Dealing with singularities in nonlinear unconstrained optimization

M. Bierlaire M. Thémans[*]

December 7, 2006

Report TRANSP-OR-061207
TRANSP-OR Transport And Mobility Laboratory
School of Architecture, Civil and Environmental Engineering
Ecole Polytechnique Fédérale de Lausanne.

Abstract

We propose new trust-region based optimization algorithms for solving unconstrained nonlinear problems whose second derivatives matrix is singular at a local solution. We give a theoretical characterization of the singularity in this context and we propose an iterative procedure which allows to identify a singularity in the objective function during the course of the optimization algorithm, and artificially adds curvature to the objective function. Numerical tests are

[*]Ecole Polytechnique Fédérale de Lausanne, School of Architecture, Civil and Environmental Engineering, TRANSP-OR Transport and Mobility Laboratory, CH-1015 Lausanne, Switzerland. Email: {michel.bierlaire,michael.themans}@epfl.ch

performed on a set of unconstrained nonlinear problems, both singular and non-singular. Results illustrate the significant performance improvement compared to classical trust-region and filter algorithms proposed in the literature.

1 Introduction

We consider a nonlinear unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x) \tag{1}$$

where f is a twice differentiable function, possibly nonconvex. The most efficient methods to identify a local minimum of (1) are variants of Newton's method, based on globalization techniques such as linesearch and trust-region methods, as described in many textbooks including Dennis and Schnabel, 1983, Nocedal and Wright, 1999, Bertsekas, 1999 and Bierlaire, 2006.

The convergence analysis of these algorithms assumes that the curvature of the objective function at the solution x^* is bounded away from 0, that is the second derivatives matrix is positive definite at x^*.

However, this major assumption cannot always be guaranteed in practice. This is typically the case for the maximum likelihood estimation of the parameters of econometrics model. In this context, the source of singularity is twofold. On the one hand, a lack of variability in the data may preclude the identification of some parameters. On the other hand, some advanced models require complicated normalization which cannot be performed, imposing the estimation of an unidentified model (see Walker, 2001 and Thémans and Bierlaire, 2006).

In the presence of singularity, not only the convergence theory cannot be applied as such anymore, but significant deterioration of the algorithm performance is observed. In this paper, we propose a variant to existing trust-region and filter trust-region methods in order to deal with this issue.

2 Literature review

The convergence theory of Newton-like methods guarantees local quadratic convergence if the eigenvalues of the second derivatives matrix of the iterates are bounded away from 0. Griewank and Osborne (1983) have shown that if a problem is singular for an algorithm, the iterates produced are at best linearly convergent (even if the second derivatives matrix is singular only at the solution, and not at all iterates). Furthermore, when solving singular problems, standard methods can encounter numerical problems as the curvature of the function converges towards zero.

In the literature, singular problems have been mainly considered in the context of solving systems of nonlinear equations (see, for instance, Decker and Kelley, 1980, Decker et al., 1983, Griewank and Osborne, 1983, Schnabel and Frank, 1984, Griewank, 1985 and Izmailov and Solodov, 2002). Decker and Kelley (1980) have worked on the theoretical implications of singularity in the Jacobian of the system at a local solution. They have shown that the convergence deteriorates and can be proved to be asymptotically linear of ratio $\frac{2}{3}$ for some classes of singular systems. Griewank and Osborne (1983) have analyzed the behavior of Newton's method near singularities in the Jacobian. In the singular case, Newton's method can either converge with a limiting linear ratio, or diverge from arbitrarily closed starting points or even behave chaotically. Decker et al. (1983) have analyzed in details the linear convergence rates of Newton's method on several classes of singular problems. They also propose a modification of the method, constraining iterates in regions where the Jacobian is invertible, which allows to restore the quadratic rate of convergence for some of these classes of singular problems. Schnabel and Frank (1984) have introduced a new class of methods, called tensor methods, for solving systems of nonlinear equations. Tensor methods are particularly well adapted when the Jacobian matrix at the solution is singular or badly conditioned. The main idea is to consider a quadratic model instead of using the classical linear model as in Newton-like methods. The second-order term of this new model is determined such that the model interpolates the function values at several previous iterates, as well as the function value and its gradient value at the current iterate. Griewank (1985) also proposed a quadratical model in order to deal with singular solutions. Moreover, two modifications of the Newton's recurrence scheme are proposed to solve singular problems more efficiently. Izmailov and Solodov (2002) have proposed a new algorithm to solve singular problems such as smooth reformulations of nonlinear complementarity problems. The idea is to regularize a singular solution $\bar{x}$ by adding another term to the left-hand size, which vanishes at $\bar{x}$ (so that $\bar{x}$

remains a solution), and such that its Jacobian at $\bar{x}$ "compensates" for the singularity. They suggest to base this extra term on the information about the derivative of the system.

In the context of unconstrained optimization, Schnabel and Chow (1991) have proposed to use tensor methods as an adaptation of tensor methods for systems of nonlinear equations. Tensor methods dedicated to optimization construct a fourth-order model using third and fourth derivatives tensors of the objective function f. These higher-order derivatives allow to deal with singularity in the second derivatives matrix at local solutions.

In the next section, we give a characterization of the singularity and a procedure which allows to identify this singularity during the course of the optimization algorithm. We present in Section 4 a class of algorithms designed to deal with singular problems in an efficient way. Based on the trust-region framework, it is able to accomodate advanced variants based on preconditioning and filter.

3 Characterization and identification of the singularity

Due to the possible non-convexity of f in (1), the objective function may exhibit several local optima. Some of them may correspond to a singular second derivatives matrix, but not necessarily all of them. We are interested here in the case where a given algorithm a converges to a singular local optimum. Consequently, we say that problem (1) is *singular* for algorithm a if the algorithm generates a sequence x_k, converging to x^* such that $\nabla f(x^*) = 0$, $\nabla^2 f(x^*)$ is semi positive definite, and $\nabla^2 f(x^*)$ is singular.

We denote by A the $n \times m$ matrix characterizing the singularity. Its range is the eigen-subspace associated with the null eigenvalues of the second derivatives matrix $\nabla^2 f(x^*)$. Formally, let's assume that $\nabla^2 f(x^*)$ has $m < n$ null eigenvalues, so that its Schur decomposition is

$$\nabla^2 f(x^*) = (A\ B) \begin{pmatrix} 0 & 0 \\ 0 & \Lambda_2 \end{pmatrix} \begin{pmatrix} A^T \\ B^T \end{pmatrix} = B\Lambda_2 B^T$$

where $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{n\times(n-m)}$ are orthonormal, and the columns of A are the eigenvectors corresponding to 0 eigenvalues. In this case, any direction in the range of A does not modify (at least asymptotically) the value of f. Indeed, for an arbitrary $s \in \mathbb{R}^m$,

$$\begin{aligned} f(x^* + As) &= f(x^*) + \nabla f(x^*)^T As + \tfrac{1}{2} s^T A^T B \Lambda_2 B^T As + o(\|s\|^2) \\ &= f(x^*) + o(\|s\|^2), \end{aligned}$$

as $\nabla f(x^*) = 0$ and $A \perp B$. Invoking the fundamental theorem of linear algebra, we know that the subspace orthogonal to $\mathrm{Im}(A)$ is the null-space of A^T, that is $\mathrm{Im}(A)^\perp = \ker(A^T)$.

The main idea of our algorithm is to search primarily in $\ker(A^T)$. From a geometrical viewpoint, this is where the function exhibits non zero curvature. Actually, we would like the algorithm to generate directions s such that $A^T s = 0$.

The difficulty is that A is unknown before the optimization process starts, and needs to be approximated. We use $\nabla^2 f(x_k)$ as an approximation of $\nabla^2 f(x^*)$ as the algorithm proceeds. Consequently, performing an eigen-structure analysis of $\nabla^2 f(x_k)$ enables to generate the desired approximation of A. The eigen-subspace associated with eigenvalues of $\nabla^2 f(x_k)$ which are close to zero is used as an approximation for the range of A. The quality of such an approximation improves as x_k converges to x^*.

The computational burden of a full eigen-structure analysis per iteration is often unacceptable. For example, applying the full QR-algorithm for the symmetric eigenvalue problem to $\nabla^2 f(x_k)$ would require to compute a full QR-factorization of $\nabla^2 f(x_k)$ at each iteration of the identification procedure, that is $\mathcal{O}(n^3)$ flops.

Consequently, we propose a generalization of the inverse iteration method to identify the relevant subspace. The inverse iteration is an iterative process identifying the eigenvalue of a symmetric matrix $H \in \mathbb{R}^{n\times n}$ closest (in modulus) to a given target, as well as the associated eigenvector (see, for instance, Golub and Van Loan, 1996). The method proposed in this paper generalizes this procedure and allows to compute higher-dimensional invariant subspaces. Given a symmetric matrix $H \in \mathbb{R}^{n\times n}$, r such that $1 \leq r \leq n$, and a target λ, the generalized inverse iteration considers

$\bar{H} = (H - \lambda I_{n\times n})^{-1}$ and generates a matrix $\tilde{A} \in \mathbb{R}^{n\times r}$, such that $\mathrm{Im}(\tilde{A})$ approximates the dominant invariant subspace of dimension r of $\bar{H}$, which is the subspace associated with the r eigenvalues of H which are closest (in modulus) to the given target λ.

