Bull. Nov. Comp. Center, Comp.Science, 14 (2001), 19–27

Minimization of nonlinear functions with linear restrictions*

E.A. Kotel'nikov, G.I. Zabinyako

The algorithms for minimization of large-dimension functions with linear constraints with allowance for sparseness of the limitation matrices are considered. A special case of the quadratic programming is emphasized. The algorithms have been implemented as a package of software programs to be used in the operational environments DOS, UNIX.

1. Algorithms

The following problem is considered:

$$\min f(x) \tag{1}$$

under the conditions

$$Ax = b, \tag{2}$$

$$\alpha \le x \le \beta \tag{3}$$

with the initial point x^0. Here f is a function (not necessarily smooth), the vectors α, β, x, $x^0 \in R^n$, $b \in R^m$, A is $m \times n$ matrix. The initial point x^0 may not satisfy limitations (2), (3). In this case, we find an admissible point, closest, in a sense, to x^0. If f is nonconvex, the approximations to a conditional stationary point are constructed.

For the solution of problem (1)–(3), it is proposed to use the reduced gradient method, which can be represented as a combination of elements of the modified simplex-method and one of the algorithms of unconditional minimization [1].

In connection with the fact that the object function is nonlinear, the current point of the optimizing sequence x^k is not necessarily located at the vertex of the polynomial specified by restrictions, therefore the number of components of the point x^k having the values $\alpha_i < x_i^k < \beta_i$ can exceed m. Then, in addition to the basis and the non-basis variables used in the simplex-method, we will consider the so-called superbasis variables. Let x_B, x_S, x_N be vectors of the basis, the superbasis and the non-basis variables,

*Supported by the Russian Foundation for Basic Research under Grants 99-07-90422, 01-07-90367.

respectively, and B, S, N be matrices made up of columns of the matrix A according to partitioning of components of the point x, B is a non-degenerate $m \times m$ matrix. Then

$$x_B = B^{-1}(b - Sx_S - Nx_N). \tag{4}$$

If on a certain segment of the computational process the mesh size along the direction of the shift p^k up to the minimum point of the function f is smaller than the mesh size up to the parallelepiped boundary, given by restrictions (3), the content of the basis, the superbasis and the non-basis variables does not change. When solving the subproblem involved, the non-basis variables preserve their values, and the basis and the superbasis variables take the values within their boundaries, in this case according to (4) the basis variables are dependent on the superbasis variables. Let

$$F(x_S) = f(x_B, x_S, x_N) = f(B^{-1}(b - Sx_S - Nx_N), x_S, x_N),$$

then the process of solving the subproblem reduces to minimization of $F(x_S)$ with simple restrictions $\alpha_B \le x_B \le \beta_B$, $\alpha_S \le x_S \le \beta_S$, where α_B, β_B, α_S, β_S are the vectors whose components are the values α_j, β_j corresponding to the basis and the superbasis variables. The gradient h of the function F is called the reduced gradient of the function f, and the matrix H of the second derivatives of the function F – the reduced matrix of the second derivatives of the function f. Denote $W = B^{-1}S$, where $V = [-W^T\ I\ 0]$, V is $s \times n$ matrix consisting of three submatrices: W^T corresponds to the basis variables, I is the unit matrix, and 0 is zero matrix corresponding to the superbasis and the non-basis variables, respectively. Then, according to [1] $h^k = Vg^k$ is the reduced gradient at the current point x^k, where $g^k = \nabla f(x^k)$; $H^k = VG^kV^T$ is the reduced matrix of the second derivatives of the function f; and G^k is the matrix of the second derivatives of the function f at the point x^k.

For the solution of a subproblem, three methods of unconditioned minimization are provided: the conjugate gradient or the quasi-Newton method can be used for a smooth object function, r-algorithm – for a non-smooth function. The mesh size along the direction p^k to the minimum point of the function f can be found with the help of the quadratic or the cubic interpolation; the so-called adaptive algorithm is provided for a non-smooth function.

A subproblem is considered to be solved if the value $\|h\|_\infty$ is small. The first direction p_S of any subproblem is a reduced antigradient, i.e., $p_S = -h$.

