AUTOMATIC DETERMINATION OF AN INITIAL TRUST REGION IN NONLINEAR PROGRAMMING

A. SARTENAER*

Abstract. This paper presents a simple but efficient way to find a good initial trust region radius in trust region methods for nonlinear optimization. The method consists of monitoring the agreement between the model and the objective function along the steepest descent direction, computed at the starting point. Further improvements for the starting point are also derived from the information gleaned during the initializing phase. Numerical results on a large set of problems show the impact the initial trust region radius may have on trust region methods behaviour and the usefulness of the proposed strategy.

Key Words. Nonlinear optimization, trust region methods, initial trust region, numerical results

1. Introduction. Trust region methods for unconstrained optimization were first introduced by Powell in [14]. Since then, these methods have enjoyed a good reputation on the basis of their remarkable numerical reliability in conjunction with a sound and complete convergence theory. They have been intensively studied and applied to unconstrained problems (see for instance [11], [14], and [15]), and also to problems including bound constraints (see [4], [7], [12]), convex constraints (see [2], [6], [18]), and non-convex ones (see [3], [5], and [19], for instance).

At each iteration of a trust region method, the nonlinear objective function is replaced by a simple model centered on the current iterate. This model is built using first and possibly second order information available *at* this iterate and is therefore usually suitable only in a certain limited region surrounding this point. A *trust region* is thus defined where the model is supposed to agree adequately with the true objective function. Trust region approaches then consist of solving a sequence of subproblems in which the model is approximately minimized within the trust region, yielding a candidate for the next iterate. When a candidate is determined that guarantees a sufficient decrease on the model inside the trust region, the objective function is then evaluated at this candidate. If the objective value has decreased sufficiently, the candidate is accepted as next iterate and the trust region is possibly enlarged. Otherwise, the new point is rejected and the trust region is reduced. The updating of the trust region is directly dependant on a certain measure of agreement between the model and the objective function.

A good choice for the *trust region radius* as the algorithm proceeds is crucial. Indeed, if the trust region is too large compared with the agreement between the model and the objective function, the approximate minimizer of the model is likely to be a poor indicator of an improved iterate for the true objective function. On the other hand, too small a trust region may lead to very slow improvement in the estimate of the solution.

When implementing a trust region method, the question then arises of an appropriate choice for the *initial trust region radius* (ITRR). This should clearly reflect the region around the starting point, in which the model and objective function approximately agree. However, all the algorithms the author is aware of use a rather ad hoc value for this ITRR. In many algorithms, users are expected to provide their

* Department of Mathematics, Facultés Universitaires N. D. de la Paix, 61 rue de Bruxelles, B-5000, Namur, Belgium (**as@math.fundp.ac.be**). This work was supported by the Belgian National Fund for Scientific Research.

own choice based on their knowledge of the problem (see [8], and [9]). In other cases, the algorithm initializes the trust region radius to the distance to the Cauchy point (see [13]), or to a multiple or a fraction of the gradient norm at the starting point (see [8], and [9]). In each of these cases, the ITRR may not be adequate, and, even if the updating strategies used thereafter generally allow to recover in practice from a bad initial choice, there is usually some undesired additional cost in the number of iterations performed. Therefore, the ITRR selection may be considered as important, especially when the linear-algebra required per iteration is costly.

In this paper, we propose a simple but efficient way of determining the ITRR, which consists of monitoring the agreement between the model and the objective function along the steepest descent direction computed at the starting point. Further improvements for the starting point will also be derived from the information gleaned during this initializing phase. Numerical experiments, using a modified version of the nonlinear optimization package LANCELOT (see [8]), on a set of relatively large test examples from the CUTE test suite (see [1]), show the merits of the proposed strategy.

Section 2 of the paper develops the proposed automatic determination of a suitable ITRR. The detailed algorithm is described in §3. Computational results are presented and discussed in §4. We finally conclude in §5.

2. Automatic determination of an initial trust region.

2.1. Classical trust region update. We consider the solution of the unconstrained minimization problem

$$\text{minimize}_{x \in \mathbf{R}^n} \; f(x). \tag{2.1}$$

The function f is assumed to be twice-continuously differentiable and a trust region method is used, whose iterations are indexed by k, to solve this problem.

At iteration k, the quadratic model of $f(x)$ around the current iterate $x^{(k)}$ is denoted by

$$m^{(k)}(x^{(k)} + s) = f(x^{(k)}) + g(x^{(k)})^T s + \frac{1}{2} s^T B^{(k)} s, \tag{2.2}$$

where $g(x^{(k)})$ denotes $\nabla f(x^{(k)})$ and $B^{(k)}$ is a symmetric approximation of the Hessian matrix $\nabla^2 f(x^{(k)})$. (Subsequently, we will use the notation $f^{(k)}$ and $g^{(k)}$ for $f(x^{(k)})$ and $g(x^{(k)})$, respectively.) The *trust region* is defined as the region where

$$\|s\| \leq \Delta^{(k)}. \tag{2.3}$$

Here $\Delta^{(k)}$ denotes the *trust region radius* and $\|\cdot\|$ is a given norm.

When a candidate for the next iterate, $x^{(k)} + s^{(k)}$, say, is computed that approximately minimizes (2.2) subject to the constraint (2.3), a classical framework for the trust region radius update is to set

$$\Delta^{(k+1)} = \beta^{(k)} \Delta^{(k)}, \tag{2.4}$$

for some selected $\beta^{(k)}$ satisfying

$$\beta^{(k)} \in \begin{cases} [\gamma_1, 1) & \text{if } \rho^{(k)} < \eta_1, \\ [1, \gamma_2] & \text{if } \rho^{(k)} \geq \eta_2, \\ [\gamma_3, \gamma_4] & \text{otherwise,} \end{cases} \tag{2.5}$$

where $0 < \gamma_1 < 1 < \gamma_2$, $\gamma_1 \leq \gamma_3 < 1 < \gamma_4 \leq \gamma_2$, and $0 < \eta_1 < \eta_2 < 1$. In (2.5), the quantity

$$\rho^{(k)} \stackrel{\text{def}}{=} \frac{f^{(k)} - f(x^{(k)} + s^{(k)})}{f^{(k)} - m^{(k)}(x^{(k)} + s^{(k)})} \tag{2.6}$$

represents the ratio of the achieved to the predicted reduction of the objective function. The reader is referred to [8], [9], and [10] for instances of trust region updates using (2.4)–(2.5).

2.2. Initial trust region determination. The problem in determining an ITRR $\Delta^{(0)}$ is to find a cheap way to test agreement between the model (2.2) and the objective function at the starting point, $x^{(0)}$. The method presented here is based on the use of information generally available at this point, namely the function value and the gradient. With the extra cost of some function evaluations, a reliable ITRR will be determined, whose use will hopefully reduce the number of iterations required to find the solution. As shown in §4, the possible saving produced in most cases largely warrants the extra cost needed to improve the ITRR.