The main steps of the generalized inverse iteration at iteration k of the optimization algorithm can be summarized as follows.

- Consider an initial approximation $Q_k = Q_{\mathrm{init}} \in \mathbb{R}^{n\times r}$ of A. It can be either the r first columns of the identity matrix of dimension n, $I_{n\times n}$, or the approximation Q_{k-1} obtained by the procedure at the previous iteration of the optimization algorithm.

- Repeat:

  1. Compute $Z \in \mathbb{R}^{n\times r}$ by solving
  $$(H - \lambda I_{n\times n})Z = Q_k$$
  2. Compute the new approximation Q_k by performing a partial QR-factorization of Z, that is:
  $$Q_k R = Z$$

  until a given stopping criterion is satisfied.

Note that the partial QR-factorization is applied to a matrix belonging to $\mathbb{R}^{n\times r}$ so that Q_k is only composed of r columns. In comparison, a full QR-algorithm would compute at each iteration a full QR-factorization with $Q \in \mathbb{R}^{n\times n}$. The cost of this generalized inverse iteration is $\mathcal{O}(rn^2)$, which is interesting when r is small compared to n.

The stopping criterion is based on the difference in ℓ_2-norm between two consecutive Q_k approximations. As soon as this difference is below a given threshold (typically 10^{-6}), we stop the procedure. The last Q_k approximation represents the desired approximation of A. We obtained the eigenvalues associated with this eigen subspace by computing

$$\lambda_i = \frac{q_i^T H q_i}{q_i^T q_i}$$

for $i = 1, \ldots, r$ where q_i is the i-th column of Q_k.

In our case, we apply this method with $H = \nabla^2 f(x_k)$ and would like to identify the $\bar{r}$ eigenvectors corresponding to null eigenvalues. The dimension $\bar{r}$ is not known in advance. At each iteration k of the optimization algorithm, we use the following procedure to identify the dimension of the singularity $\bar{r}$. In order to reduce the computational cost (that is, the number of times we apply the generalized inverse iteration), we make use of the value found for $\bar{r}$ at the previous iteration, which we denote r_{previous}.

Initialization $r_{\text{previous}} = 0$ and singular $= 0$ for the first iteration of the optimization algorithm (for the subsequent iterations, these values are determined by this procedure).

Phase 1 `If` singular $= 0$

- Apply the generalized inverse iteration with $r = 1$.
- If the obtained eigenvector corresponds to a non-zero eigenvalue (that is, if the problem is not declared to be singular), $\bar{r} = 0$. Set singular $= 0$, $r_{\text{previous}} = 0$ and STOP.
- If the corresponding eigenvalue is declared to be zero (according to the threshold), set singular $= 1$, $r_{\text{previous}} = 1$ and go to Phase 2.

Phase 2

- Apply the generalized inverse iteration with $r = \max(r_{\text{previous}}, 1)$.
- If all corresponding eigenvalues are close to zero, go to Phase 3a. (We apply the generalized inverse iteration with increasing values of r).
- If at least one corresponding eigenvalue is declared to be nonzero:
  If $r = 1$, $\bar{r} = 0$. Set $r_{\text{previous}} = \bar{r}$ and STOP.
  Otherwise go to Phase 3b.
  (We apply the generalized inverse iteration with decreasing values of r).

Phase 3a `for` $r = \max(r_{\text{previous}}, 1) + 1 : n$

- Apply the generalized inverse iteration with r
- If the additional eigenvalue is close to zero, continue.
- If the additional eigenvalue is non-zero, $\bar{r} = r - 1$. Set $r_{\text{previous}} = \bar{r}$ and STOP.

Phase 3b `for` $r = r_{\text{previous}} - 1 : -1 : 1$

- Apply the generalized inverse iteration with r
- If it remains at least one non-zero eigenvalue, continue. If $r = 1$, $\bar{r} = 0$. Set $r_{\text{previous}} = 0$ and STOP.
- If all obtained eigenvalues are close to zero, $\bar{r} = r$. Set $r_{\text{previous}} = \bar{r}$ and STOP.

As r is usually small compared to n and does not change too much from iteration to iteration of the optimization algorithm, the cost of this procedure using the generalized inverse iteration is significantly lower than the one of a full QR-analysis. Moreover, this allows us to compute only relevant information for our purposes.

Note that the generalized inverse iteration fails with $\lambda = 0$ and we have to use a small positive value as target, such as $\lambda = 10^{-10}$. Also, we declare an eigenvalue to be null if its absolute value is less than 10^{-6}.

Now that the singularity is identified, we need to use this information to help the optimization algorithm. The central idea described in the next section is to constrain directions to lie in the subspace in which we have relevant information about curvature by using a penalty approach.

4 Trust-region based algorithms

In this paper, we focus on trust-region based methods. Indeed, these methods present significant theoretical and practical advantages, and can easily be adapted with many variants (see Conn et al., 2000). We start by presenting the classical trust-region framework for an optimization algorithm dedicated to solve unconstrained nonlinear optimization problems. An iteration k of a trust-region based algorithm can be summarized by the following steps:

Step 1: Model definition. Define a quadratic model m_k (typically using a truncated Taylor's series) of the objective function in a region $\mathcal{B}_k$ (called the *trust-region*) where this model can be trusted.

Step 2: Step computation. Compute a step s_k that sufficiently reduces the model m_k and such that $x_k + s_k \in \mathcal{B}_k$. This step is also called the trust-region subproblem because we approximately solve the following problem

$$\begin{cases} \min m_k(x_k + s) \\ \text{s.t. } x_k + s \in \mathcal{B}_k, \end{cases}$$

that is, minimizing the model within the trust-region.

Step 3: Acceptation of the trial point. Assess the quality of the trial step s_k and decide whether $x_k + s_k$ is accepted as the next iterate x_{k+1} or not.

Step 4: Trust-region radius update. Update the size of the trust-region.

Minimizing the quadratic model under the trust-region constraint is the core of the algorithm. Many methods have been proposed in the literature, such as "dogleg" or truncated conjugate-gradient (see Conn et al., 2000 for a review). In the latter case, preconditioning techniques have shown to improve the numerical behavior of the algorithm for difficult problems, such as the modified Cholesky factorization by Schnabel and Eskow (1999), available in the LANCELOT package (Conn et al., 1992).

The assessment of the model's quality is performed in general by comparing the improvement predicted by the model with the actual improvement of the objective function. Advanced techniques inspired from multi-criteria optimization have recently emerged, exhibiting faster convergence. Originally proposed by Fletcher and Leyffer (2002), these techniques are called "filter" methods.

Now we present different variants of this general scheme. Variants A and C are from the literature. Variants B and D are new ideas proposed in this paper.

4.1 Variant A: A trust-region algorithm

We first propose to use the basic trust-region algorithm, as described in Conn et al. (2000). In this variant, we consider the following specific steps:

Step 1a: Model definition. Define m_k in $\mathcal{B}_k$ (where $\mathcal{B}_k$ is a sphere centered at x_k of radius Δ_k) as a quadratic model of f around x_k, that is:

$$m_k(x_k + s) = f(x_k) + \nabla f(x_k)^T s + \frac{1}{2} s^T \nabla^2 f(x_k) s \tag{2}$$

Step 2a: Step computation. The original trust-region subproblem is defined as

$$\begin{cases} \min m_k(x_k + s) \\ \text{s.t. } \|s\| \leq \Delta_k, \end{cases} \tag{3}$$

where Δ_k is the radius of the trust-region.

Step 3a: Acceptation of the trial point. Compute $f(x_k + s_k)$ and define

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

If $\rho_k \geq \eta_1$, then define $x_{k+1} = x_k + s_k$; otherwise define $x_{k+1} = x_k$.

4.2 Variant B: A new trust-region algorithm

We propose a new trust-region algorithm to deal with singularity. It is an extension of Variant A where the trust-region subproblem is modified, involving the matrix Q_k defined in Section 3.

To achieve our objective of generating directions s such that $A^T s = 0$, we propose to penalize directions s such that $\|Q_k^T s\| > 0$, by modifying the model of the objective function as well as the trust-region subproblem. We consider the following specific steps:

Step 1b: Model definition. Define $\widehat{m}_k$ as follows:

$$\widehat{m}_k(x_k + s) = f(x_k) + \nabla f(x_k)^T s + \frac{1}{2} s^T \nabla^2 f(x_k) s + \frac{1}{2} c \|Q_k^T s\|^2 \tag{4}$$

Step 2b: Step computation. The corresponding trust-region subproblem is defined as

$$\begin{cases} \min \widehat{m}_k(x_k + s) = m_k(x_k + s) + \frac{1}{2} c \|Q_k^T s\|^2 \\ \text{s.t. } \|s\| \leq \Delta_k, \end{cases} \tag{5}$$

where $c \geq 0$ is the penalty parameter.

Step 3b: Acceptation of the trial point. Identical to Variant A.