1.1. Conjugate gradients algorithm. Let an object function $f(x)$ be a twice differentiable function of the vector argument $x \in R^n$. Approximations to the minimum are constructed on the basis of the iterative process:

$$\begin{aligned} p^0 &= -\nabla f(x^0), \\ p^i &= -\nabla f(x^i) + \beta_i p^{i-1} + \gamma_i p^t, \\ x^{i+1} &= \arg\min_{\lambda>0} f(x^i + \lambda p^i), \end{aligned}$$

where $\nabla f(x^i)$ is the gradient f at the current point x^i, p^{i-1} and p^t are the directions used in previous iterations, β_i and γ_i are coefficients. In a standard scheme of the conjugate gradient method, the direction of an antigradient is used in restart iterations at $i = n, 2n, \ldots$. In [2], a linear conjugate gradients algorithm (for the quadratic functions f), is substantiated, in which the other vector is used for restart. For a general case, this algorithm is developed in [3]. When developing the program, we made use of the improved algorithm [4] with a non-standard restart.

A correct application of the non-standard renewal makes it possible to gain some progress when solving ill-conditioned problems by the conjugate gradients method.

1.2. Quasi-Newton algorithm. In the quasi-Newton algorithm, the recurrent k-th direction of the descent p^k is found from the system of equations

$$B^k p^k = -\nabla f(x^k),$$

where B^k is the current estimation of the second derivatives matrix of the function $f(x)$. The most efficient quasi-newtonian algorithms are obtained with the use of the estimations B^k of the BFGS formula for the recalculation:

$$B^{k+1} = B^k + \frac{1}{(\nabla f(x^k), p^k)} \nabla f(x^k) \nabla f^T(x^k) + \frac{1}{\lambda_k (Y^k, p^k)} Y^k (Y^k)^T,$$

where $Y^k = \nabla f(x^{k+1}) - \nabla f(x^k)$, and λ_k is a mesh size along the direction p^k.

In [5], there is proposed an efficient procedure of supporting in iterations the representation of the matrices B^k in the factorized form $B^k = L^k D^k (L^k)^T$, where L^k is the left triangular matrix with 1 on the main diagonal, D^k is the diagonal matrix. Application of such a factorized form allows ensuring the strict positive definiteness of the matrices B^k with allowance for the rounding off.

1.3. *r*-algorithm. To minimize non-differentiable functions, algorithms of the subgradient form with the space extended towards the difference of two subsequent subgradients (r-algorithms) proved to be most effective [6].

Let f be a function to be minimized, and $\partial f(x)$ be its subgradient at the point $x \in R^n$. At the beginning of the process we set the matrix B^0 equal to the unit $n \times n$ matrix I and the coefficient of extension $\alpha > 0$ (usually $2 \leq \alpha \leq 3$). The displacement from the initial point x^0 is done in

the direction opposite $\partial f(x^0)$. Further, at any k-th iteration of r-algorithm, the following values are sequentially determined:

$$Y^k = \partial f(x^k) - \partial f(x^{k-1}), \tag{5}$$

$$r^k = (B^k)^T Y^k, \tag{6}$$

$$\xi^k = \frac{r^k}{\|r^k\|_2}, \tag{7}$$

$$B^{k+1} = B^k(I + (\alpha^{-1} - 1)\xi^k(\xi^k)^T), \tag{8}$$

$$p^k = (B^{k+1})^T \partial f(x^k), \tag{9}$$

$$x^{k+1} = x^k - \lambda_k B^{k+1} \frac{p^k}{\|p^k\|}, \tag{10}$$

where $\lambda_k > 0$ is the displacement at the k-th iteration. It is evident from (5)–(10) that it appears possible to use the symmetric matrices $H^k = B^k(B^k)^T$ instead of the matrices B^k in the iterative process. In this case

$$H^{k+1} = H^k + (\beta^2 - 1)\frac{H^k Y^k (Y^k)^T H^k}{(H^k Y^k, Y^k)},$$

where $\beta = 1/\alpha$, and (10) will take the form

$$x^{k+1} = x^k - \lambda_k \frac{H^{k+1}\partial f(x^k)}{\sqrt{(H^{k+1}\partial f(x^k), \partial f(x^k))}}.$$

The use of the symmetric matrices H^k instead of B^k would allow us to considerably reduce the requirements of the algorithm for the memory volume. In addition, the matrix B^k with an increase of k tends to zero matrix. The direct calculation of $H^k = B^k(B^k)^T$ can result the fact that the matrix H^k will not be strictly positive definite.