The basic idea is to determine a *maximal radius* that guarantees a *sufficient agreement* between the model and the objective function *in the direction* $-g^{(0)}$, using an iterative search along this direction. At each iteration i of the search, given a radius estimate $\Delta_i^{(0)}$, the model and the objective function values are computed at the point $x^{(0)} - \Delta_i^{(0)}\hat{g}^{(0)}$, where $\hat{g}^{(0)} \stackrel{\text{def}}{=} g^{(0)}/\|g^{(0)}\|$. Writing

$$m_i^{(0)} = m^{(0)}(x^{(0)} - \Delta_i^{(0)}\hat{g}^{(0)}) \quad \text{and} \quad f_i^{(0)} = f(x^{(0)} - \Delta_i^{(0)}\hat{g}^{(0)}), \tag{2.7}$$

the ratio

$$\rho_i^{(0)} = \frac{f^{(0)} - f_i^{(0)}}{f^{(0)} - m_i^{(0)}} \tag{2.8}$$

is also calculated, and the algorithm then stores the maximal value among the estimates $\Delta_\ell^{(0)}$ ($\ell = 0, \ldots, i$), whose associated $\rho_\ell^{(0)}$ is "close enough to one" (following some given criterion). It finally updates the current estimate $\Delta_i^{(0)}$.

The updating phase for $\Delta_i^{(0)}$ follows the framework presented in (2.4)–(2.5), but includes a more general test on $\rho_i^{(0)}$ because the *predicted change* in (2.8) (unlike that in (2.6)) is not guaranteed to be positive. That is, we set $\Delta_{i+1}^{(0)} = \beta_i^{(0)}\Delta_i^{(0)}$ where

$$\beta_i^{(0)} \in \begin{cases} [\gamma_1, 1) & \text{if } |\rho_i^{(0)} - 1| > \mu_1, \\ [1, \gamma_2] & \text{if } |\rho_i^{(0)} - 1| \leq \mu_2, \\ [\gamma_3, \gamma_4] & \text{otherwise,} \end{cases} \tag{2.9}$$

for some $0 \leq \mu_2 < \mu_1$. Note that update (2.9) only takes the adequacy between the objective function and its model into consideration, without taking care of the minimization of the objective function f. That is, it may happen that the radius estimate is decreased ($\beta_i^{(0)} \in [\gamma_1, 1)$), because $\rho_i^{(0)}$ is not close enough to one ($|\rho_i^{(0)} - 1| > \mu_1$), even though a big reduction is made in the objective function (if $f^{(0)} - f_i^{(0)} > (1 + \mu_1)(f^{(0)} - m_i^{(0)}) > 0$, for instance). On the other hand, the radius estimate could be augmented ($\beta_i^{(0)} \in [1, \gamma_2]$), because $\rho_i^{(0)}$ is close enough to one ($|\rho_i^{(0)} - 1| \leq \mu_2$),

when actually the objective function has increased (if $f^{(0)} - f_i^{(0)} = f^{(0)} - m_i^{(0)} < 0$, for instance). This is not contradictory, as far as we forget temporarily about the minimization of f and concentrate exclusively on the adequacy between the objective function and its model to find a good ITRR. In the next section, we shall consider an extra feature that will take account of a possible decrease in f during the process.

In order to select a suitable value for $\beta_i^{(0)}$ satisfying (2.9), a careful strategy detailed below is applied, which takes advantage of the current available information. This strategy uses quadratic interpolation (as already done in some existing framework for trust region updates, see [9]), and has been inspired by the trust region updating rules developed in [8].

The univariate function $f(x^{(0)} - \beta\Delta_i^{(0)}\hat{g}^{(0)})$ is first modeled by the quadratic $q_i^{(0)}(\beta)$ that fits $f^{(0)}$, $f_i^{(0)}$, and the directional derivative $-\Delta_i^{(0)}g^{(0)T}\hat{g}^{(0)}$:

$$q_i^{(0)}(\beta) = f^{(0)} - \beta g^{(0)T}d_i^{(0)} + \beta^2(f_i^{(0)} - f^{(0)} + g^{(0)T}d_i^{(0)}), \tag{2.10}$$

where $d_i^{(0)} = \Delta_i^{(0)}\hat{g}^{(0)}$. Assuming that this quadratic does not coincide with the univariate quadratic $m^{(0)}(x^{(0)} - \beta d_i^{(0)})$, it is used to provide candidates for $\beta_i^{(0)}$ for which the ratio $\rho_i^{(0)}$ *would be* close to one (slightly smaller and slightly larger than one, respectively) *if* f *were the quadratic* $q_i^{(0)}(\beta)$. That is, equations

$$\frac{f^{(0)} - q_i^{(0)}(\beta)}{f^{(0)} - m^{(0)}(x^{(0)} - \beta d_i^{(0)})} = 1 - \theta \quad \text{and} \quad \frac{f^{(0)} - q_i^{(0)}(\beta)}{f^{(0)} - m^{(0)}(x^{(0)} - \beta d_i^{(0)})} = 1 + \theta \tag{2.11}$$

are solved (where $\theta > 0$ is a small positive constant), yielding candidates

$$\beta_{i,1}^{(0)} = \frac{-\theta g^{(0)T}d_i^{(0)}}{\theta(f^{(0)} - g^{(0)T}d_i^{(0)}) + (1-\theta)m_i^{(0)} - f_i^{(0)}} \tag{2.12}$$

and

$$\beta_{i,2}^{(0)} = \frac{\theta g^{(0)T}d_i^{(0)}}{-\theta(f^{(0)} - g^{(0)T}d_i^{(0)}) + (1+\theta)m_i^{(0)} - f_i^{(0)}}, \tag{2.13}$$

respectively. These two candidates will be considered as possible choices for a suitable $\beta_i^{(0)}$ satisfying (2.9), provided a careful analysis based on two principles is first performed.

The first principle is to select and exploit, as much as possible, the *relevant* information that may be drawn from $\beta_{i,1}^{(0)}$ and/or $\beta_{i,2}^{(0)}$. For instance, if $\beta_{i,1}^{(0)}$ is greater than one and the radius estimate must be decreased (because $|\rho_i^{(0)} - 1| > \mu_1$ in (2.9)), it should be ignored. The second principle consists in favouring the *maximal* value for $\beta_i^{(0)}$ among the relevant ones. Based on the observation that the linear-algebraic costs during a trust region iteration are generally less when the trust region has been contracted (because part of the computation may be reused after a contraction but not after an expansion), this corresponds to favour an ITRR choice on the large rather than small side.

As in (2.9), we distinguish three mutually exclusive cases. The first case, for which $\beta_i^{(0)} \in [\gamma_1, 1)$ is sought, occurs when $|\rho_i^{(0)} - 1| > \mu_1$. Several possibilities are considered in this first case, that produce choice (2.14).

- Both $\beta_{i,1}^{(0)}$ and $\beta_{i,2}^{(0)}$ are irrelevant, that is, they recommend an increase of the radius estimate while in this case, in reality it should be decreased. These values are then ignored, and $\beta_i^{(0)}$ is set to a fixed constant $\gamma_3 \in [\gamma_1, 1)$.
- *All the* available relevant information provides a smaller value than the lower bound γ_1. Set $\beta_i^{(0)} = \gamma_1$.
- Either $\beta_{i,1}^{(0)}$ (or $\beta_{i,2}^{(0)}$) belongs to the appropriate interval, while $\beta_{i,2}^{(0)}$ (or $\beta_{i,1}^{(0)}$, respectively) is irrelevant or too small. The relevant one is selected.
- Both $\beta_{i,1}^{(0)}$ and $\beta_{i,2}^{(0)}$ are within the acceptable bounds. The maximum is then chosen.