We set $c = 0$ if $\nabla^2 f(x_k)$ is detected to be nonsingular. The second derivatives matrix of the new model is given by

$$\nabla^2 \widehat{m}_k(x_k) = \nabla^2 f(x_k) + c\, Q_k Q_k^T. \tag{6}$$

It means that we add a multiple of the $Q_k Q_k^T$ matrix to the second derivatives matrix of f when it is close to singularity. Geometrically, it amounts to "bending" the function in the subspace where there is originally no curvature. More precisely, eigenvalues of $\nabla^2 f(x_k)$ close to 0 take the value $c > 0$ in $\nabla^2 \widehat{m}_k(x_k)$.

The penalty parameter c is chosen as small as possible so that the perturbation of the model is not too severe. In practice, we start with $c = 1$, and test if the direction s^*, solution of (5), is such that $\|Q_k^T s^*\|$ is sufficiently close to zero (typically, $\|Q_k^T s^*\| \leq 10^{-3}$). If not, c is multiplied by 10 for the next iteration, until it reaches the upper bound κ_c (typically 10^5).

In addition to the obvious numerical reasons, this upper bound allows the new model to satisfy the general assumptions of the trust-region framework, in particular the fact that all eigenvalues of the second derivative matrix of the model must stay bounded. Consequently, convergence to a first-order critical point of the optimization problem can be guaranteed.

According to Conn et al. (2000), the trust-region based algorithm described above converges to first-order critical points if the following assumptions on the model are valid:

A.M.1 For all k, the model $\widehat{m}_k$ is twice differentiable.

A.M.2 The values of the objective function and of the model coincide at the current iterate; that is, for all k

$$\widehat{m}_k(x_k) = f(x_k).$$

A.M.3 The values of the gradient of the objective function and of the gradient of the model coincide at the current iterate; that is, for all k

$$\nabla \widehat{m}_k(x_k) = \nabla f(x_k).$$

A.M.4 The Hessian of the model remains bounded within the trust-region; that is,

$$\|\nabla^2 \widehat{m}_k(x_k)\| \leq \kappa_{umh} - 1 \text{ for all } x \in \mathcal{B}_k,$$

for all k, where $\kappa_{umh} \geq 1$ is a constant independent of k.

We briefly prove that the model $\widehat{m}_k$ satisfies these assumptions. To do this we first compute the first and second-order derivatives of $\widehat{m}_k$ which gives:

$$\nabla \widehat{m}_k(x_k + s) = \nabla f(x_k) + \nabla^2 f(x_k)^T s + c\, Q_k Q_k^T s, \tag{7}$$

and

$$\nabla^2 \widehat{m}_k(x_k + s) = \nabla^2 f(x_k) + c\, Q_k Q_k^T. \tag{8}$$

Using (7) and (8) and the assumption that the objective function f is twice differentiable, we directly obtain A.M.1. A.M.2 results from (5). Taking $s = 0$ in (7) gives immediately A.M.3. A.M.4 remains to be proved.

From (8), we have that:

$$\|\nabla^2 \widehat{m}_k(x_k)\| \leq \|\nabla^2 f(x_k)\| + c\|Q_k Q_k^T\| \leq \kappa_{ufh} + c \tag{9}$$

by using assumptions on f (namely the boundedness of the Hessian matrix) and the fact that columns of the matrix Q_k generated by the identification procedure have norm 1. We can conclude as we put an upper bound κ_c on the value of the penalty parameter c. Thus there exists a constant $\kappa_{umh} \geq 1$ such that

$$\|\nabla^2 \widehat{m}_k(x_k)\| \leq \kappa_{umh} - 1 \tag{10}$$

for all k. It is sufficient to take $\kappa_{umh} \geq \kappa_{ufh} + c + 1$. The constant being independent from k, we have the uniform boundedness.

4.3 Variant C: A standard filter algorithm

The concept of the filter has been introduced in nonlinear optimization by Fletcher and Leyffer (2002) and Fletcher et al. (2002). Inspired from multi-criteria optimization, it provides a great deal of flexibility to measure progress toward the solution of a problem, both in terms of optimality and feasibility. Fletcher and Leyffer (2002) define a 2-dimensional filter associated with the two objectives of constrained optimization, namely minimizing the objective function while satisfying the constraints. Gould et al. (2005) generalize the concept by using a multidimensional filter to solve systems of nonlinear equations as well as nonlinear least-squares. A multidimensional filter is also used in Gould et al. (2006) in the context of unconstrained optimization. The advantage of the filter is the increased flexibility in the optimization algorithm to accept new iterates, and consequently, a potentially faster convergence.

Our third algorithm is an adaptation of the algorithm proposed by Gould et al. (2006), with the following two modifications:

1. the flag RESTRICT is never set;
2. the test to accept the trial step (step 3) has been modified.

The first two steps of this variant are the same as Variant A, that is we used the classic model (2) and the original trust-region subproblem (3). The specific feature of this variant is the test for acceptance of the trial point $x_k^+ = x_k + s_k$.

Step 1c: Model definition. Identical to Variant A.

Step 2c: Step computation. Identical to Variant A.

Step 3c: Acceptation of the trial point.

- If x_k^+ is acceptable for the filter $\mathcal{F}$ and nonconvex[1] is unset
  Set $x_{k+1} = x_k^+$ and add g_k^+ to the filter $\mathcal{F}$ if $\rho_k < \eta_1$.
- If x_k^+ is not acceptable for the filter $\mathcal{F}$ or nonconvex is set
  If $\rho_k \geq \eta_1$ then
    Set $x_{k+1} = x_k^+$ and, if nonconvex is set, set $f_{sup} = f(x_{k+1})$ and reinitialize the filter $\mathcal{F}$ to the empty set;
  else Set $x_{k+1} = x_k$.

This filter variant accepts more often the trial point than the original trust-region algorithm. Indeed, if the trial point is acceptable for the filter, we move toward this point and if it is not, we look at the quality of the reduction factor ρ_k as in the first algorithm. Note that an iteration of this filter method is equivalent to a basic trust-region iteration when the function is nonconvex. The idea is to let the filter play the major role while convexity is encountered and falling back to the classical trust-region framework if non-convexity is detected.

4.4 Variant D: A new filter algorithm

We now consider a new filter algorithm to deal with singularity based on variant C exactly in the same way that we derived Variant B from Variant A. We consider the following specific steps:

Step 1d: Model definition. Define $\widehat{m}_k$ as follows:

$$\widehat{m}_k(x_k + s) = f(x_k) + \nabla f(x_k)^T s + \frac{1}{2} s^T \nabla^2 f(x_k) s + \frac{1}{2} c \|Q_k^T s\|^2 \tag{11}$$

Step 2d: Step computation. The corresponding trust-region subproblem is defined as

$$\begin{cases} \min \widehat{m}_k(x_k + s) \\ \text{s.t. } \|s\| \leq \Delta_k, \end{cases} \tag{12}$$

where $c \geq 0$ is the penalty parameter.

[1]see Gould et al., 2006 for details

Step 3d: Acceptation of the trial point. Identical to Variant C.

Following the convergence theory in Gould et al. (2006), the new model we propose in this filter variant must satisfy a major assumption in order to guarantee that the sequence of iterates produced by the filter algorithm converges to first-order critical points. More precisely, for all k, the model

$$\widehat{m}_k(x_k + s) = m_k(x_k + s) + \frac{1}{2}c\|Q_k^T s\|^2$$

has to be twice differentiable on $\mathbb{R}^n$ and must have a uniformly bounded Hessian.

Firstly, it is obvious to prove the twice differentiability (see (7) and (8)). Secondly, the uniform boundedness is obtained directly from (9) and (10) as this new filter algorithm makes use of the same model as Variant B corresponding to the new trust-region algorithm.

We also consider preconditioned versions of variants A and C, denoted A_p and C_p. As preconditioning matrix, we use a modified Cholesky factorization of the second derivatives matrix $\nabla^2 f(x_k)$. More precisely, the preconditioner is obtained following the lines of Schnabel and Eskow (1999).

To summarize, we consider a total of 6 algorithms, namely:

- The trust-region algorithm presented in Section 4.2 (Variant B) and the filter-trust-region algorithm presented in Section 4.4 (Variant D) both designed to handle singularity by the means of the perturbed trust-region subproblem (5) and the procedure described in Section 3.
- The basic trust-region algorithm (Variant A) and an adaptation of the standard filter-trust-region method (Variant C) using the classical model of the objective function (2).
- The preconditioned versions of Variant A and Variant C, A_p and C_p.

4.5 Implementation issues

- In practical tests, the trust-region subproblem consists in minimizing model (2) subject to the trust-region constraint, except that we approximate the second order derivatives matrix at the current iterate x_k, that is $\nabla^2 f(x_k)$, by a matrix H_k obtained using finite differences.

- The trust-region subproblem for the four first algorithmic variants is solved using a Truncated Conjugate Gradient method (see Toint, 1981, Steihaug, 1983 or Conn et al., 2000).
- For Variants A_p and C_p, we use a preconditioned conjugate gradient framework (see, for instance, Conn et al., 2000) instead of the standard conjugate gradient algorithm for solving the trust-region subproblem (3).

5 Numerical experiments

In this section, we present an analysis of the performances of new algorithmic variants compared to classical trust-region and filter algorithms from the literature. Section 5.1 contains a description of the set of test problems which have been used for the numerical experiments. The methodology for performance analysis is described Section 5.2. Sections 5.3-5.6 present results on singular problems while Section 5.7 shows the performance of proposed algorithms on non-singular problems.

