpreconditioned) variant (109/122) with its diagonally preconditioned version (103/122) or its variant using a banded matrix of semi-bandwidth 5 (107/122), but these measures obscure the fact that some problems (`GLIDER`, `CAMSHAPE`, `ROCKET`, `FLOSP2TL`, `ROTDISC`, `CHEMRCTA`, `FLOSP2HH` and `FLOSP2TL`) simply require preconditioning to be solved, while preconditioning prevents convergence on others (`POWELLSQ`, `TRAINF`, `CORKSCRW`, `YATP2SQ`). Experience with specific classes of problems therefore remains the ultimate deciding factor, but the fact that FILTRANE allows preconditioning (user-defined included) is clearly valuable.

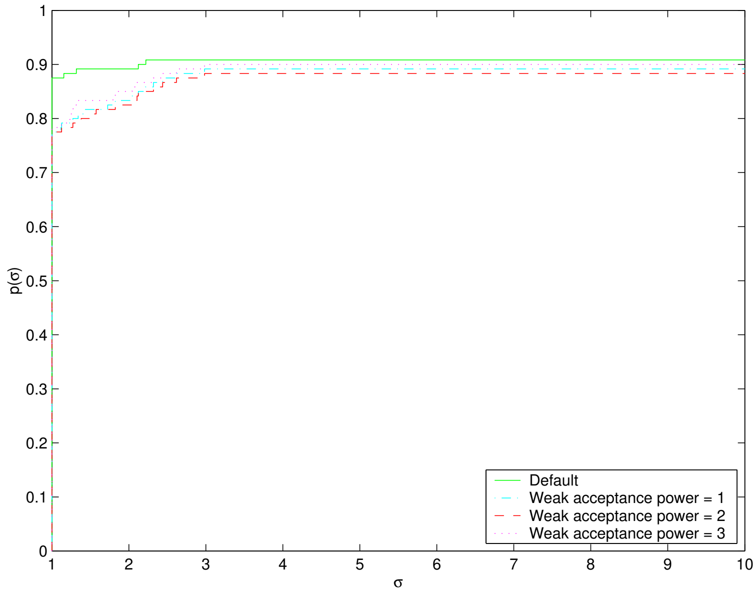

Figure 3.22: Iteration performance profile for the FILTRANE variants depending on the trial point acceptance rule

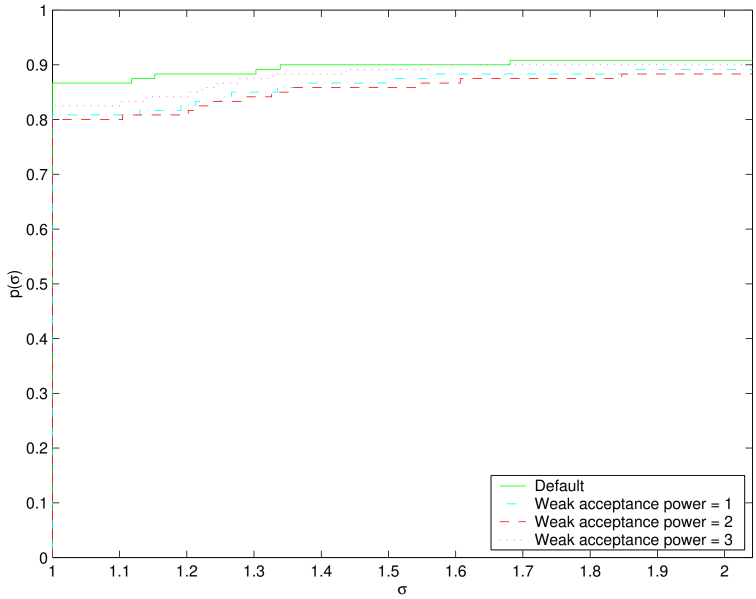

Figure 3.23: CPU time performance profile for the FILTRANE variants depending on the trial point acceptance rule

3.7 Other issues

We conclude our experimental analysis of the FILTRANE package by briefly mentioning some remaining issues.

We also investigated whether keeping dominated filter entries in the filter might save time at the expense of memory, but we could not isolate any significant difference, possibly because the pre-filtering technique described above is already fairly efficient in saving time in the process of comparing trial and filter points.

We finally tested grouping equations or inequalities and balancing those groups, as described in Gould et al. (2003*a*), but obtained results entirely parallel to those reported in that reference. These indicate that keeping as many groups as possible appears beneficial, and that the effect of balancing the groups is limited on the CUTEr test problems.

4 Conclusion

We have presented FILTRANE, a new Fortran 95 package for the solution of the nonlinear feasibility problem, and have shown that the main feature of the underlying algorithm (use of a mutidimensional unsigned filter) produces significant reliability and efficiency gains compared to a more classical trust-region approach. FILTRANE is available as part of the GALAHAD library at `http://galahad.rl.ac.uk/galahad-www/`.

