An example of non-convergence of trust region algorithms

by

Ya-xiang Yuan

Report No. **ICM-96-067** September 1996

An example of non-convergence of trust region algorithms

Ya-xiang Yuan

State Key Laboratory of Scientific and Engineering Computing

Institute of Computational Mathematics and Scientific/Engineering Computing

Chinese Academy of Sciences, POB 2719, Beijing 100080, China

September 1996

Abstract

It is well known that trust region algorithms have very nice convergence properties. Trust region algorithms can be classified into two kinds: one requires sufficient reduction in objective function value (merit function value, in the case of constrained optimization), the other only needs reduction in objective function value. In general, it can be shown that the algorithms that require sufficient reductions have strong convergence result, namely all accumulation points are stationary points. The algorithms that do not require sufficient reductions have the nice properties of accepting any better iterates, but the convergence result is weak, only one accumulation point is a stationary point.

In this paper, we construct an example to show that it can happen that for a class of trust region algorithms that do not require sufficient reductions the whole sequence need not to converge. In our example, only one accumulation point is a stationary point while all other accumulation points are non-stationary points.

Key words: nonlinear optimization, trust region algorithms, non-convergence

1. Introduction

Trust region algorithms are relatively new algorithms. The trust region approach is strongly associated with approximation. Assume we have a current guess of the solution of the optimization problem, an approximate model can be constructed near the current point. A solution of the approximate model can be taken as the next iterate point. In fact, most line search algorithms also solve approximate models to obtain search directions. However, in a trust region algorithm, the approximate model is only "trusted" in a region near the current iterate. This seems reasonable, because for general nonlinear functions local approximate models (such as linear approximation and quadratic approximation) can only fit the original function locally. The region that the approximate model is trusted is called a trust region, which is normally a neighbourhood centered at the current iterate. The trust region is adjusted from iteration to iteration. Roughly speaking, if the computations indicate the approximate model fit the original problem quite well, the trust region can be enlarged. Otherwise when the approximate model seems to be not good enough (for example, a solution of the approximate model turns out to be a "bad" point), the trust region will be reduced.

The key contents of a trust region algorithm are how to compute the trust region trial step and how to decide whether a trial step should be accepted. An iteration of a trust region algorithm has the following form. At the beginning of the iteration, a trust region is available. An approximate model is constructed, and it is solved within the trust region, giving a solution s_k which is called the trial step. A merit function is chosen, which is used for updating the next trust region and for choosing the new iterate point.

The phenomenon that we are going to reveal is about cycle property of a class of trust region algorithms. To make the analysis simple, we consider an example of unconstrained optimization. It can be easily seen that similar examples exist for trust region algorithms for constrained optimization.

Consider the unconstrained optimization problem:

$$\min_{x\in\Re^n} f(x) \tag{1.1}$$

where $f(x)$ is a nonlinear continuous differentiable function in $\Re^n$. The trust region trial step s_k is a solution or an approximate solution of the so called "trust region subproblem":

$$\min_{d\in\Re^n} g_k^T d + \frac{1}{2} d^T B_k d = \phi_k(d) \tag{1.2}$$

$$s.\ t. \qquad ||d||_2 \le \Delta_k \tag{1.3}$$

where $g_k = \nabla f(x_k)$ is the gradient at the current approximate solution, B_k is an $n \times n$ symmetric matrix which approximates the Hessian of $f(x)$ and $\Delta_k > 0$ is a trust region radius. Besides the computations of the trial step s_k, deciding whether s_k can be accepted is another key issue of a trust region algorithm. After the trial step s_k is calculated, The reduction in the approximate model, that is

$$Pred_k = \phi_k(0) - \phi_k(s_k), \tag{1.4}$$

is called the predicted reduction. The actual reduction in the objective function is

$$Ared_k = f(x_k) - f(x_k + s_k). \tag{1.5}$$

The ratio between the actual reduction and the predicted reduction

$$r_k = \frac{Ared_k}{Pred_k} \tag{1.6}$$

plays an essential role in deciding whether the trial step s_k should be accept and in setting the length of trust region radius for the next iteration.