5.1 Description of test problems

The set of test functions has been proposed by Moré et al. (1981). It is composed, among other things, of 34 unconstrained optimization problems. Most of these problems have a non-singular second derivatives matrix at the local minimum. As we want to perform tests on singular problems, we use the technique proposed by Schnabel and Frank (1984) to modify the problems of Moré et al. (1981) and create singular optimization problems such that the second derivatives matrix has a rank $n-k$ at the local solution where n is the dimension of the problem and $1 \leq k \leq n$ is the dimension of the singularity. In this paper we focus on problems having a second-order derivatives matrix of rank $n-1$ or $n-2$ at the local solution as in Schnabel and Chow (1991). Tests have been actually performed on 38 problems containing a singularity of dimension 1 (that is one null eigenvalue) at the local solution:

- 29 problems with dimension between 2 and 11,
- 3 problems with a dimension n which can be parametrized. In this case, we have used $n = 10, 20, 40$.

We also carried out tests on a set of 38 test functions whose second derivatives matrix has rank $n-2$ at x^*, namely:

- 29 problems with dimension between 3 and 11,
- 3 problems with a dimension n which can be parametrized. In this case, we have used $n = 10, 20, 40$.

For each problem, we have used the starting point given in the original paper of Moré et al. (1981).

Note that all tested algorithms have converged to the same solution for all 76 problems (when they did not fail to converge). Moreover, this solution corresponds to the local solution at which a given problem is singular.

To summarize, we thus have a set of 76 test problems in which the singularity has been explicitly incorporated.

5.2 Performance analysis

We present in the next sections a performance analysis of the variants presented in Section 4. All algorithms and test functions have been implemented with the package Octave (see `www.octave.org` or Eaton, 1997) and computations have been done on a desktop equipped with 3GHz CPU, in double precision.

The stopping criterion for all algorithms is a composition of two conditions: gradient close to zero, that is $\|\nabla f(x_k)\| \leq 10^{-6}$, and maximum number of iterations fixed to 1000. The measure of performance is the number of iterations or the CPU time necessary to reach convergence (as defined above). We are presenting the results following the performance profiles analysis method proposed by Dolan and Moré (2002). If $f_{p,a}$ is the performance index (the number of function evaluations, or the CPU time) of algorithm a on problem p, then the *performance ratio* is defined by

$$r_{p,a} = \frac{f_{p,a}}{\min_b\{f_{p,b}\}}, \tag{13}$$

if algorithm a has converged for problem p, and $r_{p,a} = r_{\text{fail}}$ otherwise, where r_{fail} must be strictly larger than any performance ratio (13) corresponding to a success. For any given threshold π, the overall performance of algorithm a is given by

$$\rho_a(\pi) = \frac{1}{n_p}\Phi_a(\pi) \tag{14}$$

where n_p is the number of problems considered, and $\Phi_a(\pi)$ is the number of problems for which $r_{p,a} \leq \pi$. In particular, the value $\rho_a(1)$ gives the proportion of times that algorithm a wins over all other algorithms. The value $\rho_a(\pi)$ with $\pi \geq r_{\text{fail}}$ gives the proportion of times that algorithm a solves a problem and, consequently, provides a measure of the robustness of each method.

Note that the sum of $\rho_a(1)$ values for all algorithms a considered in a given profile may exceed 1 in the case where some algorithms performs exactly the same on some of the tested problems.

5.3 TR and filter methods

We first compare variants A to D. Figure 1 represents the full profile while Figure 2 provides a zoom on π between 1 and 3. In terms of number of iterations, we can see that the two best algorithms are the new variants B and D. These new algorithms significantly outperform the classical ones both in efficiency and robustness. Note also that the new filter algorithm (Variant D) outperforms the new trust-region (Variant B) algorithm. Similarly, the standard filter method (Variant C) shows a better efficiency than the basic trust-region method (Variant A), consistently with the findings of Gould et al. (2006). Note also that filter variants are more robust than trust-region variants as they are able to solve all 76 problems on which algorithms have been tested.

Figures 3 and 4 show the performance of the four same variants in terms of CPU time. From Figure 4, we can already see that there is a computational overhead associated with the new variants proposed in this paper. It is mainly due to the computational cost of the identification procedure described in Section 3. It is easy to measure this overhead on

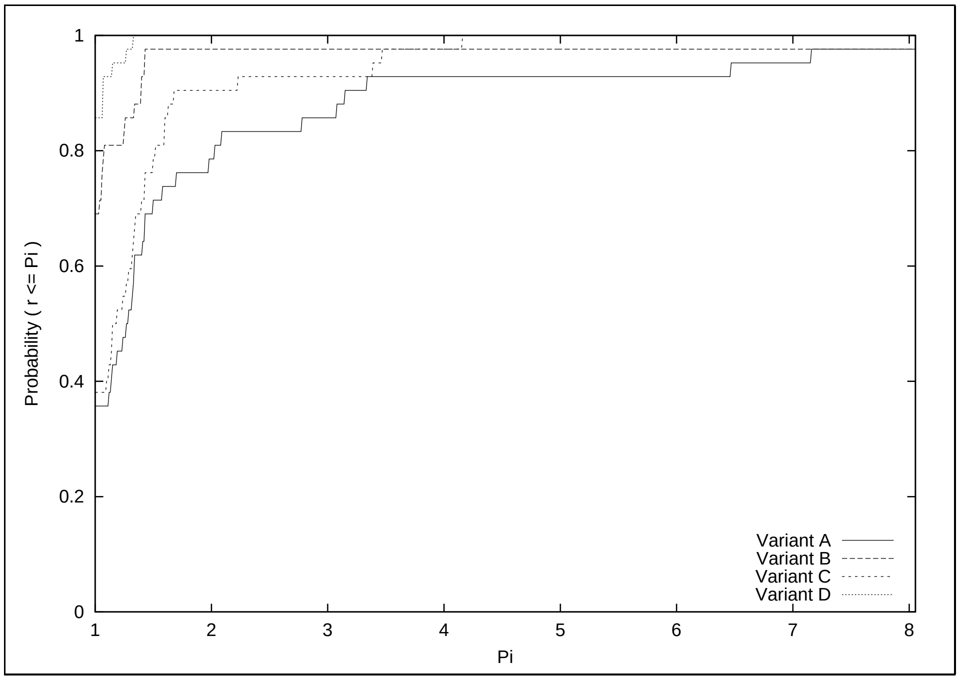


Figure 1: Comparison of the number of iterations for Variants A,B,C,D

specific profiles for trust-region and filter variants presented in the next subsections. We can also easily see that filter methods compensate the numerical algebra associated with the management of the filter by a better efficiency compared to trust-region algorithms on which they are based.

5.4 TR methods

We now compare variants A and B in Figure 5(a). Figure 5(b) provides a zoom on π between 1 and 3. The performance criterion is the number of iterations to reach convergence. Variant B performs significantly better than the classical algorithm in terms of both efficiency and robustness. From Figure 5(b), we see that it is the best on 90% of the 76 singular problems tested. When it is not the best algorithm, it converges within a factor around 1.25 of the classical trust-region algorithm on all 76 tested problems.

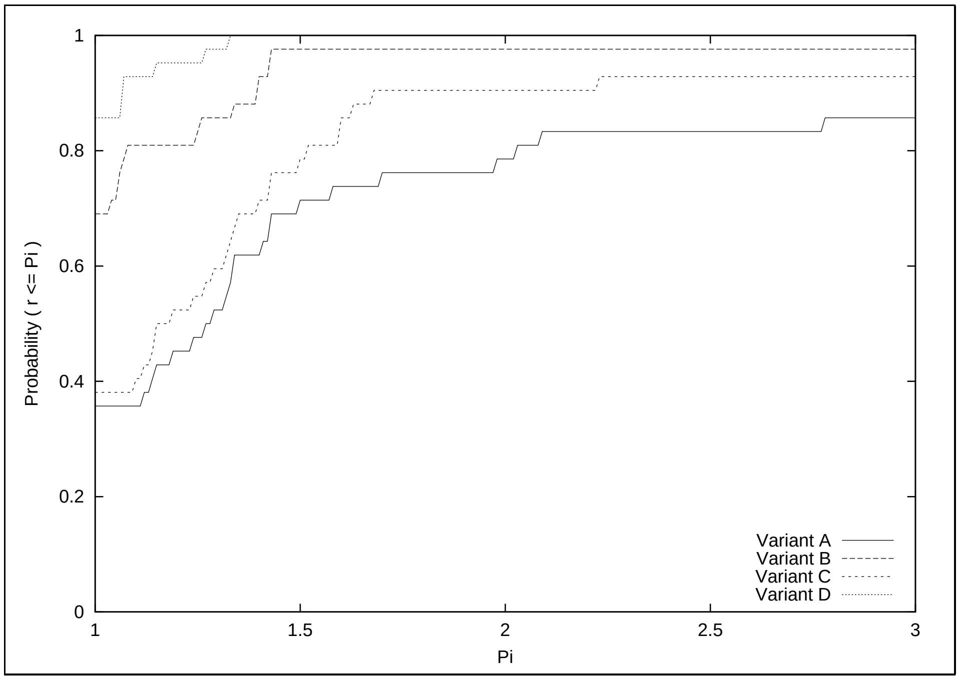


Figure 2: Zoom on the number of iterations for Variants A,B,C,D

In Figures 6(a) and 6(b), we compare variants A and B with regard to the CPU time. Our variant B is still the best method with regard to this measure of performance, even if we can see from these profiles that there is a computational overhead. As we already mentioned, it is mainly due to the numerical algebra of the identification procedure, that is the procedure described in Section 3 based on the generalized inverse iteration. Indeed the difference between the profiles associated with the competitors is smaller than previously. However, it is important to note that, even if the test problems do not have an objective function heavy to compute, the higher efficiency of the new variant compensates its computional overhead. Despite the additional effort in computation due to the singularity identification process, we see that the new algorithm takes, on more than 60% of the problems, less time to reach convergence thanks to the smaller number of iterations necessary to converge to a local solution. On some problems, the new algorithm is up to 5 times faster than the standard one in term of