In [7], based on the algorithm from [5], a reliable way of recalculation in iterations of the iterative matrices H^k in the factorized form is proposed. The proposed procedure, as in the quasi-Newton algorithm, provides the iterative transfer from the representation H^k in the form $L^k D^k (L^k)^T$ to $L^{k+1} D^{k+1} (L^{k+1})^T$ for H^{k+1}.

For the search for the displacement λ_k it is possible to use the algorithms based on the quadratic, the cubic interpolation, or a special adaptive algorithm similar to the algorithm from [6]. The first two are offered for smooth f, and the adaptive algorithm is applicable in the case of non-smooth f.

2. Solution of problems

To verify the reliability of software, a series of calculations have been carried out. As an object function, the well-known test problems of unconditional

optimization were selected. Matrices of constraints were selected from a test set of the linear programming problems NETLIB [8]. The problems were solved on the computer P-75.

Further, examples of minimization of the generalized Rosenbroc function for $n > 2$ are given:

$$f(x) = \sum_{i=2}^{n} (100(x_i - x_{i-1}^2)^2 + (1 - x_i)^2)$$

with the initial point $x^0 = (-1.2, 1, \ldots, 1)^T$. It is a known fact that the unconditional minimization of the Rosenbroc function results in $x^* = (\pm 1, 1, \ldots, 1)^T$ and $f^* = 0$. In each problem, the vector of the right parts of the constraint (2) is redetermined: for $i = 1, 2, \ldots, m$, $b_i = \sum\limits_{j=1}^{n} a_{ij} x_j^* + \delta_i$, where the values δ_i are taken at random. Depending on the sign δ_i, the type of a constraint is introduced. The values α_j and β_j from (3) are selected so that the conditions $\alpha_j \le x_j^* \le \beta_j$, $j = 1, 2, \ldots, n$, be fulfilled.

In Tables 1 and 2, the results of solution of these problems by various methods are presented. The following notations are used:

it – the total number of iterations;

NF – the number of calculations of a function;

NG – the number of calculations of a gradient;

$$\delta x = \max_i |x_i^* - 1.0|; \qquad \delta f = |f^*|;$$

$$h_r = \max_i \left| \sum_{j=1}^{n} a_{ij} x_j^* - b_i \right|; \qquad h_c = \max_{j \in I_B} \left| \sum_{i=1}^{m} a_{ij} y_i^* - c_j^* \right|,$$

where I_B is a list of basis variables, y_i^* are dual estimations, c_j^* is the j-th component of gradient of the function f at the point x^*, t is time in seconds.

Table 3 shows results of minimization of the non-smooth function from [9]

$$f(x) = \sum_{j=1}^{101} \left| \sum_{i=1}^{n} x_i t_j^{i-1} - \sum_{i=1}^{n} x_i^* t_i^{j-1} \right|,$$

$t_j = 0.01(j-1)$, $j = 1, 2, \ldots, 101$, $x^0 = (0, 0, \ldots, 0)^T$ with linear constraints. The minimum of this function is $f^* = 0$ at the point $x^T = (\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n})$. Here $\delta x = \max\limits_j |x_j^* - \frac{1}{n}|$.

