Journal of Computational Mathematics, Vol.14, No.3, 1996, 195–202.

A RESTRICTED TRUST REGION METHOD WITH SUPERMEMORY FOR UNCONSTRAINED OPTIMIZATION*1)

L.P. Sun
(*Department of Mathematics, Nanjing University, Jiangsu, China*)

Abstract

A new method for unconstrained optimization problems is presented. It belongs to the class of trust region method, in which the descent direction is sought by using the trust region steps within the restricted subspace. Because this subspace can be specified to include information about previous steps, the method is also related to a supermemory descent method without performing multiple dimensional searches. Trust region methods have attractive global convergence property. Supermemory information has good scale independence property. Since the method possesses the characteristics of both the trust region methods and the supermemory descent methods, it is endowed with rapidly convergence. Numerical tests illustrate this point.

1. Introduction

In unconstrained optimization the basic problem considered is

$$\text{Min } f(x) \tag{1.1}$$

where $f(x) : R^n \to R$ is a real differentiable function. Many algorithms have been proposed for solving (1.1). The supermemory descent method is one of them. Its main idea is to combine a descent direction with the displacements generated by previous iterations for obtaining a new search direction. the typical form of the method is shown by Wolfe and Viazminsky[14]. That is, for the kth iteration, calculate α_k, $\beta_k^{(i)}$, s_k and x_{k+1} from

$$f\left(x_k + \alpha_k p_k + \sum_{i=1}^{m} \beta_k^{(i)} s_{k-1}\right) = \min_{\alpha,\beta^{(i)}} f\left(x_k + \alpha_k p_k + \sum_{i=1}^{m} \beta_k^{(i)} s_{k-1}\right),$$

$$s_k = \alpha_k p_k + \sum_{i=1}^{m} \beta_k^{(i)} s_{k-1}, \tag{1.2}$$

and

$$x_{k+1} = x_k + s_k.$$

* Received January 14, 1991.
1) The Project Supported by National Natural Sciences Foundation of China.

where p_k is a basic search direction and m is the number of memory terms. For a quadratic function with positive definite Hessian matrix, the iteration (1.2) with exact line search has the finite step termination property. Choosing different p_k, we obtain different supermemory descent algorithm: supermemory gradient methods, supermemory quasi-Newton methods, etc. Numerical experience show that it is more rapidly convergent than quasi-Newton methods, in general. the major weakness in this class of methods is the computational labour required to perform the (m+1)-dimentional search at each iteration. In order to overcome this defect, Sun[13] constructed a kind of supermemory descent algorithm that does not require the multiple dimentional search. But the method requires that the objective function possesses fairly strong quadratic properties in the neighbourhood of the iterative points to ensure convergence.

On the other hand, trust region methods is an effective way to overcome the difficulty caused by non-positive definite Hessian matrices in Newton's method. The basic idea is that the step is restricted by region of validity of the Taylor series. Given $x_k \in R^n$, consider the subproblem

$$\begin{cases} \text{Minimize} \quad \varphi_k(s) = f_k + g_k^T s + \frac{1}{2} s^T B_k s \\ \text{Subject to} \quad \| s \|_2 \leq \Delta_k \end{cases} \tag{1.3}$$

where B_k is an approximation to the Hessian matrix $\bigtriangledown^2 f(x)$ at x_k and Δ_k is the trust radius. The iteration consists of solving (1.3), and then comparing the actual reduction of the objective function

$$\text{ared}_k = f_k - f(x_k + s_k) \tag{1.4}$$

to the reduction predicted by the quadratic model

$$\text{pred}_k = f_k - \varphi(s_k). \tag{1.5}$$

If the reduction is satisfactory, then the step can be taken and a large trust region tried. Otherwise the trust region is reduced and the minor iteration is repeated.

The motivation for the idea of this paper is to find a means whereby the potential of a good quasi-Newton algorithm is exploited. The scheme suggested is one in which the descent step is sought by using trust region steps within restricted subspace. Because each subspace can be specified to include information about previous steps, the method is also related to a supermemory descent method but avoids the need for performing multiple dimensional searches. Information of this kind may be useful in providing local geometry information. Trust region methods have attractive global convergence property. Supermemory information has good scale independence property. Since the method possesses the characteristics of both the trust region methods and the supermemory descent methods, it is endowed with rapidly convergence. In Section 2 we specify the restricted trust region method. In Section 3 we discuss a rule for constructing the subspace. In Section 4 the convergence of the method is proved. In Section 5 numerical results are presented.