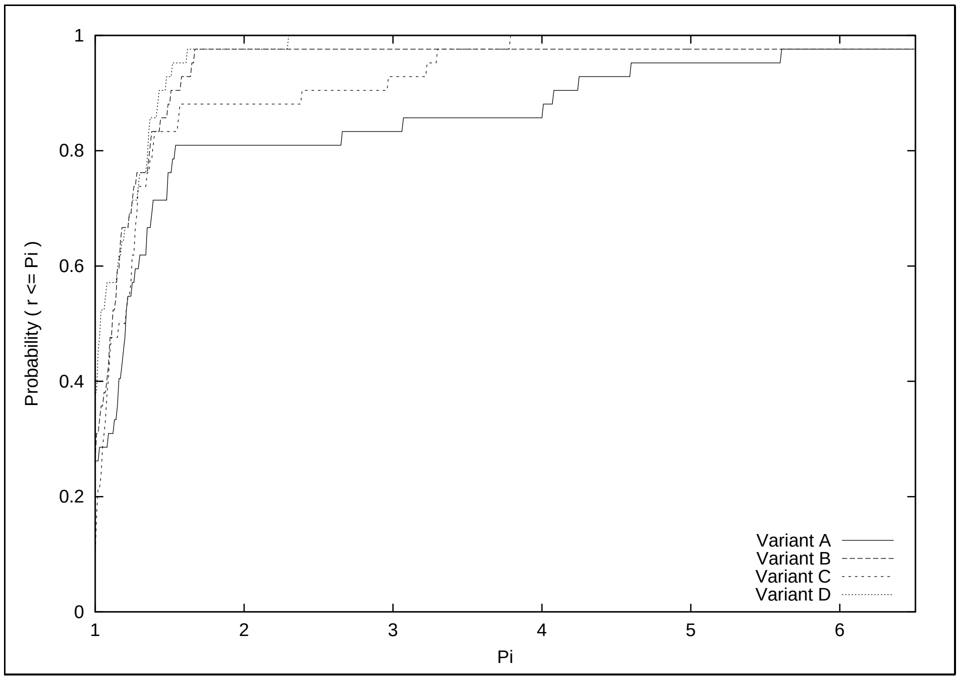


Figure 3: Comparison of the CPU time for Variants A,B,C,D

computational time. It is an indication that the new method is particularly appropriate when the function is computationally expensive to compute.

5.5 Filter methods

Here we compare the standard filter method (Variant C) with the variant proposed in Section 4.4 (Variant D). Figure 7(a) represents the full profile while Figure 7(b) provides a zoom on π between 1 and 3. The proposed variant significantly outperforms the adaptation of the filter algorithm proposed by Gould et al. (2006) in terms of number of iterations necessary to reach the convergence criterion. The new filter algorithm is the most efficient on almost all 76 tested problems. When it is not the best algorithm, it converges within a factor close to 1 of the standard filter algorithm. Note that methods are similar in terms of robustness.

Figures 8(a) and 8(b) show the performance of variants C and D in term

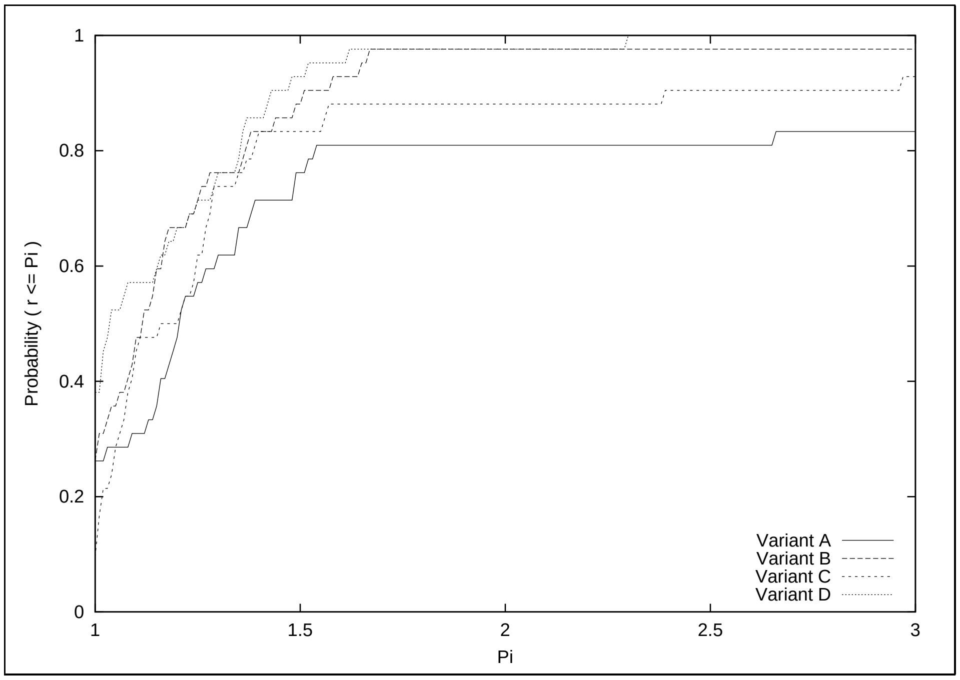
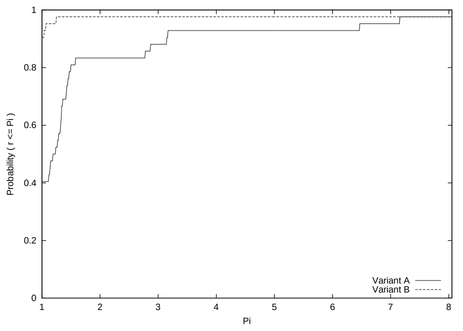


Figure 4: Zoom on the CPU time for Variants A,B,C,D

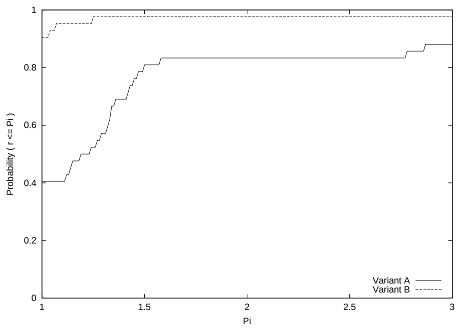
of CPU time. As it was the case when analyzing the performances of the new trust-region algorithm, we can easily observe that the computational overhead associated with the proposed filter method is compensated by its better efficiency. Our variant is the fastest algorithm in CPU time on nearly 65% of the tested problems. On some of the problems, it is up to 4 times faster to reach a solution. Again, we expect the advantage in CPU time to be larger for expensive functions.

5.6 Preconditioned versions vs. our variants

Here we compare preconditioned versions of trust-region (Variant $\mathrm{A_p}$) and filter (Variant $\mathrm{C_p}$) algorithms with variants B and D. We want to check if well-known preconditioning techniques would be a simple way of efficiently dealing with singularity issues in unconstrained optimization problems. Indeed, these techniques have shown their advantages when solving problems



(a) Full profile



(b) Zoom

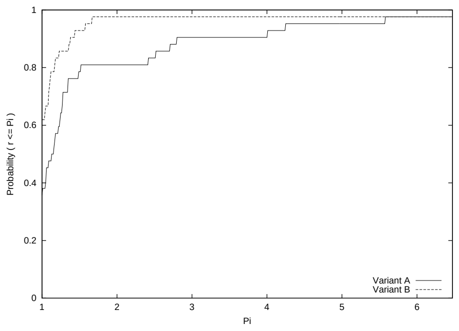
Figure 5: Comparison of the number of iterations for Variants A and B

presenting numerical difficulties. However, we clearly see from Figures 9 and 10 that our variants perform significantly better than preconditioned versions of A and C.