Table 1. Conjugate gradients method

Name of problem	it	NF	NG	δx	δf	h_r	h_c	t
BRANDY	2478	7566	9840	2.E−8	5.E−13	1.E−11	2.E−20	24
CAPRI	5907	14941	20481	7.E−8	8.E−12	2.E−12	4.E−19	77
GROW7	381	749	1009	2.E−8	1.E−12	6.E−12	2.E−20	4
GROW15	1180	2867	3723	2.E−8	2.E−12	2.E−11	1.E−16	27
SCTAP1	12817	26659	39032	2.E−7	1.E−10	2.E−9	8.E−20	160
FINNIS	5685	16588	21633	3.E−8	4.E−12	6.E−9	2.E−12	121
GFRD-PNC	16663	35208	51085	4.E−6	6.E−8	5.E−8	2.E−17	403
E226	592	590	857	1.E−8	4.E−13	3.E−12	4.E−12	5
SCFXM1	3537	6749	9736	7.E−8	1.E−11	9.E−11	3.E−19	49
LOTFI	1544	3919	5121	8.E−9	3.E−9	5.E−10	3.E−10	14

Table 2. Quasi-Newton method

Name of problem	it	NF	NG	δx	δf	h_r	h_c	t
BRANDY	561	927	1284	1.E−8	2.E−12	1.E−11	9.E−20	15
CAPRI	1306	2228	3167	6.E−9	1.E−13	7.E−11	1.E−19	54
GROW7	370	646	895	9.E−9	5.E−15	5.E−12	1.E−20	21
GROW15	709	946	1331	5.E−9	3.E−13	2.E−11	2.E−17	156
SCTAP1	2211	3812	5438	3.E−7	1.E−10	4.E−9	3.E−20	202
FINNIS	1976	2980	4316	3.E−8	1.E−12	4.E−10	3.E−12	181
GFRD-PNC	2189	2954	4358	3.E−6	7.E−9	5.E−8	3.E−17	1226
E226	407	168	249	2.E−8	3.E−12	9.E−13	8.E−11	7
SCFXM1	1261	1623	2334	2.E−8	4.E−12	9.E−11	3.E−18	72
LOTFI	616	683	957	5.E−9	2.E−13	2.E−8	5.E−8	23

Table 3

Name of problem	it	NF	NG	δx	δf	t
SC105	204	593	146	0.027	4.7E−6	8
KB2	357	3117	322	2.3E−4	2.6E−6	9
RECIPE	440	2697	399	0.022	2.2E−4	33
SC50A	227	1489	190	4.8E−9	3.6E−8	5
SC50B	156	1112	126	6.7E−9	2.9E−8	4
SHARE2B	469	2323	382	2.4E−3	2.3E−5	13

3. Quadratic programming

The software for convex quadratic problems is collected in a separate package. This is connected with simple realization of the conjugate gradients method for such problems, the efficiency of computing process and the presence of reliable means of control over computational errors.

The problem of quadratic programming is in the minimization of the functions $f(x) = \frac{1}{2}x^T Qx + c^T x$, with constraints (2), (3), where Q is a symmetric positive definite matrix.

As known, the scheme of the linear conjugate gradients method is of the form:

$$\begin{aligned} p^0 &= -r^0, \\ p^k &= -r^k + \beta_k p^{k-1}, \quad k > 0, \\ x_{k+1} &= x_k + \alpha_k p^k, \\ r^{k+1} &= r^k + \alpha_k Q p^k, \end{aligned} \tag{11}$$

where $r^k = Qx_k + c$ is the gradient of the function at the current point x_k; the coefficients β_k are calculated by the formula providing conjugation of the vectors p^k: $\beta_k = \|r^k\|_2^2 / \|r^{k-1}\|_2$; α_k is the value of displacement along the direction p^k to the minimum point: $\alpha_k = \|r^k\|_2^2 / ((p^k)^T Q p^k)$.

With linear restrictions in scheme (11), the gradient r^k and the matrix Q are replaced by the reduced gradient and the reduced matrix, respectively, which brings about the change of calculations of α_k, β_k, and p^k.

In the space of superbasis variables, the direction of the descent $p_S^k = -h^k + \beta_k p_S^{k-1}$ is found, where $\beta_k = \|h^k\|_2^2 / \|h^{k-1}\|_2^2$. Then the direction p_B^k for the basis variables is calculated:

$$p_B^k = -B^{-1} S p_S^k = -W p_S^k.$$

Non-basis variables do not change their values, therefore the vector p_N^k is equal to zero. Consequently, the general direction of the displacement in the original space $(p^k)^T = ((p_B^k)^T, (p_S^k)^T, 0^T)$ will satisfy the condition $Ap^k = 0$.