In this paper, the following notations are used: x^* denotes a solution of the problem (1.1). g_k is the gradient of f(x) at x_k. I denotes an unit matrix.

2. Trust Region Methods on a Subspace

In the standard trust region method, if the trust region steps are restricted within a sequence of subapaces, the kth step is generated by solving the problem

$$\begin{cases} \text{Minimize} & \varphi_k(s) = f_k + g_k^T s + \dfrac{1}{2} s^T B_k s \\ \text{Subjectto} & s \in S_k \\ & \| s \|_2 \leq \Delta_k. \end{cases} \tag{2.1}$$

Assume that Z_k is a $n \times m_0$ matrix such that $Z_k^T Z_k = I$ and that the columns of Z_k span S_k. Then the subspace constraint can be satisfied by setting $s_k = Z_k s_z$. Substituting this in (2.1) gives the problem

$$\begin{cases} \text{Minimize} & \psi_k(s_z) = f_k + g_z^T s_z + \dfrac{1}{2} s_z^T B_z s_z \\ \text{Subject to} & \| s_z \|_2 \leq \Delta_k. \end{cases} \tag{2.2}$$

where $g_z = Z_k^T g_k$, $B_z = Z_k^T B_k Z_k$ and $\| Z_k s_z \|_2 = \| s_z \|_2$. If $m_0 \ll n$, the subproblem (2.2) is a lower-dimensional version of the general trust region model (1.3). Obviously, the trust region step can be obtained by solving (2.2) since $s_k = Z_k s_z$.

A trust region algorithm with restricted subspace is given below.

Algorithm 2.1.

Step 0 Let k be specified. Given $\Delta_k > 0$, $x_k \in R^n$ and a symmetric positive definite matrix B_k.

Step 1 Calculate f_k and g_k. If the condition for termination is achieved, then stop.

Step 2 Update B_k by using a formula satisfying the quasi-Newton condition.

Step 3 Construct the matrix Z_k such that $Z_k^T Z_k = I$.

Step 4 Calculate $g_z = Z_k^T g_k$, $B_z = Z_k^T B_k Z_k$.

Step 5 If $\| B_z^{-1} g_z \|_2 \leq \Delta_k$, set $s_z = -B_z^{-1} g_z$ and go to Step 7.

Step 6 Solve the subproblem (2.2) and obtain s_z.

Step 7 Calculate $s_k = Z_k s_z$, $f(x_k + s_k)$ and $\tau_c = \frac{ared_k}{pred_k}$.

Step 8 Set $\Delta_{k+1} = \frac{1}{4} \| s \|_2$ if $\tau_c < 0.25$; set $\Delta_{k+1} = 2\Delta_k$ if $\tau_c > 0.75$ and $\| s \|_2 = \Delta_k$; otherwise set $\Delta_{k+1} = \Delta_k$.

Step 9 If $\tau_c \leq 0$, set $x_{k+1} = x_k$; else $x_{k+1} = x_k + s_k$.

Step 10 Set $k = k + 1$ and go to Step 1.

3. Choice of Subspace

The first reported use of the subproblem (2.1) appears to be due to Bulteau and

Vial[1] who proposed a restricted trust region algorithm by constructing S_k using the steepest descent direction and the quasi-Newton direction. On the other hand, Cullum and Brayton[2] point out that an algorithm has the quadratic termination property if, at each iteration, an exact line search is done and the direction of search is

$$d_k = \mu H_k g_k + \sum \beta_j s_j$$

where μ and β_j are suitable constants. Thus it seems that the subspace should be spanned by some basic descent direction and some linearly independent displacements of x_k to achieve fast asymptotic convergence. One of the basic descent direction is the steepest descent direction $-g_k$. The other usually depends on the positive definiteness of B_k. Here B_k is constructed by the update OCSSR1 (Osborne and Sun[7]). That is,

$$B_{k+1} = \omega_k B_k + \frac{(y_k - \omega_k B_k s_k)(y_k - \omega_k B_k s_k)^T}{(y_k - \omega_k B_k s_k)^T s_k} \tag{3.1}$$

where ω_k is a scaling factor, and it can be chosen automatically by satisfying Davidon's[3] criterion for an optimally conditioned Hessian estimate. Since B_k is always positive definite, the direction

$$d_k = -B_k^{-1} g_k \tag{3.2}$$

is taken as a basic descent direction. A rule to compute the matrix Z is given below.