These possibilities yield:

$$\beta_i^{(0)} = \begin{cases} \gamma_3 & \text{if } \min(\beta_{i,1}^{(0)}, \beta_{i,2}^{(0)}) > 1, \\ \gamma_1 & \text{if } \max(\beta_{i,1}^{(0)}, \beta_{i,2}^{(0)}) < \gamma_1, \quad \text{or,} \\ & \text{if } \min(\beta_{i,1}^{(0)}, \beta_{i,2}^{(0)}) < \gamma_1 \text{ and} \\ & \quad \max(\beta_{i,1}^{(0)}, \beta_{i,2}^{(0)}) \geq 1, \\ \beta_{i,1}^{(0)} & \text{if } \beta_{i,1}^{(0)} \in [\gamma_1, 1) \text{ and } \beta_{i,2}^{(0)} \notin [\gamma_1, 1), \\ \beta_{i,2}^{(0)} & \text{if } \beta_{i,2}^{(0)} \in [\gamma_1, 1) \text{ and } \beta_{i,1}^{(0)} \notin [\gamma_1, 1), \\ \max(\beta_{i,1}^{(0)}, \beta_{i,2}^{(0)}) & \text{otherwise.} \end{cases} \tag{2.14}$$

In the second case (i.e. when $|\rho_i^{(0)} - 1| \leq \mu_2$), the choice (2.15) is performed to select a suitable $\beta_i^{(0)} \in [1, \gamma_2]$, based on the following reasoning.

- Both $\beta_{i,1}^{(0)}$ and $\beta_{i,2}^{(0)}$ are irrelevant because they recommend a decrease of the radius estimate. $\beta_i^{(0)}$ is set to a fixed constant $\gamma_4 \in (1, \gamma_2]$.
- *At least one* available piece of relevant information provides a larger value than the upper bound γ_2. Since any maximal pertinent information is favoured, $\beta_i^{(0)}$ is set to this bound.
- Either $\beta_{i,1}^{(0)}$ or $\beta_{i,2}^{(0)}$ belongs to the appropriate interval, while the other is irrelevant. $\beta_i^{(0)}$ is set to the relevant one.
- Both $\beta_{i,1}^{(0)}$ and $\beta_{i,2}^{(0)}$ are within the acceptable bounds. The maximum is then selected.

This gives the following:

$$\beta_i^{(0)} = \begin{cases} \gamma_4 & \text{if } \max(\beta_{i,1}^{(0)}, \beta_{i,2}^{(0)}) < 1, \\ \gamma_2 & \text{if } \max(\beta_{i,1}^{(0)}, \beta_{i,2}^{(0)}) > \gamma_2, \\ \beta_{i,1}^{(0)} & \text{if } \beta_{i,1}^{(0)} \in [1, \gamma_2] \text{ and } \beta_{i,2}^{(0)} < 1, \\ \beta_{i,2}^{(0)} & \text{if } \beta_{i,2}^{(0)} \in [1, \gamma_2] \text{ and } \beta_{i,1}^{(0)} < 1, \\ \max(\beta_{i,1}^{(0)}, \beta_{i,2}^{(0)}) & \text{otherwise.} \end{cases} \tag{2.15}$$

Finally, three situations are considered in the third case for selecting $\beta_i^{(0)} \in [\gamma_3, \gamma_4]$, when $\mu_2 < |\rho_i^{(0)} - 1| \leq \mu_1$. Note that, since it is not clear from the value of $\rho_i^{(0)}$ that the radius estimate should be decreased or increased, $\beta_{i,1}^{(0)}$ and $\beta_{i,2}^{(0)}$ are trusted and indicate if a decrease or an increase is to be performed.

- *Both* $\beta_{i,1}^{(0)}$ and $\beta_{i,2}^{(0)}$ advise a decrease of the radius estimate, but smaller than the lower bound allowed. This lower bound, γ_3, is then adopted.

- *At least one* among $\beta_{i,1}^{(0)}$ and $\beta_{i,2}^{(0)}$ recommends an increase of the radius estimate, but larger than the upper bound allowed, γ_4. This upper bound is used.
- The maximal value, $\max(\beta_{i,1}^{(0)}, \beta_{i,2}^{(0)})$, belongs to the appropriate interval and defines $\beta_i^{(0)}$. The radius estimate is thus either increased or decreased, depending on this value.

That is:

$$\beta_i^{(0)} = \begin{cases} \gamma_3 & \text{if } \max(\beta_{i,1}^{(0)}, \beta_{i,2}^{(0)}) < \gamma_3, \\ \gamma_4 & \text{if } \max(\beta_{i,1}^{(0)}, \beta_{i,2}^{(0)}) > \gamma_4, \\ \max(\beta_{i,1}^{(0)}, \beta_{i,2}^{(0)}) & \text{otherwise.} \end{cases} \tag{2.16}$$

3. The algorithm. We are now in position to define our algorithm in full detail. Beside $\gamma_1, \gamma_2, \gamma_3, \gamma_4, \mu_1, \mu_2$ and θ as used in (2.5), (2.9), (2.12) and (2.13), the ITRR Algorithm depends on the constant $\mu_0 > 0$. This one determines the lowest acceptable level of agreement between the model and the objective function that must be reached at a radius estimate to become a candidate for the ITRR.

The iterations of Algorithm ITRR will be denoted by the index i. While the algorithm proceeds, Δ_{max} will record the current *maximal* radius estimate which guarantees a sufficient agreement between the model and the objective function. Finally, the imposed limit on the number of iterations will be denoted by $imax$ and fixes the degree of refinement used to determine the ITRR.

ITRR ALGORITHM.

Step 0. Initialization. Let the starting point $x^{(0)}$ be given. Compute $f^{(0)}$, $g^{(0)}$, and $B^{(0)}$. Choose or compute an ITRR estimate $\Delta_0^{(0)}$ and set $i = 0$. Also set $\Delta_{max} = 0$.

Step 1. Maximal radius estimate update. Compute $m_i^{(0)}$, $f_i^{(0)}$ and $\rho_i^{(0)}$ as defined in (2.7) and (2.8). If

$$|\rho_i^{(0)} - 1| \leq \mu_0, \tag{3.1}$$

set

$$\Delta_{max} = \max(\Delta_{max}, \Delta_i^{(0)}).$$

Step 2. Radius estimate update. If $i \geq imax$, go to Step 3. Otherwise, compute $\beta_{i,1}^{(0)}$ and $\beta_{i,2}^{(0)}$ using (2.12) and (2.13), respectively, compute

$$\beta_i^{(0)} \begin{cases} \text{using (2.14)} & \text{if } |\rho_i^{(0)} - 1| > \mu_1, \\ \text{using (2.15)} & \text{if } |\rho_i^{(0)} - 1| \leq \mu_2, \\ \text{using (2.16)} & \text{otherwise,} \end{cases}$$

and set

$$\Delta_{i+1}^{(0)} = \beta_i^{(0)} \Delta_i^{(0)}.$$

Increment i by one and go to Step 1.

Step 3. Final radius update. If $\Delta_{max} > 0$, set

$$\Delta^{(0)} = \Delta_{max}. \tag{3.2}$$

Otherwise, set

$$\Delta^{(0)} = \Delta_i^{(0)}. \tag{3.3}$$

Stop ITRR Algorithm.

The trust region algorithm may then begin, with $\Delta^{(0)}$ as ITRR.