Extensive numerical experience was also used to investigate the dependence of the package on some of its algorithmic constants, which was shown to be very moderate. This investigation also provided the opportunity to study, for the first time, the relative importance of some parameters that are imbedded in filter methods, and to indicate that such methods may not depend too strongly on suitable choices for these parameters.

As always, the true potential of the FILTRANE package will only be correctly assessed with its continued use in a variety of applications, but the results presented here are clearly encouraging.

Acknowledgements

Ph. Toint would like to express his gratitude to the University of Canterbury Erskine Fellowship for its support during the spring 2003, as well as his thanks to W. Baritompa for his interest and stimulating discussions on filters and trust regions. Some of this research was sponsored by EPSRC grants GR/R46641 and GR/S02969/01.

References

A. R. Conn, N. I. M. Gould, and Ph. L. Toint. LANCELOT*: a Fortran package for large-scale nonlinear optimization (Release A).* Number 17 *in* 'Springer Series in Computational Mathematics'. Springer Verlag, Heidelberg, Berlin, New York, 1992.

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 and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1983. Reprinted as *Classics in Applied Mathematics 16*, SIAM, Philadelphia, USA, 1996.

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

R. Fletcher and S. Leyffer. User manual for filterSQP. Numerical Analysis Report NA/181, Department of Mathematics, University of Dundee, Dundee, Scotland, 1998.

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

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 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, S. Leyffer, and Ph. L. Toint. A multidimensional filter algorithm for nonlinear equations and nonlinear least-squares. Technical Report TR-2003-004, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2003*a*.

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. *Transactions of the ACM on Mathematical Software*, **29**(4), (to appear), 2003*b*.

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), (to appear), 2003*c*.

J. J. Moré and D. C. Sorensen. Newton's method. *in* G. H. Golub, ed., 'Studies in Numerical Analysis', number 24 *in* 'MAA Studies in Mathematics', pp. 29–82, Providence, Rhode-Island, USA, 1984. American Mathematical Society.

J. Nocedal. Trust region algorithms for solving large systems of nonlinear equations. *in* W. Liu, T. Belytschko and K. C. Park, eds, 'Innovative Methods for Nonlinear Problems', pp. 93–102. Pineridge Press, 1984.

J. M. Ortega and W. C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*. Academic Press, London, 1970.

Ph. L. Toint. Numerical solution of large sets of algebraic nonlinear equations. *Mathematics of Computation*, **46**(173), 175–189, 1986.

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

Test problem characteristics

Problem	$n_{\rm fr}$	$n_{\rm b}$	$n_{\rm r}$	$n_{\rm fx}$	m	q
AIRCRFTA	2	0	0	3	5	0
ARGAUSS	3	0	0	0	15	0
ARGLALE	200	0	0	0	400	0
ARGLBLE	200	0	0	0	400	0
ARGLCLE	200	0	0	0	399	0
ARGTRIG	200	0	0	0	200	0
ARTIF	4998	0	0	2	5000	0
ARWDHNE	500	0	0	0	998	0
BATCH	0	2	46	0	12	61
BDVALUE	100	0	0	2	100	0
BDVALUES	10000	0	0	2	10000	0
BOOTH	2	0	0	0	2	0
BRATU2D	4900	0	0	284	4900	0
BRATU2DT	4900	0	0	284	4900	0
BRATU3D	3375	0	0	1538	3375	0
BROYDN3D	5000	0	0	0	5000	0
BROYDNBD	5000	0	0	0	5000	0
BROWNALE	200	0	0	0	199	0
CAMSHAPE	800	0	0	0	0	1603
CBRATU2D	2888	0	0	312	2888	0
CBRATU3D	2000	0	0	1456	2000	0
CHAIN	800	0	0	2	400	0
CHANDHEQ	100	0	0	0	100	0
CHANNEL	9598	0	0	2	9598	0
CHEMRCTA	5000	0	0	0	5000	0
CHEMRCTB	5000	0	0	0	5000	0
CHNRSBNE	50	0	0	0	98	0
CLNLBEAM	5001	0	9998	0	10000	0
CLUSTER	2	0	0	0	2	0
COOLHANS	9	0	0	0	9	0

Table A.1: The test problems and their characteristics (continued)