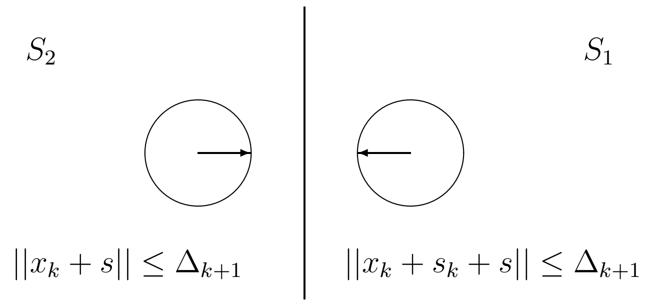
A general trust region algorithm for unconstrained optimization can be given as follows.

Algorithm 1.1 *(Trust Region Algorithm for Unconstrained Optimization)*

```
Step 1  Given $x_1 \in \Re^n$, $\Delta_1 > 0$, $\epsilon \geq 0$, $B_1 \in \Re^{n\times n}$ symmetric;
        $0 < \tau_3 < \tau_4 < 1 < \tau_1$, $0 \leq \tau_0 \leq \tau_2 < 1$, $\tau_2 > 0$, $k := 1$.
Step 2  If $||g_k||_2 \leq \epsilon$ then stop;
        Solve (1.2)-(1.3) giving $s_k$.
Step 3  Compute $r_k$;
            $x_{k+1} = \begin{cases} x_k & \text{if } r_k \leq \tau_0, \\ x_k + s_k & \text{otherwise}; \end{cases}$    (1.7)
        Choose $\Delta_{k+1}$ that satisfies
            $\Delta_{k+1} \in \begin{cases} [\tau_3||s_k||_2,\ \tau_4\Delta_k] & \text{if } r_k < \tau_2, \\ [\Delta_k,\ \tau_1\Delta_k] & \text{otherwise}; \end{cases}$    (1.8)
Step 4  Update $B_{k+1}$;
        $k := k+1$; go to Step 2.
```

The constants τ_i (i=0,..,4) can be chosen by users. Typical values are $\tau_0 = 0, \tau_1 = 2, \tau_2 = \tau_3 = 0.25, \tau_4 = 0.5$. For other choices of those constants, please see [3], [2], [4], [6], etc.. The parameter τ_0 is usually zero (e.g. [3], [5]) or a small positive constant (e.g. [1] and [7]).

Convergence results and proofs of the above algorithm are independent of the values of $c_i (i = 1, 2, 3, 4)$. They are also the same for all $c_0 \in (0, 1)$. However, if $c_0 = 0$, the proof and result can be obtained will be different. the advantage of using zero τ_0 is that a trial step is accepted whenever the objective function is reduced. Hence it would not throw away a "good point", which is a desirable property especially when the function evaluations are very expensive. As pointed by [8], another intuitive argument for preferring $\tau_0 = 0$ is as follows. Consider the case that $r_k > 0$. No matter how small the ratio r_k is, the objective function $f(x)$ has a smaller function value at $x_k + s_k$ than at x_k. Hence intuitively one would expect that the minimum of the objective function should be closer to $x_k + s_k$ than to x_k. In other words, it is more likely that the solution of the original problem is in the half space $S_1 = \{s \mid ||x_k + s_k + s|| \leq ||x_k + s||\}$ instead of $S_2 = \{s \mid ||x_k + s|| \leq ||x_k + s_k + s||\}$ (see Picture 1.1). Normally trust region algorithms reduce the new trust region bound to at most a half of $||s_k||$ whenever s_k is rejected ($x_{k+1} = x_k$), Hence for those algorithms that reject s_k, the trust region for the next iteration will be $\{s \mid ||x_k + s|| \leq \Delta_{k+1} \leq ||s_k||/2\}$ which is a subset of S_2. That contradicts to our above rough analyses that indicate the solution is more likely in S_1. Hence we believe it is better to set $x_{k+1} = x_k + s_k$ in this case, which will enable the next trust region in S_1. That is to say, intuitively it is better to set $x_{k+1} = x_k + s_k$ whenever $r_k > 0$.