We end this section by introducing an extra feature in the above scheme, which takes advantage of the computations of $f_i^{(0)}$, the function values at the trial points $x^{(0)} - \Delta_i^{(0)} \hat{g}^{(0)}$ (see Step 1). That is, during the search of an improved radius estimate, we simply monitor a possible decrease in the objective function at each trial point. Doing so, at the end of Algorithm ITRR, rather than updating the final radius, we move to the trial point that produced the *best decrease* in the objective function (if at least one decrease has been observed). This point then becomes a *new* starting point, at which Algorithm ITRR is repeated to compute a good ITRR. Of course, a limit is needed on the number of times the starting point is allowed to change. Denoting by $jmax$ this limit and by j the corresponding counter (initialized to one in Step 0), this extra feature may be incorporated in Algorithm ITRR using two further instructions. The first one, added at the end of Step 1, is

$$\text{If} \quad j \leq jmax \ \text{ and } \ (f^{(0)} - f_i^{(0)}) > \delta, \quad \text{set} \quad \delta = f^{(0)} - f_i^{(0)} \ \text{ and } \ \sigma = \Delta_i^{(0)}.$$

Here δ denotes the current best decrease observed in the objective function and σ stores the associated radius. (These two quantities should be initialized to zero in Step 0). The second instruction, which comes at the beginning of Step 3, is

$$\text{If} \quad j \leq jmax \ \text{ and } \ \delta > 0, \quad \text{set} \quad x^{(0)} \ \text{ to } \ x^{(0)} - \sigma \hat{g}^{(0)},$$

$$\text{increment } j \text{ by one and go to Step 0.}$$

When starting a trust region algorithm with a rather crude approximation of the solution, this kind of improvement, which exploits the steepest descent direction, may be very useful. It is particularly beneficial when the cost of evaluating the function is reasonable. A similar concept is used in truncated Newton methods (see [16], and [17]).

Note that a change in the starting point requires the computation of a new gradient and a new model, while the cost for determining the ITRR is estimated in terms of function evaluations. Suitable choices for the limits $imax$ and $jmax$ and for the constants used in Algorithm ITRR may depend on the problem type and will be discussed in §4.

4. Numerical results. For a good understanding of the results, it is necessary to give a rapid overview of the framework in which Algorithm ITRR has been embedded, namely the large-scale nonlinear optimization package LANCELOT/SBMIN (see [8]), designed for solving the bound-constrained minimization problem,

$$\text{minimize}_{x \in \mathbf{R}^n} \ f(x) \tag{4.1}$$

subject to the simple bound constraint

$$l \leq x \leq u, \tag{4.2}$$

where any of the bounds in (4.2) may be infinite.

SBMIN is an iterative trust region method whose version used for our testing has the following characteristics:

- Exact first and second derivatives are used.
- The trust region is defined using the infinity norm in (2.3) for each k.
- The trust region update strategy follows the framework (2.4)–(2.5), and implements a mechanism for contracting the trust region which is more sophisticated than that for expanding it (see [8], p. 116).
- The solution of the trust region subproblem at each iteration is accomplished in two stages. In the first, the *exact Cauchy point* is obtained to ensure a sufficient decrease in the quadratic model. This point is defined as the first local minimizer of $m^{(k)}(x^{(k)} + d^{(k)}(t))$, the quadratic model along the *Cauchy arc* $d^{(k)}(t)$ defined as

$$d^{(k)}(t) = P(x^{(k)} - tg^{(k)}, l^{(k)}, u^{(k)}) - x^{(k)}, \qquad t \geq 0, \tag{4.3}$$

  where $l^{(k)}$, $u^{(k)}$ and the projection operator $P(x, l^{(k)}, u^{(k)})$ are defined componentwise by $l_i^{(k)} = \max(l_i, x_i^{(k)} - \Delta^{(k)})$, $u_i^{(k)} = \min(u_i, x_i^{(k)} + \Delta^{(k)})$, and

$$P(x, l^{(k)}, u^{(k)})_i = \begin{cases} l_i^{(k)} & \text{if } x_i < l_i^{(k)}, \\ u_i^{(k)} & \text{if } x_i > u_i^{(k)}, \\ x_i & \text{otherwise.} \end{cases}$$

  The Cauchy arc (4.3) is continuous and piecewise linear, and the exact Cauchy point is found by investigating the model behaviour between successive pairs of breakpoints (points at which a trust region bound or a true bound is encountered along the Cauchy arc), until the model starts to increase. The variables which lie on their bounds at the Cauchy point (either a trust region bound or a true bound) are then fixed.
- The second stage applies a *truncated conjugate gradient* method (in which an 11-band preconditioner is used), to further reduce the quadratic model by changing the values of the remaining free variables.

The reader is referred to [8], Chapter 3, for a complete description of SBMIN.

We selected our 77 test examples as the majority of large and/or difficult nonlinear unconstrained or bound-constrained test examples in the CUTE (see [1]) collection. Only problems which took excessive cpu time (more than 5 hours), or excessive number of iterations (more than 1500), were excluded, since it was not clear that they would have added much to the results. All experiments were made in double precision, on a DEC 5000/200 workstation, using optimized (-O) Fortran 77 code and DEC-supplied BLAS.

The values for the constants of Algorithm ITRR used in our tests are $\gamma_1 = 0.0625$, $\gamma_2 = 5$, $\gamma_3 = 0.5$, $\gamma_4 = 2$, $\mu_0 = \mu_1 = 0.5$, $\mu_2 = 0.35$, and $\theta = 0.25$. The values for $\gamma_1, \gamma_3, \gamma_4$, and θ have been inspired from the trust region update strategy used in [8]. Suitable values for the other constants have been determined after extensive testing. (Note that, fortunately, slight variations for these constants have no significant impact on the behaviour of Algorithm ITRR). We set $jmax = 1$, meaning that at most one move is allowed in the starting point, and $imax = 4$, such that 5 radius estimates (including the first one) are examined per starting point. These values result from a compromise between the *minimum* number of radius estimates that should be sampled to produce a reasonable ITRR, and the *maximum* number of extra function evaluations which may amount to $(imax + 1) * (jmax + 1)$ (i.e. 10 here).

4.1. The quadratic case. Before introducing our results for the general nonlinear case, a preliminary study of LANCELOT's behaviour on quadratic problems is

presented in this section, that is intended to enlighten some of the characteristics of the specific trust region method implemented there. This should be helpful to set up a more adequate framework, in which a reliable interpretation of our testing for the general nonlinear case will become possible.

When the objective function f in problem (4.1)–(4.2) is a quadratic function, model (2.2) is identical to f (since exact second derivatives are used in (2.2)). The region where this model should be trusted is therefore infinite at any stage of a trust region algorithm. Hence, a logical choice for the ITRR in that case is $\Delta^{(0)} = \infty$. However, when no particular choice is specified by the user for the ITRR, LANCELOT does not make any distinction when solving a quadratic problem and sets $\Delta^{(0)} = 0.1\|g^{(0)}\|_2$. On the other hand, equations in (2.11) have no solution for $\theta > 0$ when both models $q_i^{(0)}(\beta)$ and $m^{(0)}(x^{(0)} - \beta d_i^{(0)})$ coincide (which is the case if f is a quadratic). Therefore, in order to circumvent this possibility, the next instruction has been added in Algorithm ITRR (before (3.1) in Step 1):

$$\text{(4.4)} \qquad \text{If} \quad \rho_i^{(0)} = 1, \quad \text{set} \quad \Delta_{max} = \infty \quad \text{and} \quad \text{go to Step 3.}$$

Note that this test *does not* ensure that f is a quadratic. If needed, a careful strategy should rather be developed to properly detect this special situation.