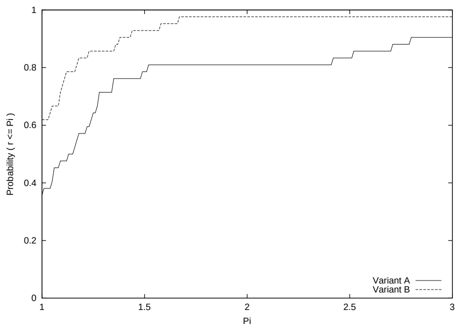
These preconditioning techniques are not designed to deal with the type of problem we consider in the scope of this paper. Indeed, the difficulty is due to the very small eigenvalues in the Hessian matrix of the objective function f. This specificity is taken into account by defining a new model of the objective function in (5) when a singularity is identified. As the second derivatives matrix of this model is given by (8), this procedure can be viewed as shifting very small eigenvalues of the Hessian matrix at the current iterate to moderate values whose magnitude is controlled by the penalty parameter c. It means that the technique we use in the proposed variants of trust-region and filter methods is acting exactly on the eigenvalues causing numerical difficulty.

5.7 Test on non-singular problems

We now present some tests on non-singular optimization problems. The idea is to analyze the computational overhead associated with the procedure described in Section 2 but also to see how our algorithmic variants behave on classical unconstrained optimization problems which do not ex-



(a) Full profile



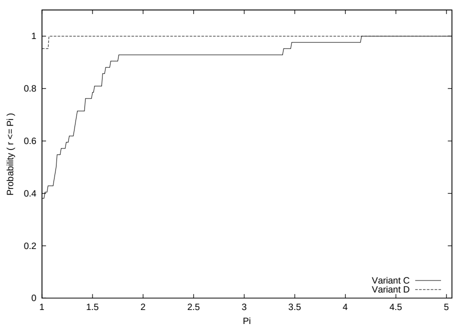
(b) Zoom

Figure 6: Comparison of the CPU time for Variants A and B

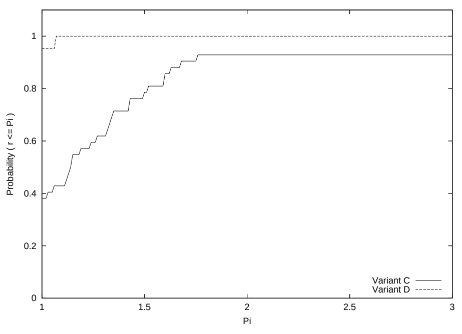
hibit singularity issues. The tests presented below have been achieved on 32 problems among the set of test functions proposed by Moré et al. (1981) which have been themselves selected from the CUTEr collection (see Gould et al., 2002).

We first compare the basic trust-region algorithm (Variant A) with the corresponding variant proposed in this paper (Variant B). Figure 13 represents the profile in terms of number of iterations while Figure 14 provides the profile in terms of CPU time. From Figure 13, we can see that performances of algorithms are similar on standard problems with a slight deterioration for the new algorithm. This is not surprising in the sense that our variant basically falls back to the classical trust-region framework if no singularity has been identified during the course of the algorithm. When looking at Figure 14, it clearly shows the computational cost of additional numerical algebra of our variant. Indeed, profiles are closer to each other compared to the profiles of Figure 6(a) obtained on singular problems. Moreover, the classical trust-region algorithm is faster in terms of CPU time on more than 60% of the tested problems as expected.

Figures 15 and 16 present the performance profiles associated with both filter variants (Variants C and D) on the same 32 non-singular problems. Similarly to trust-regions algorithms tested above, filter methods exhibit the same performance in terms of efficiency and robustness, as showed by



(a) Full profile



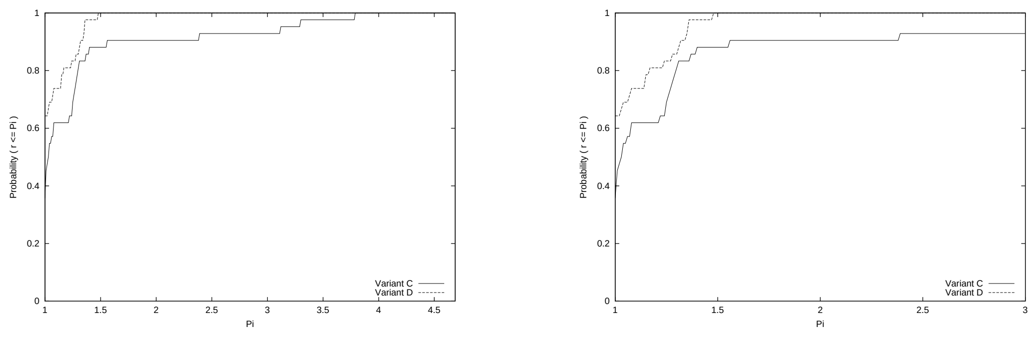
(b) Zoom

Figure 7: Comparison of the number of iterations for Variants C and D

Figure 15. Again, we can see the impact of the overhead associated with the new filter when solving non-singular problems if we compare Figures 16 and 8(a).

6 Conclusion

The paper deals with an important and difficult problem: dealing with singular problems in nonlinear optimization. It is important because it arises often in practice, especially in early stages of a modeling process, when the models to be optimized are not completely well defined. It is difficult because the efficiency of existing algorithms is characterized by the curvature of the objective function, which is 0 (or numerically close to it) for singular problems. We have proposed a simple technique to deal with singularities. It consists in artificially adding curvature, to allow existing methods to perform decently. This requires the identification of the subspace where the function is singular, which is achieved by the generalization of a classical technique in numerical linear algebra, that is the inverse iteration method. We have shown the superiority of our approach on a large set of problems. Namely, it appears that the computational overhead of the generalized inverse iteration method is compensated by the significant decrease in the number of iterations. This makes the method particularly appealing for



(a) Full profile (b) Zoom

Figure 8: Comparison of the CPU time for Variants C and D

problems where the CPU time spent in function evaluations is important, such as those involving simulation.

No specific theoretical analysis of the convergence of the method has been performed. We have shown that the method is consistent with the general framework of trust-region methods, and inherit its convergence properties. A specific analysis of the speed of convergence is left for future work. Also, it would be natural to generalize the proposed approach to constrained problems.

References

Bertsekas, D. P. (1999). *Nonlinear Programming*, 2nd edn, Athena Scientific, Belmont.

Bierlaire, M. (2006). *Introduction à l'optimisation différentiable*, Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland. In press.

Conn, A. R., Gould, N. I. M. and Toint, P. (2000). *Trust region methods*, MPS–SIAM Series on Optimization, SIAM.

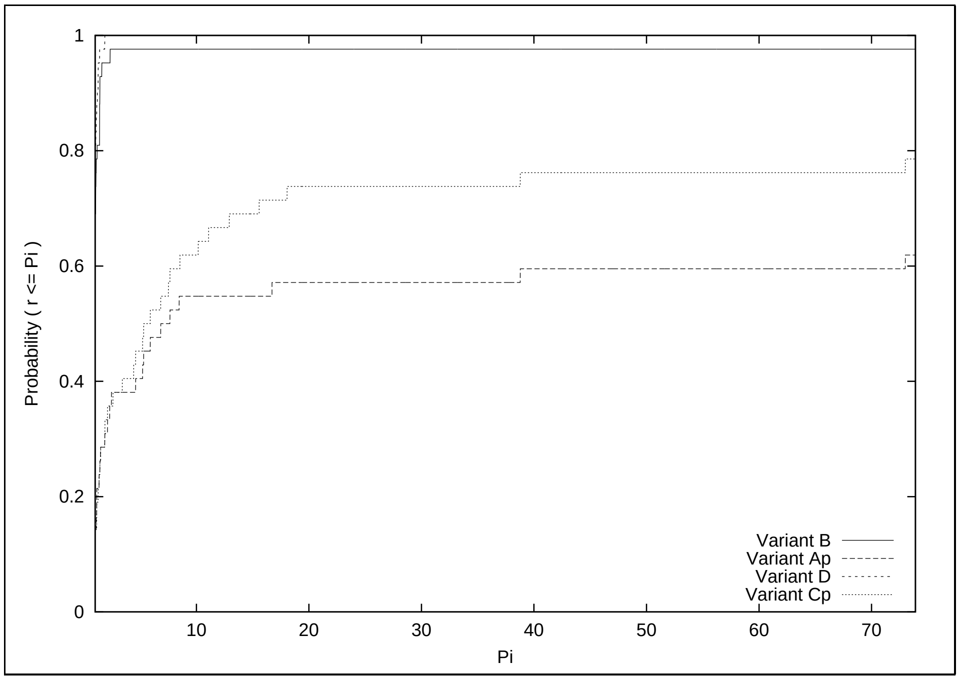


Figure 9: Comparison of the number of iterations for Variants B,A_{p},D,$\mathrm{C_p}$

Conn, A. R., Gould, N. I. M. and Toint, P. L. (1992). 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.

Decker, D. W. and Kelley, C. T. (1980). Newton's method at singular points ii., *SIAM Journal on Numerical Analysis* **17**(3): 465–471.

Decker, D. W., Kelley, H. B. and Kelley, C. T. (1983). Convergence rates for newton's method at singular points, *SIAM Journal on Numerical Analysis* **20**(2): 296–314.

Dennis, J. E. and Schnabel, R. B. (1983). *Numerical methods for unconstrained optimization and nonlinear equations*, Prentice-Hall, Englewood Cliffs, USA.