Algorithm 3.1. (an additional condition on Step 3 of Algorithm 2.1)

Step 3.1 Calculate the descent direction d_k by (3.2).

Step 3.2 Select the s_{j1}, s_{j2},...,$s_{j(m_0-2)}$ from s_{k-1}, s_{k-2},... so that -g_k, d_k, s_{j1}, s_{j2},...,$s_{j(m_0-2)}$ are linearly independent.

Step 3.3 Using -$g_k, d_k, s_{j1}, s_{j2}, ..., s_{j(m_0-2)}$ constructs m_0 column vectors of Z_k by the Gram-Schmidt orthogonalization procedure.

Remark 3.2. If -g_k and d_k are linearly dependent, find the vectors s_{j1}, s_{j2},..., $s_{j(m_0-1)}$ such that $-g_k, s_{j1}, s_{j2}, ..., s_{j(m_0-1)}$ are linearly independent. Using them constructs Z_k by the Gram-Schmidt orthogonalization procedure.

Remark 3.3. If every s_j, $1 \le j \le k-1$, is linearly dependent to the basic descent direction -g_k and d_k, then $m_0 = 2$ and Z_k is constructed by the Gram-Schmidt orthogonalization procedure of -g_k and d_k.

4. Convergence Analysis

The global convergence of the algorithm 2.1 is straightforward, as it can be derived from Powell's results[8,9].

Because $Z_k Z_k^T$ is the orthogonal projection from R^n *to* S_k and $g_k \in S_k$, it is obvious that $Z_k Z_k^T g_k = g_k$ which, in turn, implies that $\| g_z \|_2 = \| g_k \|_2$. From Powell [8], we have that

$$\begin{aligned} f_k - \psi_k(s_k) &\ge f_k - \min_{s \in \text{span} g_k, \|s\|_2 \le \Delta_k} \psi_k(s) \\ &\ge 0.5 \| g_k \|_2 \min\{\Delta_k, \| g_k \|_2 / \| B_k \|_2\}. \end{aligned} \tag{4.1}$$

The global convergence of the algorithm 2.1 follows from the above inequality as long as $\| B_k \|_2$ increases not faster than linearly (Powell [9]). Therefore, we can establish directly the following result:

Theorem 4.1. *Assume that $f : R^n \to R$ is bounded below, and that $\nabla f(x)$ is uniformly continuous. Let $\{x_k\}$ be the sequence produced by the algorithm* 2.1. *If* (4.1) *and the bounds $\| B_k \|_2 \leq c_1 + c_2 k$ hold for all k, where c_1 and c_2 are constants, and if none of the gradients $g_k (k = 1, 2, 3...)$ is zero, then*

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

is obtained.

Osborne and Sun[7] prove the convergence of the matrices generates by the OCSSR1 update under some assumed conditions. That is

$$\lim_{k\to\infty} \| B_k - \nabla^2 f(x^*) \|_2 = 0.$$

Thus from Powell [8], we obtain directly the following conclusion:

Theorem 4.2. *Assume that $f(x)$ is twice continuously differentiable, and that $\nabla^2 f(x)$ is bounded and Lipschitz continuous. Let the sequence $\{x_k\}$ generated by the algorithm* 2.1 *with the* OCSSR1 *update converges to x^*. If in every iteration,*

$$| (y_k - \omega_k B_k s_k)^T s_k | \geq c \| y_k - \omega_k B_k s_k \|_2 \| s_k \|_2,$$

where $c \in (0, 1)$, if the sequence s_k is uniformly linearly independent, if the limit of the sequence ω_k is one, and if $\nabla^2 f(x^*)$ is positive definite, then the algorithm 2.1 *with the* OCSSR1 *update causes the sequence x_k to converge superlinearly.*

5. Numerical Results

The algorithm 2.1 was implemented with the OCSSR1 update. The resulting method is denoted by TR-OCSSR1. It is compared with our implementations of the following algorithm: DM-DOGLEG (Dennis-Mei's double dogleg method[4]) and BV-RTR (Bulteau-Vial's redtricted trust region method[1]). The number of terms with memory is decided by following creterion:

$$m = \begin{cases} 3, & 2 \leq n \leq 10, \\ 4, & n > 10. \end{cases}$$

The number of terms is not too critical, but there is some advantage in increasing it as the dimension of the problem increases.

The algorithm described in More and Sorensen[6] was used to solve the subproblem (2.2).