Further the value of the displacement α_k along the direction p_S^k to the minimum point of the function F is found as follows:

$$\alpha_k = -\frac{(h^k)^T p_S^k}{(p_S^k)^T H p_S^k}.$$

For calculation of the quadratic form in the denominator, the matrix H is not needed, because of

$$(p_S^k)^T H p_S^k = (p_S^k)^T V Q V^T p_S^k = (p^k)^T Q p^k.$$

For estimation of the accuracy of solution of the subproblem, the criterion proposed in [10] is made use of. After any $\sqrt{s}$ iterations of the subproblem, the values $\varphi = \|\bar{r}^k - r^k\|^2$ and $\psi = \exp((\frac{t}{S})^2)$, are calculated, where $\bar{r}^k = Qx_k + c$, t is the number of the current iteration of the subproblem, k is the general number of iterations at a concrete moment. Then, if $\|r^k\|_2^2 \geq \varphi\psi$, it is considered that the process can be continued; otherwise, it is necessary to regenerate the conjugate gradients scheme.

For testing the software, matrices of constraints were selected from the set NETLIB, and the matrix Q was generated using various versions of distribution of the eigenvalues $d_i > 0$, $Q = T^T DT$, where T is an orthogonal matrix, D is a diagonal matrix with the entries d_i on the diagonal. The testing has shown a high reliability of the software.

4. Some information about the package

The software can be executed on the computers IBM PC in DOS-environment, Silicon Graphics and RM600 in media similar to UNIX; in the programming language FORTRAN-77.

The software, in addition to procedures of the numerical solution of problems, contains data processing programs, compilation of dictionaries and reference tables, translation from the external MPS-format [1] to the internal package format. For matrices, the sparse column format is used, which is determined by analogy with the line sparse format [11]. In the linear programming procedures, the matrices reverse to the basis matrices are presented in the multiplicative form. A special data structure is used for storing multiplicators [12].

In the sequel, it is assumed to supplement the software with algorithms of solution of nonlinear discrete problems and to develop their parallel versions for the computer RM-600.

References

[1] Murtaf B. Modern Linear Programming. Theory and Practice. – Moscow: Mir, 1984.

[2] Beale E.M.L. A derivation of conjugate gradients // Numerical methods for nonlinear optimization / Ed. F.A. Lootsma. – London; New York: Academic Press, 1972. – P. 39–43.

[3] Powell M.J.D. Restart procedures for the conjugate gradient method // Math. Prog. – 1977. – № 12. – P. 241–254.

[4] Zabinyako G.I. Restart procedures in the conjugate gradients method // Optimizatsia. – 1989. – № 46 (63). – P. 5–13.

[5] Gill P.E., Murray W. Quasi-Newton methods for unconstrained optimization // J. Inst. Maths. Applics. – 1972. – Vol. 9, № 1. – P. 91–108.

[6] Shor N.Z., Stetsenko S.I. Quadratic Extremal Problems and Non-differentiable Optimization. – Kiev: Naukova Dumka, 1989.

[7] Zabinyako G.I. On numerically stable method of recalculation of transformation matrices in r-minimization algorithms // Sistemnoe Modelirovanie. – Novosibirsk: VTs SO AN SSSR, 1990. – №. 16. – P. 11–15.

[8] Gay D.M. Electronic mail distribution of linear programming test problems // Mathematical Programming Society CO AL Newsletter. – 1985. – Vol. 13. – P. 10–12.

[9] Polyak B.T. Introduction into Optimization. – Moscow: Nauka, 1983.

[10] Wilkinson J., Reinsch K. Reference Book on Algorithms in ALGOL. Linear Algebra. – Moscow: Mashinostroenie, 1976.

[11] Pissanetsky S. Technology of Sparse Matrices. – Moscow: Mir, 1988.

[12] Zabinyako G.I., Kotel'nikov E.A. Programs of minimization of nonlinear functions with linear restrictions. – Novosibirsk, 1997. – (Report / RAS. Siberian Branch. ICMMG; GR №№ 01.9.30 001317, 02.9.70 004793).