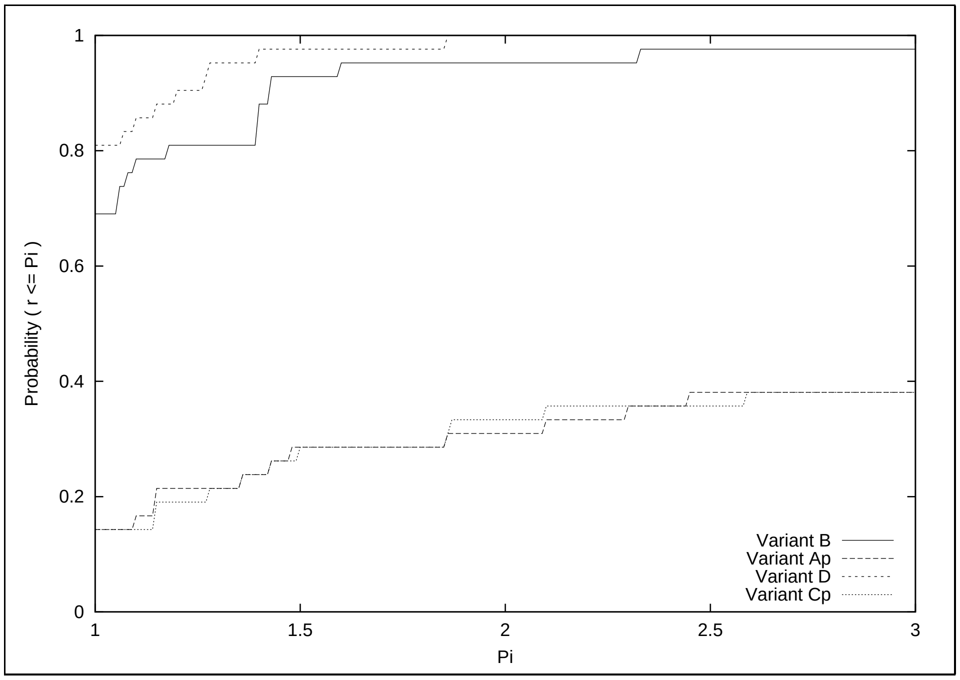


Figure 10: Zoom on the number of iterations for Variants B,A_p,D,C_p

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

Eaton, J. W. (1997). GNU *Octave Manual*, Series on Optimization, Network Theory Limited, Bristol, United Kingdom.

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

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

Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations*, third edn, Johns Hopkins University Press, Baltimore and London.

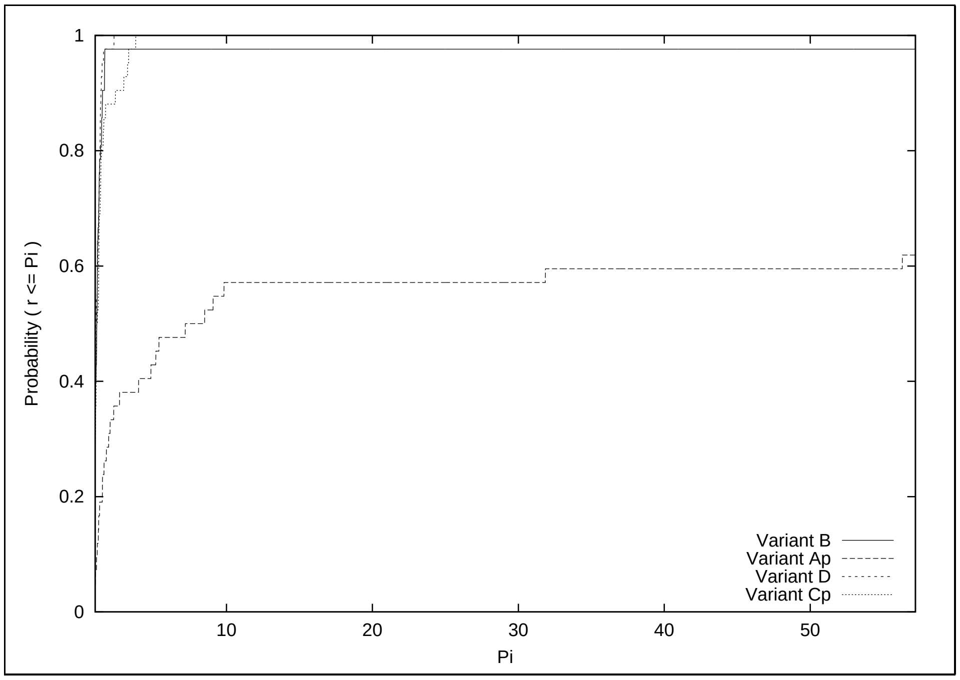


Figure 11: Comparison of the CPU time for Variants B,A_p,D,C_p

Gould, N. I. M., Leyffer, S. and Toint, P. L. (2005). A multidimensional filter algorithm for nonlinear equations and nonlineat least-squares, *SIAM Journal on Optimization* **15**(1): 17–38.

Gould, N. I. M., Orban, D. and Toint., P. L. (2002). General CUTEr documentation, *Technical Report TR/PA/02/13*, CERFACS.

Gould, N. I. M., Sainvitu, C. and Toint, P. L. (2006). A filter-trust-region method for unconstrained optimization, *SIAM Journal on Optimization* **16**(2): 341–357.

Griewank, A. O. (1985). On solving nonlinear equations with simple singularities or nearly singular solutions, *SIAM Review* **27**(4): 537–563.

Griewank, A. O. and Osborne, M. R. (1983). Analysis of newton's method at irregular singularities, *SIAM Journal on Numerical Analysis* **20**(4): 747–773.

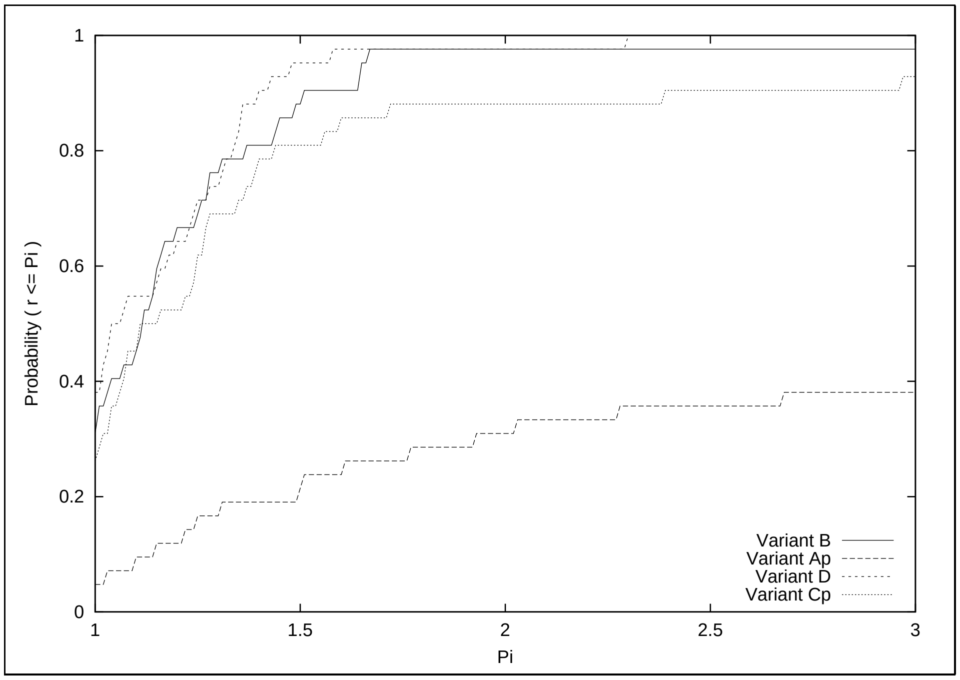


Figure 12: Zoom on the CPU time for Variants B,A_p,D,C_p

Izmailov, A. and Solodov, M. (2002). Superlinearly convergent algorithms for solving singular equations and smooth reformulations of complementarity problems, *SIAM Journal on Optimization* **13**: 386–405.

Moré, J. J., Garbow, B. S. and Hillstrom, K. E. (1981). Testing unconstrained optimization software, *ACM Transactions on Mathematical Software* **7**(1): 17–41.

Nocedal, J. and Wright, S. J. (1999). *Numerical optimization*, Operations Research, Springer Verlag, New-York.

Schnabel, R. B. and Eskow, E. (1999). A revised modified Cholesky factorization, *SIAM Journal on Optimization* **9**: 1135–1148.

Schnabel, R. B. and Frank, P. D. (1984). Tensor methods for nonlinear equations, *SIAM Journal on Numerical Analysis* **21**(5): 815–843.