The test function are outlined as follows:

TF.1 Brown Badly Scaled x_0=(1,1)

TF.2 Beale x_0=(1,1)
TF.3 Biggs x_0=(1,2,1,1)
TF.4 Dixon x_0=(-2,...,-2)
TF.5 Hilbert (n=4) x_0=(-4,-2,-1.333,-1)
Hilbert (n=6) x_0=(-4,-2,-1.333,-1,-0.8,-0.6667)
TF.6 Miele x_0=(1,2,2,2)
TF.7 Extended Powell x_0=(3,-1,0,1,...,3,-1,0,1)
TF.8 Power x_0=(1,1,1,1)
TF.9 Extended Rosenbrock x_0=(-1.2,1,...,-1.2,1)
TF.10 Trigonometric $x_0=(\frac{1}{n},...,\frac{1}{n})$
TF.11 Wood x_0=(-3,-1,-3,-1)
TF.12 Nondia x_0=(-1,...,-1)

where TF.1, TF.2, TF.7, TF.9, TF.10 and TF.11 appear in More, Garbow and Hillstrom [5]; TF.3, TF.4 and TF.6 appear in Wolfe-Viazminsky [14]; TF.5 appears in Schittkowski [11]; TF.8 appears in Spedicato [12]; TF.12 appears in Shanno [10].

Table 5.1.
Numerical Results for TR-OCSSR1
(x_0 is standard initial point and $||g||_2 < 10^{-8}$)

Test Function	N	CPU	N_t	N_f	N_g	f^*	$\|\|g\|\|_2$
TF.1	2	0.17″	15	20	16	0.11x10^{-28}	0.66x10^{-8}
TF.2	2	0.28″	18	22	19	0.38x10^{-25}	0.16x10^{-11}
TF.3	4	0.55″	26	32	27	0.83x10^{-17}	0.54x10^{-8}
TF.4	10	1.10″	37	41	38	0.33x10^{-18}	0.72x10^{-9}
TF.5	4	0.11″	7	10	8	0.17x10^{-29}	0.14x10^{-15}
	6	0.16″	6	9	7	0.41x10^{-12}	0.46x10^{-8}
TF.6	4	0.88″	64	77	65	0.56x10^{-11}	0.53x10^{-8}
TF.7	4	0.77″	50	57	51	0.94x10^{-17}	0.33x10^{-8}
	16	3.19″	50	57	51	0.38x10^{-16}	0.65x10^{-8}
	64	77.22″	51	58	52	0.52x10^{-16}	0.63x10^{-8}
TF.8	20	2.20″	23	27	24	0.26x10^{-36}	0.36x10^{-17}
	50	34.82″	44	49	45	0.20x10^{-18}	0.40x10^{-8}
TF.9	2	0.33″	23	33	24	0.49x10^{-25}	0.89x10^{-11}
	50	18.84″	23	33	24	0.13x10^{-23}	0.46x10^{-10}
	100	116.33″	23	33	24	0.36x10^{-23}	0.79x10^{-10}
TF.10	5	0.60″	28	38	29	0.21x10^{-17}	0.15x10^{-8}
	10	2.36″	50	63	51	0.77x10^{-15}	0.95x10^{-9}
TF.11	4	0.71″	39	59	40	0.25x10^{-22}	0.17x10^{-9}
TF.12	20	5.11″	49	69	50	0.41x10^{-28}	0.31x10^{-12}
	50	46.85″	57	73	58	0.10x10^{-20}	0.13x10^{-8}
	100	288.30″	57	76	58	0.70x10^{-25}	0.11x10^{-10}

Tests were carried out in double precision on an IBM PC/AT clone. The corresponding machinc precision is of the order of 10^{-16}.

Table 5.2.
(x_0 is standard initial point and $||g||_2 < 10^{-8}$)