Problem	n_{fr}	n_{b}	n_{r}	n_{fx}	m	q
CORKSCRW	2497	0	2000	9	3000	500
CUBENE	2	0	0	0	2	0
DECONVNE	61	0	0	0	40	0
DRCAVTY1	961	0	0	264	961	0
DRCAVTY2	3969	0	0	520	3969	0
DRCAVTY3	961	0	0	264	961	0
DRUGDISE	100	300	199	4	500	0
EIGENA	110	0	0	0	110	0
EIGENB	110	0	0	0	110	0
EIGENC	462	0	0	0	462	0
EIGMAXA	101	0	0	0	101	0
EIGMAXB	101	0	0	0	101	0
EIGMAXC	202	0	0	0	202	0
EIGMINA	201	0	0	0	201	0
EIGMINB	301	0	0	0	301	0
EIGMINC	302	0	0	0	302	0
FEEDLOC	0	0	87	3	19	240
FLOSP2HH	2763	0	0	120	2761	0
FLOSP2HM	2763	0	0	120	2761	0
FLOSP2HL	2763	0	0	120	2761	0
FLOSP2TM	2763	0	0	120	2761	0
FLOSP2TL	803	0	0	64	803	0
GASOIL	10398	3	0	2	10398	0
GAUSSELM	20501	39	1599	0	61542	0
GLIDER	1006	200	101	7	1208	0
GOTTFR	2	0	0	0	2	0
GROWTH	3	0	0	0	12	0
HAGER4	2500	2500	0	1	2500	0
HATFLDF	3	0	0	0	3	0
HATFLDG	25	0	0	0	25	0
HEART6	6	0	0	0	6	0
HEART8	8	0	0	0	8	0
HELSBY	741	649	18	0	1399	0
HIMMELBA	2	0	0	0	2	0
HIMMELBC	2	0	0	0	2	0
HIMMELBD	2	0	0	0	2	0
HIMMELBE	3	0	0	0	3	0
HYDCAR6	29	0	0	0	29	0
HYDCAR20	99	0	0	0	99	0
HYPCIR	2	0	0	0	2	0

Table A.1: The test problems and their characteristics (continued)

Problem	n_{fr}	n_{b}	n_{r}	n_{fx}	m	q
INTEGREQ	500	0	0	0	500	0
JUNKTURN	9996	0	0	14	7000	0
LEAKNET	80	70	6	0	153	0
LEWISPOL	0	0	6	0	9	0
MANNE	0	4749	1250	1	0	4000
MARINE	11200	15	0	0	11192	0
METHANB8	31	0	0	0	31	0
METHANL8	31	0	0	0	31	0
METHANOL	11997	5	0	3	11997	0
MRIBASIS	0	24	0	12	51	3
MSQRTA	1024	0	0	0	1024	0
MSQRTB	1024	0	0	0	1024	0
NGONE	0	496	1	3		31373
NONMSQNE	49	0	0	0	49	0
NYSTROM5	15	0	0	3	18	0
OPTMASS	3006	0	0	4	2004	501
OPTCDEG2	1500	1499	1500	3	1500	1500
ORTHREGA	8197	0	0	0	4096	0
ORTHREGD	10003	0	0	0	5000	0
ORTHREGF	30033	2	0	0	10000	0
PFIT1	1	0	2	0	3	0
PFIT2	1	0	2	0	3	0
PFIT3	1	0	2	0	3	0
PFIT4	1	0	2	0	3	0
PINENE	8795	5	0	5	8795	0
POLYGON	0	198	0	2	0	5049
POROUS1	3844	0	0	252	3844	0
POROUS2	3844	0	0	252	3844	0
POWELLBS	2	0	0	0	2	0
POWELLSQ	2	0	0	0	2	0
PT	2	0	0	0	0	501
QR3D	590	20	0	0	610	0
QR3DBD	1552	33	0	0	1650	0
RECIPE	3	0	0	0	2	0
ROBOTARM	1999	2400	0	12	3202	0
ROCKET	802	801	800	4	2002	0
ROTDISC	180	181	531	13	360	721
RSBRNE	2	0	0	0	2	0
SEMICON1	5000	0	0	2	5000	0
SEMICON2	5000	0	0	2	5000	0

Table A.1: The test problems and their characteristics (continued)

Problem	n_{fr}	n_{b}	n_{r}	n_{fx}	m	q
SINVALNE	2	0	0	0	2	0
SMMPSF	0	720	0	0	240	23
SNAKE	2	0	0	0	0	2
SPMSQRT	10000	0	0	0	16664	0
STOCFOR3	0	15965	0	0	8829	7846
STEERING	1597	0	401	7	1600	0
TRAINF	2000	0	2000	8	2002	0
TRIGGER	6	0	0	1	6	0
TRUSPYR1	3	8	0	0	3	0
TWIRIBG1	0	0	3127	0	922	317
VANDERM1	100	0	0	0	199	0
VANDERM2	100	0	0	0	199	0
VANDERM3	100	0	0	0	199	0
WOODSNE	4000	0	0	0	3001	0
YATP1SQ	123200	0	0	0	123200	0
YATP2SQ	123200	0	0	0	123200	0
YFITNE	3	0	0	0	17	0
ZANGWIL3	3	0	0	0	3	0

Table A.1: The test problems and their characteristics