S_2
S_1
$||x_k + s|| \leq \Delta_{k+1}$
$||x_k + s_k + s|| \leq \Delta_{k+1}$

Picture 1.1

But, the price we pay for letting $\tau_0 = 0$ is that the global convergence result is only

$$\liminf_{k\to\infty} \quad ||g_k||_2 = 0 \tag{1.9}$$

instead of

$$\lim_{k\to\infty} ||g_k||_2 = 0 \tag{1.10}$$

which can be achieved if $\tau_0 > 0$.

The main aim of this paper is to investigate for algorithms with $\tau = 0$ whether the convergence result (1.9) can be improved. We will construct an example to show that the sequence $\{x_k\}$ generated satisfies

$$\limsup_{k\to\infty} \quad ||g_k||_2 > 0. \tag{1.11}$$

Therefore the result (1.9) can not be further improved. Indeed, in our example, $\{x_k\}$ has three accumulation point, one of them is a stationary and the two others are not.

In the next section, we give the main idea of constructing the example, and the example is given in section 3. Finally a short discussion is also given.

2. Idea of Example

In our example, we force the sequence $\{x_k\}$ cycle. From the weakly convergence result (1.9), at least one of the accumulation point is a stationary point. Denote $\bar{x}$ is a stationary accumulation point. We also require the trust region radius cycles. Because we need to have some accumulation points that are not stationary, at the iterations near those "bad" accumulation point the predicted reduction will be bounded away from zero. Because the sequence cycle implies that the sequence $\{f(x_k\}$ will be bounded below, it is easy to see that the actual reduction $Ared_k$ will converge to zero. Therefore at the iterations that are close to "bad" accumulation points, the ratio between actual reduction and predicted reduction will converges to zero. Thus the trust region radius have to be reduced at these "bad" iterations. In order to make the trust region radius cycle, the trust region bound should be increased at iterations near the good accumulation point $\bar{x}$.

When $r_k < \tau_2$, some algorithms let $\Delta_{k+1} = \tau_4\Delta_k$ and some $\Delta_{k+1} = \tau_4||s_k||$. Furthermore, some algorithms increase the trust region bound only when the $r_k > \tau_2$ and $||s_k|| = \Delta_k$. In order to make our example valid for many algorithms, we force

$$||s_k|| = \Delta_k \tag{2.1}$$

for all k. The above relation and the fact that Δ_k has to be reduced at iterations near a "bad" accumulation point indicate that it is not possible for the sequence $\{x_k\}$ cycle near only two points.

We will construct the example as simple as possible. We consider a one-dimensional example with three accumulation points: $\bar{x}$, $\hat{x}$, $\tilde{x}$. $\bar{x}$ is a stationary point, but $\hat{x}$ and $\tilde{x}$ are not stationary. Let $\bar{k}$, $\hat{k} = \bar{k}+1$, and $\tilde{k} = \bar{k}+2$ are the indices of the iterations near $\bar{x}$, $\hat{x}$, and $\tilde{x}$ respectively. $\overline{k+1} = \bar{k}+3$. Our discussions given above imply that

$$\lim_{\hat{k}\to\infty} r_{\hat{k}} = 0, \quad \lim_{k\to\infty} r_{\tilde{k}} = 0. \tag{2.2}$$

Let

$$\lim_{\bar{k}\to\infty} \Delta_{\bar{k}} = \bar{\Delta}, \quad \lim_{\hat{k}\to\infty} \Delta_{\hat{k}} = \hat{\Delta}, \quad \lim_{\tilde{k}\to\infty} \Delta_{\tilde{k}} = \tilde{\Delta}. \tag{2.3}$$

It follows from the above relations and (2.1)-(2.2) that

$$\bar{\Delta}_k < \tilde{\Delta}_k < \hat{\Delta}_k. \tag{2.4}$$

This inequality shows that

$$\Delta_{\bar{k}+1} > \Delta_{\bar{k}} \tag{2.5}$$

for all sufficiently large $\bar{k}$, which requires that

$$r_{\bar{k}} > \tau_2. \tag{2.6}$$

We let

$$\bar{x} = 0, \quad \hat{x} = -1, \quad \tilde{x} = 2, \tag{2.7}$$

which, together with (2.1) and (2.3), requires that

$$\bar{\Delta} = 1, \quad \hat{\Delta} = 3, \quad \tilde{\Delta} = 2. \tag{2.8}$$