In order to compare both issues, we have tested quadratic problems from the CUTE collection, using LANCELOT with $\Delta^{(0)} = 0.1\|g^{(0)}\|_2$ and with Algorithm ITRR in which (4.4) has been added (see LAN and ITRR, respectively, in the tables below). Results are presented in Tables 1 and 2 for a representative sample of quadratic problems (6 unconstrained and 6 bound-constrained). In these tables and the following ones, n denotes the number of variables in the problem, "#its" is the number of major iterations needed to solve the problem, "#cg" reports the number of conjugate gradient iterations performed beyond the Cauchy point, and the last column gives the cpu times in seconds. Note that, for all the tests reported in this section, only one additional function evaluation has been needed by Algorithm ITRR to set $\Delta^{(0)} = \infty$.

TABLE 1
A comparison for the unconstrained quadratic problems.

Problem$_n$	$\Delta^{(0)}$		#its		#cg		time	
	LAN	ITRR	LAN	ITRR	LAN	ITRR	LAN	ITRR
DIXON3DQ$_{1000}$	5.7D-1	∞	3	1	3	1	1.99	0.90
DQDRTIC$_{1000}$	3.8D+3	∞	1	1	1	1	1.23	1.41
HILBERTA$_{100}$	4.8D+0	∞	7	1	22	14	16.76	7.50
HILBERTB$_{100}$	3.4D+1	∞	3	1	3	4	6.89	4.75
TESTQUAD$_{1000}$	2.0D+6	∞	4	1	20	16	4.75	2.88
TRIDIA$_{1000}$	3.7D+3	∞	1	1	1	1	0.96	1.05

Table 1 shows that, as expected, an infinite choice is the best when f is a quadratic function, and the problem is unconstrained. On the other hand, a substantial increase in the number of conjugate gradient iterations is observed in Table 2 (except for problem TORSIONF) when bound constraints are imposed, while the number of major iterations decreases. At first glance, these results may be quite surprising, but they closely depend on the LANCELOT package itself. This package includes a branch, after the conjugate gradient procedure, that allows re-entry of this conjugate gradient procedure when the convergence criterion (based on the relative residual) has been satisfied, but the step computed is small relative to the trust region radius and the model's gradient norm. This is intended to save major iterations, when possible. In

TABLE 2
A comparison for the bound-constrained quadratic problems.

Problem$_n$	$\Delta^{(0)}$		#its		#cg		time	
	LAN	ITRR	LAN	ITRR	LAN	ITRR	LAN	ITRR
JNLBRNG1$_{1024}$	5.8D-1	∞	6	5	93	220	25.58	50.86
JNLBRNG2$_{1024}$	1.3D+0	∞	4	3	68	128	17.40	27.81
OBSTCLAE$_{1024}$	7.9D-1	∞	4	3	410	438	140.55	147.75
OBSTCLAL$_{1024}$	8.0D-2	∞	8	7	48	117	15.55	24.75
TORSION1$_{1024}$	5.5D-2	∞	10	9	73	144	20.98	32.50
TORSIONF$_{1156}$	5.9D-2	∞	6	3	295	14	93.13	5.51

the absence of bound constraints, this avoids an early termination of the conjugate gradient process, allowing attainment of the solution in a single major iteration (see Table 1). In the presence of bounds however, these (possibly numerous) re-entries may not be justified as long as the correct set of active bounds has not yet been identified. This behaviour is detailed in Table 3 for a sequence of increasing initial radii, and exhibits, in particular, a high sensitivity to a variation of the ITRR, which is a rather undesirable feature.

TABLE 3
A comparison for a sequence of increasing initial trust region radii with LANCELOT.

Problem$_n$		Initial radius $\Delta^{(0)}$					
		0.1	1	10	100	1000	∞
JNLBRNG1$_{1024}$	#its	8	6	6	6	5	5
	#cg	103	115	120	142	166	220
	time	33.02	30.01	31.03	35.95	39.53	50.86
OBSTCLAL$_{1024}$	#its	8	8	8	7	7	7
	#cg	48	55	70	76	93	117
	time	14.64	15.73	18.52	18.28	20.97	24.75
TORSIONF$_{1156}$	#its	4	3	3	3	3	3
	#cg	312	11	12	13	13	14
	time	100.63	5.12	5.27	5.39	5.37	5.51

For comparison purposes, Tables 4 and 5 present the results when removing the aforementioned branch in LANCELOT. This time, an infinitely large value for the ITRR is justified. The conjugate gradient and timing results for Algorithm ITRR are much closer to those of LANCELOT in Table 4 than in Table 2, with a slightly better performance for problem OBSTCLAE and a slightly worse performance for problem JNLBRNG1 (even though a clear improvement occurred due to the branch removal). For problem JNLBRNG1 (as for others in our test set), a limited trust region acts as an extra safeguard to stop the conjugate gradient when the correct active set is not yet detected. This effect of the trust region may be considered as an advantage of trust region methods.

In order to complete the above analysis, we now consider problem TORSIONF in Table 2. This problem is characterized by a very large number of active bounds at the optimal solution, while most of the variables are free at the starting point. Because of the very small ITRR, the identification process of the correct active set during the Cauchy point determination is hindered. That is, during the early major iterations, the trust region bounds are all activated at the Cauchy point, without any freedom left for the conjugate gradient procedure. When the trust region has been slightly enlarged, besides trust region bounds, some of the true bounds are also

identified by the Cauchy point, but much fewer than the number that would be the case if the trust region was large enough. That is, the conjugate gradient procedure in LANCELOT is restarted each time a true bound is encountered (which occurs at almost every conjugate gradient iteration), in order to maintain the conjugacy between the directions, and the iteration is stopped only when a trust region bound is reached. All this produces extra linear-algebraic costs that greatly deteriorates the algorithm's performance. On the other hand, when starting with a large ITRR, a good approximation of the correct active set is immediately detected by the Cauchy point, and very little work has to be performed during the conjugate gradient process. This observation strengthens the priority given to a large choice for the ITRR, when possible.

TABLE 4
A comparison for the bound-constrained quadratic problems (modified version).

Problem$_n$	#its		#cg		time	
	LAN	ITRR	LAN	ITRR	LAN	ITRR
JNLBRNG1$_{1024}$	6	6	93	112	25.59	30.33
JNLBRNG2$_{1024}$	5	5	66	66	17.85	18.11
OBSTCLAE$_{1024}$	4	4	405	361	139.93	124.93
OBSTCLAL$_{1024}$	8	8	47	47	14.63	14.19
TORSION1$_{1024}$	10	10	76	76	21.59	21.53
TORSIONF$_{1156}$	6	4	295	11	93.06	5.55

TABLE 5
A comparison for a sequence of increasing initial trust region radii with LANCELOT *(modified version).*

Problem$_n$		Initial radius $\Delta^{(0)}$			
		0.1	1	10	∞
JNLBRNG1$_{1024}$	#its	9	6	6	6
	#cg	99	112	112	112
	time	33.43	30.38	30.57	30.33
OBSTCLAL$_{1024}$	#its	8	8	8	8
	#cg	47	47	47	47
	time	14.21	14.24	14.28	14.19
TORSIONF$_{1156}$	#its	5	4	4	4
	#cg	312	11	11	11
	time	102.02	5.66	5.61	5.55

In the light of the above analysis, we tested the 77 nonlinear problems with the original version of LANCELOT versus a modified version, where the extra feature to improve a too small step on output of the conjugate gradient process has been ignored. Slight differences in the results have generally been observed, that were more often in favour of the modified version. For this reason and in order to avoid an excessive sensitivity of the method to the trust region size as well as preventing a large choice for the ITRR (especially when this choice reflects a real adequacy between f and its model), we decided to use the modified version for the testing of the nonlinear case presented in the next section.