Test Function	N	Algorithm								
		TR-ocssr 1			DM - dogleg			BV - rtr		
		CPU	N_t	N_f	CPU	N_t	N_f	CPU	N_t	N_f
TF.1	2	$0.17''$	15	20	$0.77''$	42	77	$0.22''$	16	20
TF.2	2	$0.28''$	18	22	$0.16''$	14	18	$0.16''$	15	20
TF.3	4	$0.55''$	26	32	$1.26''$	79	83	$0.66''$	38	43
TF.4	10	$1.10''$	37	41	$0.71''$	27	36	$0.71''$	27	35
TF.5	4	$0.11''$	7	10	$0.17''$	9	12	$0.11''$	(12	
	6	$0.16''$	6	9	$0.22''$	13	16	$0.22''$	13	15
TF.6	4	$0.88''$	64	73	$0.88''$	61	84	$1.54''$	99	132
TF.7	4	$0.77''$	50	57	$0.60''$	47	57	$0.60''$	44	52
	16	$3.19''$	50	57	$5.72''$	135	151	$7.75''$	167	212
	64	$77.22''$	51	58	—	> 200	—	—	> 200	—
TF.8	20	$2.20''$	23	27	$3.02''$	33	47	$1.76''$	27	33
	50	$34.82''$	44	49	—	> 200	—	$28.12''$	60	68
TF.9	2	$0.33''$	23	33	$0.27''$	21	29	$0.33''$	19	28
	50	$18.84''$	23	33	—	> 200	—	$90.08''$	192	217
	100	$116.33''$	23	33	—	> 200	—	—	> 200	—
TF.10	5	$0.60''$	28	38	$0.44''$	23	28	$0.39''$	22	24
	10	$2.36''$	50	63	$0.88''$	26	28	$0.99''$	26	28
TF.11	4	$0.71''$	39	59	$0.83''$	52	72	$0.83''$	55	73
TF.12	20	$5.11''$	49	69	$13.40''$	154	195	$5.94''$	90	109
	50	$46.85''$	57	73	—	> 200	—	—	> 200	—
	100	$288.30''$	57	76	—	> 200	—	—	> 200	—

The results of the numerical experiments are summarised in Tables 5.1 and 5.2. Table 5.1 gives results obtained by the TR-OCSSR1 algorithm on some classical test functions for a range of different dimensions of the parameter vector. Table 5.2 gives the comparisons between TR-OCSSR1, DM-DOGLEG and DV-RTR. The basic data reported for each method are the dimension of the objective function algument (n), the CPU time (CPU), the number of iteration (N_t), the number of function evalutions (N_f), the number of gradient evaluations (N_g). If $N_t > 200$ the method is regarded as having failed. The convergence criterion is

$$\| g \|_2 < 10^{-8}.$$

Numerical tests show that the TR-OCSSR1 algorithm is very efficient and that it is suitable for medium-sized unconstrained optimization problems in comparison with other similar methods[1,4].

Acknowledgments. The author would like to thank Professor Y. Yuan for his valuable comments and suggestions.

References

[1] J.P. Bulteau and J.P. Vial, A redtricted trust region algorithm for unconstrained optimization, *JOTA*, 47 (1985), 413–435.

[2] J. Cullum and R.K. Brayton, Some remarks on the symmetric rank-one update, *JOTA*, 29 (1979), 493–520.

[3] W.C. Davidon, Optimally conditioned optimization algorithms without line searches, *Math. Prog.*, 9 (1975), 1–30.

[4] J.E. Dennis and H.H.W. Mei, Two new unconstrained optimization algorithms which use function and gradient values, *JOTA*, 28 (1979), 453–473.

[5] J.J. More, B.S. Garbow and K.E. Hillstrom, Testing unconstrained optimization software, *ACM Transaction on Mathematical Software*, 7 (1981), 17–41.

[6] J.J. More and D.C. Sorensen, Computing a trust region step, *SIAM J. Sci. Stat. Comput.*, 4 (1983), 553–572.

[7] M.R. Osborne and L.P. Sun, A new approach to the symmetric rank one updating algorithm, Research Report NMO/02, Stat.Res.Sec., SMS, ANU, 1988.

[8] M.J.D. Powell, Convergence properties of a class of minimization algorithms, *Nonlinear Programming 2*, (1975), 1–27.

[9] M.J.D. Powell, On the global convergence of trust region algorithms for unconstrained minimization, *Math. Prog.*, 29 (1984), 297–303.

[10] D.F. Shanno, Conjugate gradient methods with inexact searches, *Math. Oper. Res.*, 3 (1978), 244–256.

[11] K. Schittkowski, More test examples for nonlinear programming codes, t Lecture Notes Economics and Mathematical Systems 282, Springer-Verlag, (1987) .

[12] E. Spedicato, Computational experience with quasi-Newton algorithms for minimization problems of moderataly large size, Report CISE-N-175, CISE Documentation Serrice, Segrato, (1975).

[13] L.P. Sun, Adeptive supermemory descent method for unconstrained optimization, *Numer. Math. J. Chinese Univ.*, 4 (1982), 107–114.

[14] M.A. Wolfe and C. Viazminsky, Supermemory descent methods for unconstrained minimization, *JOTA*, 18 (1976), 455–469.