Many algorithms use positive semi-definite approximate Hessian B_k. Thus it from

$$f(x_{\overline{k+1}}) < f(x_{\tilde{k}}) < f(x_{\hat{k}}) < f(x_{\bar{k}}) \tag{2.9}$$

and (2.7) that

$$\nabla f(x_{\bar{k}}) > 0, \quad \nabla f(x_{\hat{k}}) < 0, \quad \nabla f(x_{\tilde{k}}) > 0. \tag{2.10}$$

(2.9) also implies that

$$f(\bar{x}) = f(\hat{x}) = f(\tilde{x}). \tag{2.11}$$

Thus, we can try the following function

$$f(x) = x^3(x-2)(x+1)(x-1) \tag{2.12}$$

which satisfies (2.10)-(2.11).

3. The Example

We are trying to see whether the function $f(x)$ defined in (2.12) is what we need. If cycle happens near the three point 0, -1 and 2, we would have

$$x_{3k} = \epsilon_k, \quad x_{3k+1} = -1 - \bar{\epsilon}_k, \quad x_{3k+2} = 2 + \hat{\epsilon}_k, \tag{3.1}$$

where ϵ_k, $\bar{\epsilon}_k$ and $\hat{\epsilon}_k$ are positive sequences that converge to zero. (3.1) and (2.12) give that

$$f(x_{3k}) \approx 2\epsilon_k^3, \quad f(x_{3k+1}) \approx 6\bar{\epsilon}_k, \quad f(x_{3k+2}) \approx 24\hat{\epsilon}_k. \tag{3.2}$$

If we set

$$\bar{\epsilon}_k = \epsilon_k^3/6, \quad \hat{\epsilon}_k = \epsilon_k^3/48 \tag{3.3}$$

and

$$\epsilon_{k+1} = \epsilon_k/2\ . \tag{3.4}$$

Then we have

$$f(x_{k+1}) \approx f(x_k)/2 \tag{3.5}$$

for all k. In order to satisfy (2.1), we require

$$\Delta_{3k} = 1 + \epsilon_k^3/6 \tag{3.6}$$
$$\Delta_{3k+1} = 3 + 9\epsilon_k^3/48 \tag{3.7}$$
$$\Delta_{3k+2} = 2 - \epsilon_k + \epsilon_k^3/48. \tag{3.8}$$

Due to (3.5), we can easily see that $Ared_k > 0$ for all k. Therefore we can have $x_{k+1} = x_k + s_k$ for all k. Because the points $\bar{x} = -1$ and $\hat{x} = 2$ are non-stationary points, the predicted reductions $Pred_{3k+1}$ and $Pred_{3k+2}$ will be bounded away from zero provided the approximate Hessian B_k are uniformly bounded (which is normally assumed). Therefore we have

$$\lim_{k\to\infty} r_{3k+1} = 0, \quad \lim_{k\to\infty} r_{3k+2} = 0 \tag{3.9}$$

Thus, if 2/3 is in the interval (τ_3, τ_4), Δ_{3k+2} can be set to the value of (3.8) if Δ_{3k+1} has the value of (3.7) when ϵ_k is very small. Similarly if 1/2 is in the interval (τ_3, τ_4), Δ_{3k+3} can be set to the value of (3.6) (with k substituted by $k+1$), if Δ_{3k+1} has the value of (3.8). In order to set Δ_{3k+1} by (3.7) when Δ_{3k} is defined by (3.6), we require that

$$\tau_1 > 3 \tag{3.10}$$

and that

$$r_{3k} > \tau_2. \tag{3.11}$$

(3.10) can usually be satisfied as many algorithms allow to increase trust region radius up to four times at good iterations. Condition (3.11) requires that

$$Pred_{3k} < \frac{1}{\tau_2} Ared_{3k} \approx \frac{1}{2\tau_2} f(x_{3k}) \approx \frac{1}{\tau_2}\epsilon_k^3. \tag{3.12}$$

However, if B_{3k} is positive semi-definite, we have that

$$Pred_{3k} \geq f'(x_{3k}) \approx 6x_{3k}^2 = 6\epsilon_k^2. \tag{3.13}$$

We can easily see that either (3.12) or (3.13) is not true when k is large. This shows that the function $f(x)$ defined at the end of the last section has to be modified.