4.2. The general case. In order to test Algorithm ITRR, we ran LANCELOT successively with:

- Algorithm ITRR, starting with $\Delta_0^{(0)} = 0.1\|g^{(0)}\|_2$ as ITRR estimate;

- $\Delta^{(0)} = 0.1\|g^{(0)}\|_2$ (the ITRR computed by LANCELOT when no other choice is made by the user);
- $\Delta^{(0)} = \nu\|g^{(0)}\|_2$ where $\nu = \|g^{(0)}\|_2^2/(g^{(0)})^T B^{(0)} g^{(0)}$ (the distance to the unconstrained Cauchy point, as suggested by Powell in [13]), except when the quadratic model is indefinite, in wich case we omitted the test.

The detailed results are summarized in Tables 6 and 7 for the 64 unconstrained problems (possibly including some fixed variables), and in Table 8 for the 13 bound-constrained problems (see ITRR, LAN and CAU, respectively). For each case, the number of major iterations ("#its") and the cpu times in seconds ("time") are reported. Note that the number of function evaluations may then be easily deduced : for LANCELOT *without* Algorithm ITRR, it is the number of major iterations plus 1, while for LANCELOT *with* Algorithm ITRR, it is the number of major iterations plus 12 if the starting point is refined once (what is pointed out by an asterisk in the first column), and plus 6 otherwise. The tables also present the relative performances for the number of function evaluations, the number of major iterations and the cpu times (see "%f", "%its", and "%time", respectively), computed as

$$\frac{|\star_{\rm LAN} - \star_{\rm ITRR}|}{\star_{\rm LAN}} \times 100 \qquad \text{and} \qquad \frac{|\star_{\rm CAU} - \star_{\rm ITRR}|}{\star_{\rm CAU}} \times 100,$$

where "$\star$" is, in turn, the number of function evaluations, "#its", and "time". In these tables, a "+" indicates when the performance is in favour of Algorithm ITRR and a "−" when not. Note that a difference of less than five percent in the cpu times is regarded as insignificant.

The results first show that, all in all, Algorithm ITRR improves the overall cpu time performance of LANCELOT for a large number of problems:

- 46 improvements against 13 deteriorations and 18 ties when comparing with LAN;
- 39 improvements against 19 deteriorations and 12 ties when comparing with CAU.

More importantly, when they exist, these improvements may be quite significant (19 of them are greater than 30% when comparing with LAN, while 21 of them are greater than 30% when comparing with CAU), and confirm the impact the ITRR choice may have on the method behaviour. On the other hand, the damage is rather limited when it occurs (except for a few cases). Note that the larger number of improvements observed when comparing with LAN does not mean that the ITRR computed by LANCELOT is worse than the distance to the unconstrained Cauchy point. Actually, the improvements when comparing Algorithm ITRR with CAU are generally more significant, and the results show that, on average, LAN and CAU may be considered as equivalent (when compared together).

As pointed out by the number of asterisks in the first column of Tables 6 to 8, a change in the starting point occurs very often and makes a significant contribution to the good performance observed. Columns 4 and 7 detail the relative extra cost in terms of function evaluations produced by Algorithm ITRR. Note that, in the current case where the starting point is refined once, the (fixed) extra cost incurs up to 11 extra function evaluations, which is quite high on average, compared with the total number of function evaluations. However, considering the relative performance in the cpu times, the extra cost is generally covered, sometimes handsomely, by the saving produced in the number of major iterations (that is, when %its is positive, %time is generally positive too). Only few cases produce a saving that just balances the extra

TABLE 6
A comparison for the unconstrained problems.

Problem$_n$	ITRR	LAN			CAU			ITRR	LAN		CAU	
	#its	#its	%f	%its	#its	%f	%its	time	time	%time	time	%time
ARWHEAD$^*_{1000}$	4	5	−167	+20	5	−167	+20	6.4	6.4	0	6.4	0
BDQRTIC$^*_{1000}$	10	11	−83	+9	11	−83	+9	20.8	21.6	4	21.6	4
BRATU1D$^*_{1001}$	5	5	−183	0	5	−183	0	7.8	6.8	−15	6.8	−15
BROWNAL$^*_{100}$	4	3	−300	−33	3	−300	−33	19.0	17.0	−12	17.0	−12
BROYDN7D$^*_{1000}$	92	74	−39	−24	73	−41	−26	87.8	72.5	−21	71.1	−23
BRYBND$^*_{1000}$	14	16	−53	+12	30	+16	+53	35.6	41.3	+14	72.6	+51
CHNROSNB$^*_{50}$	39	67	+25	+42	75	+33	+48	2.0	3.2	+37	3.5	+43
CLPLATEA$^*_{1024}$	6	7	−125	+14	5	−200	−20	35.6	38.3	+7	31.3	−14
CLPLATEB$^*_{1024}$	3	8	−67	+62	4	−200	+25	25.0	38.0	+34	24.5	2
CLPLATEC$_{1024}$	4	6	−43	+33	14	+33	+71	258.7	286.0	+10	417.3	+38
CRAGGLVY$^*_{1000}$	11	14	−53	+21	14	−53	+21	14.0	15.8	+11	15.8	+11
DIXMAANA$^*_{1500}$	3	9	−50	+67	8	−67	+62	6.1	13.8	+56	12.7	+52
DIXMAANB$^*_{1500}$	3	9	−50	+67	9	−50	+67	6.2	14.6	+58	14.2	+56
DIXMAANC$^*_{1500}$	4	11	−33	+64	15	0	+73	7.5	17.3	+57	23.6	+68
DIXMAAND$^*_{1500}$	4	11	−33	+64	10	−45	+60	8.2	18.4	+55	17.1	+52
DIXMAANE$^*_{1500}$	12	7	−200	−71	8	−167	−50	21.7	13.2	−64	14.2	−53
DIXMAANF$^*_{1500}$	17	17	−61	0	49	+42	+65	27.9	26.7	4	74.1	+62
DIXMAANG$^*_{1500}$	12	24	+4	+50	49	+52	+76	20.8	39.0	+47	75.2	+72
DIXMAANH$^*_{1500}$	7	27	+32	+75	34	+46	+79	13.4	43.2	+69	52.7	+75
DIXMAANI$^*_{1500}$	6	8	−100	+25	9	−80	+33	12.6	15.8	+20	17.6	+28
DIXMAANJ$^*_{1500}$	37	49	+2	+24	34	−40	−9	59.8	82.9	+28	56.0	−7
DIXMAANK$^*_{1500}$	9	31	+34	+71	37	+45	+76	18.3	52.7	+65	60.7	+70
DIXMAANL$^*_{1500}$	19	48	+37	+60	59	+48	+68	32.1	83.8	+62	112.8	+72
DQRTIC$_{1000}$	28	28	−17	0	28	−17	0	18.3	18.2	1	18.2	1
EDENSCH$^*_{1000}$	7	12	−46	+42	12	−46	+42	10.0	15.0	+33	15.3	+35
EIGENALS$^*_{930}$	60	50	−41	−20				1678.9	1416.3	−19		
EIGENBLS$^*_{110}$	165	185	+5	+11	197	+11	+16	302.5	339.1	+11	357.6	+15
EIGENCLS$^*_{462}$	481	518	+5	+7	472	−4	−2	8688.2	8715.5	0	8216.1	−6
ENGVAL1$^*_{1000}$	7	7	−137	0	7	−137	0	7.9	7.5	−5	7.5	−5
ERRINROS$^*_{50}$	59	67	−4	+12	68	−3	+13	2.9	3.2	+9	3.1	+6
EXTROSNB$^*_{1000}$	7	883	+98	+99	863	+98	+99	6.5	733.0	+99	821.4	+99
FLETCHCR$_{100}$	224	224	−2	0	224	−2	0	19.8	19.6	1	19.7	1
FMINSURF$^*_{1024}$	293	227	−34	−29	315	+3	+7	1532.0	1179.3	−30	1617.5	+5
FREUROTH$^*_{1000}$	8	17	−11	+53				10.4	17.3	+40		
GENROSE$^*_{1000}$	1194	1290	+7	+7	1100	−10	−9	1023.8	1115.6	+8	920.3	−11
LIARWHD$^*_{1000}$	14	16	−53	+12	16	−53	+12	15.8	17.2	+8	17.1	+8
LMINSURF$^*_{1024}$	306	272	−16	−12	157	−101	−95	412.4	380.8	−8	279.7	−47
MANCINO$_{100}$	15	13	−50	−15	10	−91	−50	85.7	74.4	−15	58.1	−48
MOREBV$^*_{1000}$	1	8	−44	+87	1	−550	0	1.8	7.3	+75	1.3	−38
MSQRTALS$^*_{1024}$	30	38	−8	+21				10254.8	13031.2	+21		
MSQRTBLS$^*_{1024}$	31	34	−23	+9				6573.9	6925.5	+5		
NCB20$^*_{1000}$	145	143	−9	−1	159	+2	+9	1884.4	1887.2	0	2046.0	+8
NCB20B$^*_{1000}$	20	23	−33	+13	22	−39	+9	1979.0	1999.9	1	2577.0	+23
NLMSURF$^*_{1024}$	772	844	+7	+9	750	−4	−3	1158.0	1268.5	+9	1141.7	1