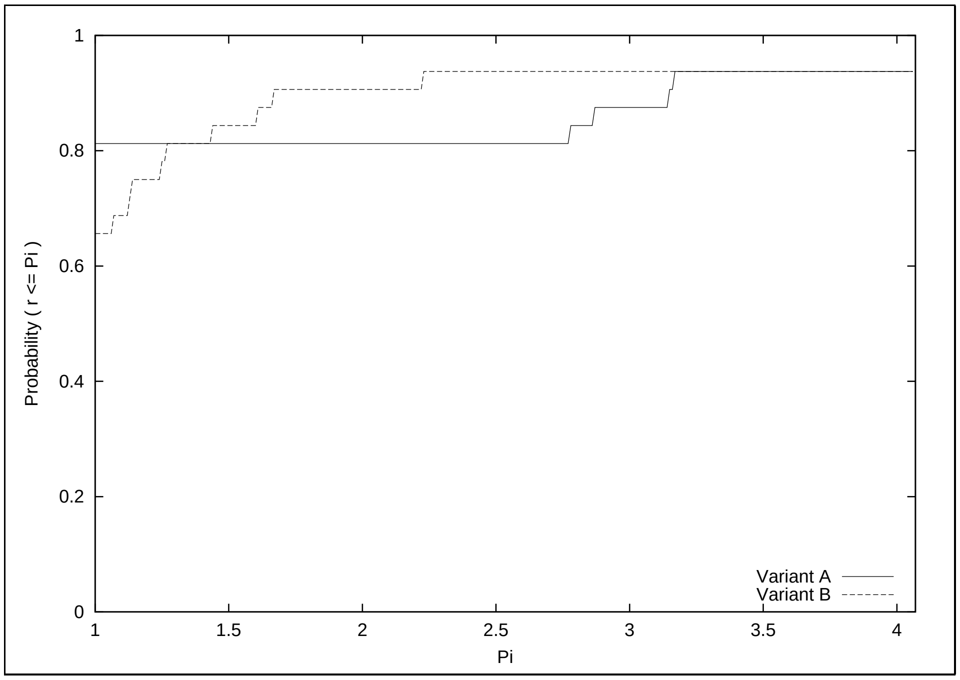


Figure 13: Comparison of the number of iterations for Variants A and B on non-singular problems

Schnabel, R. and Chow, T.-T. (1991). Tensor methods for unconstrained optimization using second derivatives, *SIAM Journal on Optimization* **1**(3): 293–315.

Steihaug, T. (1983). The conjugate gradient method and trust regions in large scale optimization, *SIAM Journal on Numerical Analysis* **20**(3): 626–637.

Thémans, M. and Bierlaire, M. (2006). Solving singularity issues in the estimation of econometric models, *6th STRC Swiss Transport Research Conference*, Monte Verita, Ascona, Switzerland. www.strc.ch.

Toint, P. L. (1981). Towards an efficient sparsity exploiting Newton method for minimization, *in* I. S. Duff (ed.), *Sparse Matrices and Their Uses*, Academic Press, London, pp. 57–88.

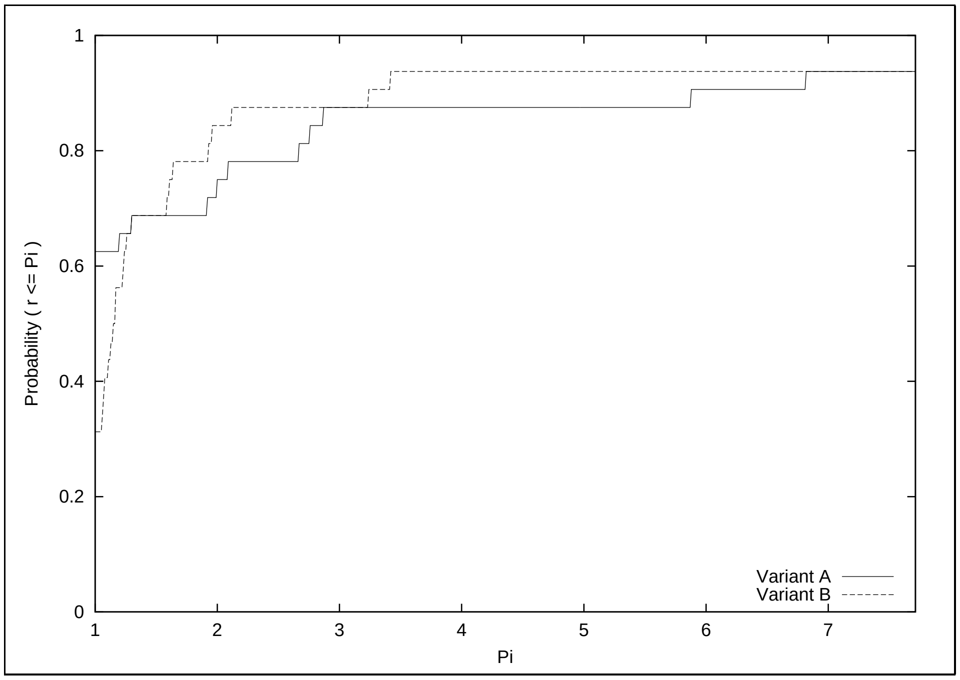


Figure 14: Comparison of the CPU time for Variants A and B on non-singular problems

Walker, J. L. (2001). *Extended discrete choice models: integrated framework, flexible error structures, and latent variables*, PhD thesis, Massachusetts Institute of Technology.

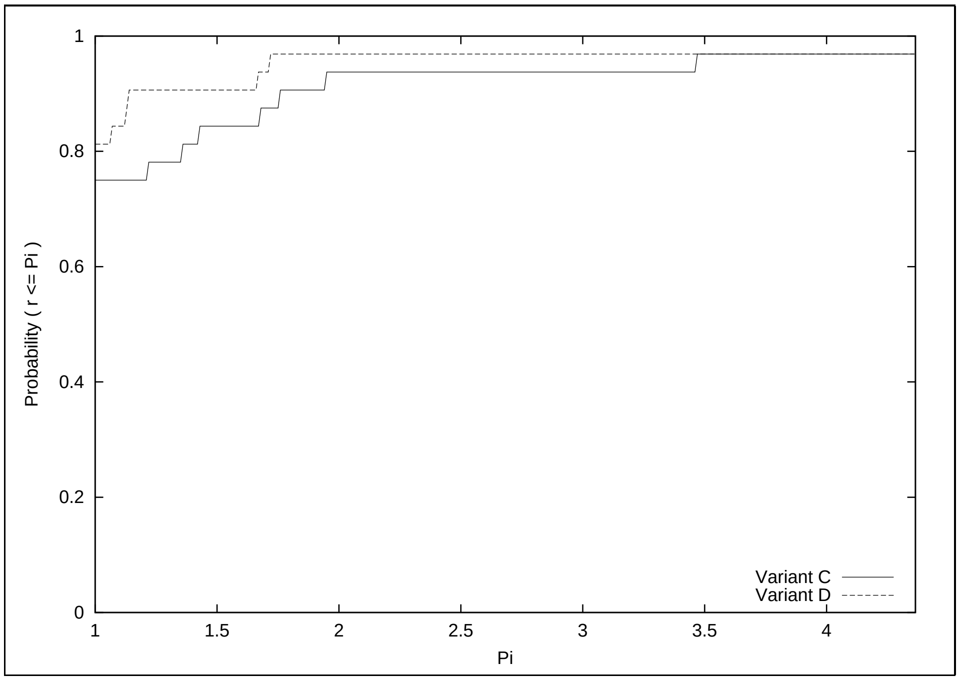


Figure 15: Comparison of the number of iterations for Variants C and D on non-singular problems

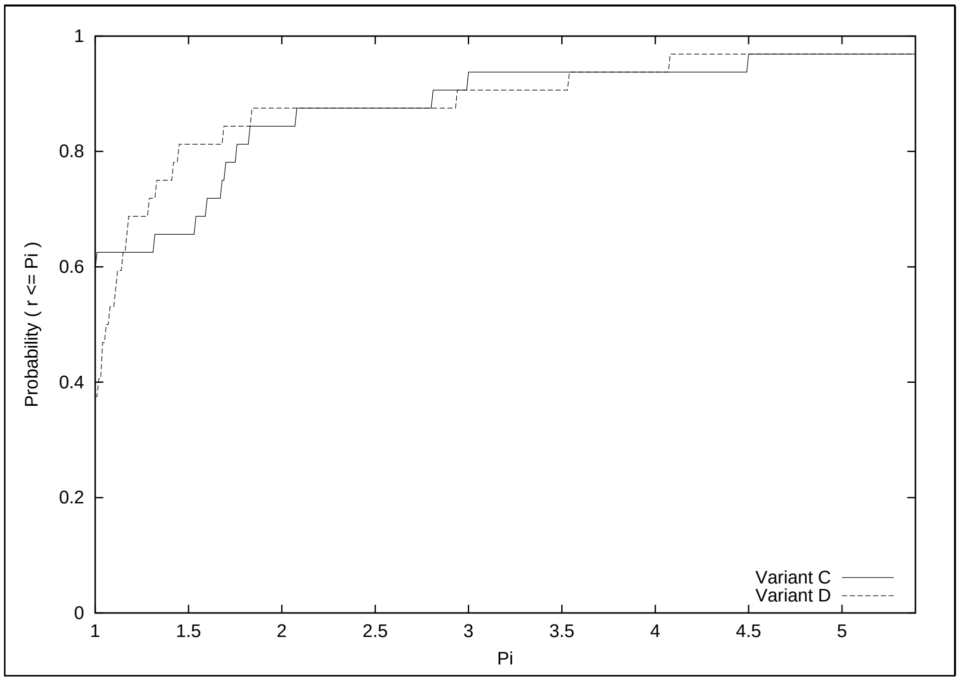


Figure 16: Comparison of the CPU time for Variants C and D on non-singular problems