The only problem with the function $f(x)$ defined by (2.12) is that r_{3k} would converges to zero. We will let

$$f(x) = \eta(x)(x-2)(x+1)(x-1) \tag{3.14}$$

which is obtained from function (2.12) by replacing the term x^3 by $\eta(x)$. Our modification is to reduce $f'(x_{3k})$ without reducing $f(x_{3k})$ so that (3.11) will be satisfied. In order to make our above analyses for (2.12) hold, we need to have

$$\eta(x_{3k}) = x_{3k}^3 + o(x_{3k}^3) \tag{3.15}$$

and

$$\eta(x_{3k+1}) = x_{3k+1}^3, \quad \eta(x_{3k+2}) = x_{3k+2}^3. \tag{3.16}$$

By setting $B_k = 0$, we have

$$Pred_{3k} \approx f'(x_{3k}) \approx 2\eta'(x_{3k}). \tag{3.17}$$

Since we also have

$$Ared_{3k} \approx \frac{1}{2} f(x_{3k}) \approx \eta(x_{3k}), \tag{3.18}$$

we require that

$$\eta(x_{3k}) \geq 2\tau_2\eta'(x_{3k}) \tag{3.19}$$

for all k where

$$x_{3k} = \frac{1}{2^k} x_0, \quad k = 1, 2, ..., \tag{3.20}$$

where $x_0 \in (0, 1)$. Define the function $\phi(x)$ by the

$$\phi(x) = x^3/4 + (6x^2 - x^3/4) sin^2(\pi x / x_{3k+3}) \quad x \in [x_{3k+3}, x_{3k}] \tag{3.21}$$

for all $k = 1, 2, ...,$ we can show that

$$\eta(x) = \int_0^x \phi(y) dy, \quad x \in (0, x_0), \tag{3.22}$$

satisfies (3.19) for all k, because we have $\tau_2 \in (0, 1)$ and

$$\eta(x_{3k}) \;=\; x_{3k}^3 + \frac{1}{32} x_{3k}^4 \tag{3.23}$$

$$\eta'(x_{3k}) \;=\; \frac{1}{4} x_{3k}^3 \tag{3.24}$$

for all k. Therefore we have made $f(x)$ satisfying the conditions required at the cycle point 0. Let $\eta(x) = x^3$ for all $x \in (-\infty, 0)$ and $x \in (2, \infty)$, $\eta(x)$ be defined by (3.22) for $x \in (0, x_0)$, and let $\eta(x)$ be a spline function in $(x_0, 2)$ so that it is three times continuous differetiable. Substitute this $\eta(x)$ into (3.14), we see that the conditions (3.16) hold. Therefore the cycle conditions for the points -1 and 2 are satisfied. Thus we have shown that there exists a required function $f(x)$ which yields (3.1) – (3.4). Our example show that for some trust region algorithms only the weekly convergence result holds but the iterate points cycle instead of convergence and furthermore only one cycle point is a stationary point.

References

[1] I.S. Duff, J. Nocedal, and J.K. Reid, "The use of linear programming for the solution of sparse sets of nonlinear equations", *SIAM J. Sci. Stat. Comput.* 8(1987) 99-108.

[2] R. Fletcher, "A model algorithm for composite NDO problem", *Math. Prog. Study* 17(1982) 67-76. (1982a)

[3] R. Fletcher, *Practical Methods of Optimization* (second edition) (John Wiley and Sons, Chichester, 1987)

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

[5] M.J.D. Powell, "Convergence properties of a class of minimization algorithms", in: O.L. Mangasarian, R.R. Meyer and S.M. Robinson, eds., *Nonlinear Programming 2* (Acadmic Press, New York, 1975) pp. 1-27.

[6] M.J.D. Powell, "Nonconvex minimization calculations and the conjugate gradient method", in: D.F. Griffiths, ed., *Numerical Analysis* Lecture Notes in Mathematics 1066 (Springer-Verlag, Berlin, 1984) pp. 122-141.

[7] D.C. Sorensen, "Newton's method with a model trust region modification", *SIAM J. Numer. Anal.* 20(1982) 409-426.

[8] Y. Yuan, *Trust Region Algorithms*, (unpublished manuscript, 1993).