TABLE 7
A comparison for the unconstrained problems (end).

Problem$_n$	ITRR	LAN			CAU			ITRR	LAN		CAU	
	#its	#its	%f	%its	#its	%f	%its	time	time	%time	time	%time
NONDIA$_{1000}$	6	6	−71	0	6	−71	0	5.6	5.1	−10	5.1	−10
NONDQUAR$^*_{1000}$	13	17	−39	+24	17	−39	+24	11.9	13.5	+12	13.6	+12
ODC$^*_{992}$	12	16	−41	+25	12	−85	0	87.0	87.5	1	82.2	−6
PENALTY1$_{1000}$	52	59	+3	+12	45	−26	−16	311.5	350.1	+11	259.2	−20
PENALTY2$^*_{100}$	17	19	−45	+11	19	−45	+11	4.4	4.5	2	4.6	4
PENALTY3$^*_{100}$	34	32	−39	−6				420.0	382.2	−10		
POWELLSG$^*_{1000}$	10	15	−37	+33	15	−37	+33	7.5	10.5	+29	10.9	+31
POWER$_{1000}$	27	27	−18	0	27	−18	0	116.9	118.8	2	121.7	4
QUARTC$_{1000}$	28	28	−17	0	28	−17	0	18.3	18.2	1	18.5	1
RAYBENDL$^*_{124}$	306	366	+13	+16	325	+2	+6	32.0	38.9	+18	34.1	+6
RAYBENDS$^*_{50}$	70	52	−55	−35	65	−24	−8	29.0	21.3	−36	24.4	−19
SINQUAD$^*_{1000}$	107	132	+11	+19	128	+8	+16	171.2	216.6	+21	208.6	+18
SPMSRTLS$^*_{1000}$	11	19	−15	+42				19.9	32.2	+38		
SROSENBR$^*_{1000}$	5	12	−31	+58	12	−31	+58	4.1	8.6	+52	8.6	+52
SSC$^*_{992}$	2	6	−100	+67	2	−367	0	24.0	32.2	+25	25.7	+7
TOINTGSS$^*_{1000}$	0	1	−500	+100	1	−500	+100	0.7	1.3	+46	1.3	+46
TQUARTIC$^*_{1000}$	16	15	−75	−7	1	−1300	−1500	18.4	17.7	4	3.1	−494
VAREIGVL$^*_{1000}$	11	13	−64	+15	13	−64	+15	62.7	71.7	+13	71.6	+12
WATSON$^*_{31}$	10	10	−100	0	11	−83	+9	4.7	5.0	+6	5.3	+11
WOODS$^*_{1000}$	11	16	−35	+31	16	−35	+31	9.9	13.5	+27	13.6	+27

TABLE 8
A comparison for the bound−constrained problems.

Problem$_n$	ITRR	LAN			CAU			ITRR	LAN		CAU	
	#its	#its	%f	%its	#its	%f	%its	time	time	%time	time	%time
BDEXP$^*_{1000}$	0	10	−9	+100	10	−9	+100	0.7	11.9	+94	13.1	+95
CHEBYQAD$^*_{50}$	63	64	−15	+2	54	−36	−17	126.0	128.0	2	106.9	−18
EXPLIN2$^*_{120}$	10	12	−69	+17	12	−69	+17	0.4	0.4	0	0.4	0
EXPLIN$^*_{120}$	10	13	−57	+23	13	−57	+23	0.3	0.5	+40	0.6	+50
EXPQUAD$^*_{120}$	12	18	−26	+33	20	−14	+40	2.1	2.6	+19	2.8	+25
HADAMALS$^*_{32}$	46	49	−16	+6	49	−16	+6	3854.9	4536.7	+15	4609.9	+16
LINVERSE$^*_{999}$	75	21	−295	−257	33	−156	−127	622.7	207.9	−200	460.2	−35
MCCORMCK$^*_{1000}$	3	6	−114	+50	10	−36	+70	4.3	5.6	+23	9.0	+52
NONSCOMP$^*_{1000}$	11	8	−156	−37	8	−156	−37	7.2	6.9	4	6.9	4
QR3DLS$^*_{610}$	236	232	−6	−2	234	−6	−1	15371.9	15584.8	1	15762.2	2
QRTQUAD$^*_{120}$	118	173	+25	+32	315	+59	+63	11.9	16.2	+27	28.4	+58
S368$^*_{100}$	7	8	−111	+12				78.6	75.8	4		
SPECAN$^*_3$	9	15	−31	+40	12	−62	+25	503.7	556.1	+9	559.7	+10

function evaluations (see the problems for which %its > 0 and $0 \leq$ %time < 5), while never a saving occurs which does not counterbalance the additional work. On the other hand, when a deterioration occurs in the cpu times (%time < 0), it is rarely due to the extra cost of Algorithm ITRR *exclusively* (%its $= 0$). As a consequence, excepting when functions are very expensive, the use of Algorithm ITRR may be considered efficient and relatively cheap compared with the overall cost of the problem solution.

We have observed that only 4 problem tests terminated Algorithm ITRR using update (3.3), while a successful maximal radius satisfying condition (3.1) was selected in the 73 other cases. We also experimented with a simpler choice for $\beta_i^{(0)}$ ($\beta_i^{(0)} = 2$ when $\rho_i^{(0)}$ is close enough to one and $\beta_i^{(0)} = \frac{1}{2}$ otherwise), that resulted in a markedly inferior performance compared with that of Algorithm ITRR. This proves the necessity of a sophisticated selection procedure for $\beta_i^{(0)}$, that allows a swift recovery from a bad initial value for the ratio $\rho_i^{(0)}$.

We conclude this analysis by commenting on the negative results of Algorithm ITRR on problem TQUARTIC (see Table 7), when comparing with CAU, and on problem LINVERSE (see Table 8), especially when comparing with LAN. For problem TQUARTIC (a quartic), the ITRR computed by both LANCELOT and Algorithm ITRR is quite small and prevents from doing rapid progress to the solution. The trust region hence needs to be enlarged several times during the minimization algorithm. On the other hand, the distance to the Cauchy point is sufficiently large to allow solving the problem in one major iteration. For problem LINVERSE, the ITRR selected by Algorithm ITRR corresponds to an excellent agreement between the function and the model in the steepest descent direction. However, the starting point produced by Algorithm ITRR, although reducing significantly the objective function value, requires more work from the trust region method to find the solution. This is due to a higher nonlinearity of the objective function in the region where this new point is located and is, in a sense, just bad luck! When testing Algorithm ITRR with $jmax = 0$ on this problem, the same ITRR as LANCELOT had been selected, hence producing the same performance. On the other hand, we also tested a series of slightly perturbed initial trust region radii, and observed a rapid deterioration of the performance of the method. Problem LINVERSE is thus very sensitive to the ITRR choice. Note that this sensitivity has been observed on other problems during our testing, and leads to the conclusion that a good ITRR sometimes may not be a sufficient condition to guarantee an improvement of the method.

We finally would like to note that no modification has been made in Algorithm ITRR (nor a *constrained* Cauchy point for CAU has been considered), when solving the bound-constrained problems reported in the paper. The purpose here was simply to illustrate the proposed method on a larger sample than only unconstrained problems. Of course, the author is aware that in the presence of bound constraints, a more reliable version of Algorithm ITRR should include a projection of each trial point onto the bound constraints.

We end this section by briefly commenting on the choice of the constants and upper bounds on the iteration counters of Algorithm ITRR. Although a reasonable choice has been used for the testing presented in this paper, a specific one could be adapted depending on the a priori knowledge of a given problem. If, for instance, function evaluations are costly, a lower value for the bounds $imax$ and $jmax$ could be selected. Note however that $imax$ should not be chosen excessively small, in order to be fairly sure that condition (3.1) will be satisfied (unless this condition is suitably relaxed by choosing the value of μ_0). On the other hand, if the starting point is

known to be far away from the solution, it may be worthwhile to increase the value of $jmax$, provided the function is cheap to evaluate. Improved values for the remaining constants $\gamma_1, \gamma_2, \gamma_3, \gamma_4, \mu_1, \mu_2$ and θ closely depend on a knowledge of the level of nonlinearity of the problem.

5. Conclusions and perspectives. In this paper, we propose an automatic strategy to determine a reliable ITRR for trust region type methods. This strategy mainly investigates the adequacy between the objective function and its model in the steepest descent direction available at the starting point. It further includes a specific method for refining the starting point by exploiting the extra function evaluations performed during the ITRR search.

Numerical tests are reported and discussed, showing the efficiency of the proposed approach and giving additional insights to trust region methods for unconstrained and bound-constrained optimization. The encouraging results suggest some direction for future research, such as the use of a truncated Newton direction computed at the starting point rather than the steepest descent direction for the search of an ITRR. An adaptation of the algorithm for methods designed to solve general constrained problems is presently being studied.

Acknowledgement. The author wishes to thank an anonymous referee for suggesting a comparison of Algorithm ITRR with the choice of setting the ITRR to the distance to the Cauchy point (as in [13]). Thanks are also due to Andy Conn, Nick Gould, Philippe Toint and Michel Bierlaire who contributed to improve the present manuscript.

REFERENCES

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

[2] J. V. Burke, J. J. Moré, and G. Toraldo, *Convergence properties of trust region methods for linear and convex constraints*, Mathematical Programming, Series A, 47 (1990), pp. 305–336.

[3] R. H. Byrd, R. B. Schnabel, and G. A. Schultz, *A trust region algorithm for nonlinearly constrained optimization*, SIAM Journal on Numerical Analysis, 24 (1987), pp. 1152–1170.

[4] P. H. Calamai and J. J. Moré, *Projected gradient methods for linearly constrained problems*, Mathematical Programming, 39 (1987), pp. 93–116.

[5] M. R. Celis, J. E. Dennis, and R. A. Tapia, *A trust region strategy for nonlinear equality constrained optimization*, in Numerical Optimization 1984, P. T. Boggs, R. H. Byrd, and R. B. Schnabel, eds., Philadelphia, USA, 1985, SIAM, pp. 71–82.

[6] A. R. Conn, N. Gould, A. Sartenaer, and P. L. Toint, *Global convergence of a class of trust region algorithms for optimization using inexact projections on convex constraints*, SIAM Journal on Optimization, 3 (1993), pp. 164–221.

[7] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Global convergence of a class of trust region algorithms for optimization with simple bounds*, SIAM Journal on Numerical Analysis, 25 (1988), pp. 433–460. See also same journal 26:764–767, 1989.

[8] ———, LANCELOT: *a Fortran package for large-scale nonlinear optimization (Release A)*, no. 17 in Springer Series in Computational Mathematics, Springer Verlag, Heidelberg, Berlin, New York, 1992.

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

[10] R. Fletcher, *Practical Methods of Optimization: Unconstrained Optimization*, J. Wiley and Sons, New-York, 1980.

[11] J. J. Moré, *Recent developments in algorithms and software for trust region methods*, in Mathematical Programming: The State of the Art, A. Bachem, M. Grötschel, and B. Korte, eds., Berlin, 1983, Springer Verlag, pp. 258–287.

[12] ———, *Trust regions and projected gradients*, in System Modelling and Optimization, M. Iri and K. Yajima, eds., vol. 113, Berlin, 1988, Springer Verlag, pp. 1–13. Lecture Notes in Control and Information Sciences.

[13] M. J. D. Powell, *A Fortran subroutine for solving systems of nonlinear algebraic equations*, in Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz, ed., London, 1970, Gordon and Breach, pp. 115–161.

[14] ———, *A new algorithm for unconstrained optimization*, in Nonlinear Programming, J. B. Rosen, O. L. Mangasarian, and K. Ritter, eds., New York, 1970, Academic Press.

[15] ———, *On the global convergence of trust region algorithms for unconstrained minimization*, Mathematical Programming, 29 (1984), pp. 297–303.

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

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

[18] ———, *Global convergence of a class of trust region methods for nonconvex minimization in Hilbert space*, IMA Journal of Numerical Analysis, 8 (1988), pp. 231–252.

[19] A. Vardi, *A trust region algorithm for equality constrained minimization: convergence properties and implementation*, SIAM Journal on Numerical Analysis, 22 (1985), pp. 575–591.