Chebyshev methods for optimization of large systems on stiff target functionals with weakly filled structured hessian matrices

Igor G. Chernorutsky
Peter the Great St. Petersburg Polytechnic University
29, Polytechnicheskaya str., St.Petersburg,195251
igcher1946@mail.ru

Nikita V. Voinov
Peter the Great St. Petersburg Polytechnic University
29, Polytechnicheskaya str., St.Petersburg,195251
voinov@ics2.ecd.spbstu.ru

Anna V. Rubtsova
Peter the Great St. Petersburg Polytechnic University
29, Polytechnicheskaya str., St.Petersburg,195251
annarub2011@yandex.ru

Lina P. Kotlyarova
Peter the Great St. Petersburg Polytechnic University
29, Polytechnicheskaya str., St.Petersburg,195251
lina1305@yandex.ru

Olga V. Aleksandrova
Peter the Great St. Petersburg Polytechnic University
29, Polytechnicheskaya str., St.Petersburg,195251
o.v.alexandrov@yandex.ru

Abstract: We analyze matrix gradient methods for optimizing large systems of arbitrary nature according to functions with a special character of Hessian matrices. It is assumed that the Hessian matrix is sparse and structured (nonzero elements occupy fixed positions). The last condition is for large systems optimization consisting of interconnected subsystems of a lower dimension. We suggest that the use of standard Newton-type methods is difficult because of the possible sign uncertainty of the Hessian matrices (non-convexity) and the need to store full-size high-order matrices. In the procedures under consideration, memory savings are achieved by storing only sparse matrices in packaged form.

Keywords: gradient methods, relaxation functions, non-convex problems, stiff functionals.

1 Introduction

To solve the problem of unconditional minimization

$$J(x) \to \min_{x}, \; x \in R^n, \; J \in C^2\left(R^n\right)$$

a class of matrix gradient methods is considered,

$$x^{k+1} = x^k - H_k\left(G_k, h_k\right) J'\left(x^k\right), \quad h_k \in R^1, \tag{1}$$

where $G_k = J''\left(x^k\right), H_k$ – matrix function G_k .

This class of methods includes some particular cases: classical procedures as gradient descent methods, Levenberg-Marquardt methods, Newton's methods, methods with exponent relaxation (ER-methods).

We present approach based on the concept of the relaxation function [1] for constructing and analyzing nontraditional gradient methods with the Chebyshev relaxation function. The main assumptions used in the construction of this class of methods are:

- high dimension of the argument vector x (n>>1) and the undesirability of storing a full-size matrix (nxn) in the computer's memory;
- a high stiffness degree [1, 2] of the functional J(x) in a wide range of the argument variation;
- convexity of the minimized functional J(x) is guaranteed only in the neighborhood of the minimum point.

If the stiffness degree of the optimality criterion J(x) is sufficiently high, then standard computational tools are not very effective due to the reasons stated in [1, 2]. Newton-type methods, as it is known [3-5], are not intended for solving non-convex problems. Moreover, they lose their effectiveness in conditions of high stiffness. The use of algorithms from the class of well-known Markuardt-Levenberg methods with a high stiffness degree of the functional J(x) is associated with the complexity of the algorithmic selection of the corresponding regularization parameter.

Most often in this situation, it is recommended to use different non-matrix forms of the conjugate gradient method (SG). However, further it will be shown that in the class of matrix gradient schemes (1) there are algorithms that are more efficient for the considered problems than the SG methods.

2 Multiparameter Optimization Tasks

Large systems we define as systems described by models with a large number of controlled parameters.

An optimized system may be represented as a set of smaller interconnected subsystems (Fig. 1).

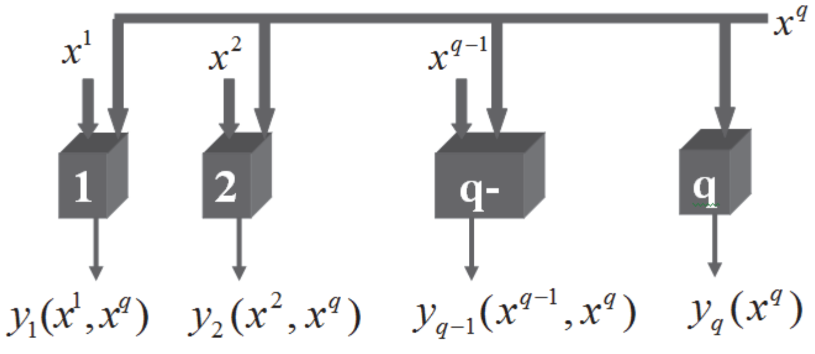

Figure 1 – The complex of interrelated subsystems

Y is the output characteristics of the subsystems.

The requirements for the output parameters of the system (specifications) are given in the form of a system of inequalities:

$$y_j(x_j, x^q) \le t_j, j \in [1: q-1]; y_q(x^q) \le t_q, \tag{2}$$

where j is n_j-dimensional local vector of controlled parameters; x^q – n_q-dimensional vector of controlled parameters, affecting all q output parameters and interconnecting individual subsystems of the system being optimized. The dimension of the full vector of controlled parameters $x = [x^1, x^2, \ldots, x^q]$ is equal to

$$n = \sum_{i=1}^{q} n_i. \tag{3}$$

Using the known technique of reducing the solution of inequality systems to optimization tasks, we obtain the target functional of the form

$$J\left(x\right) = \sum_{j=1}^{q} U_j\left(x^j, x^q\right) \to \min, x \in R^n. \tag{4}$$

If we have the system of inequalities $y_i(x) \le t_i$, it is equivalent to this system:

$$z_i(x) = t_i - y_i(x) \ge 0$$

Corresponding optimization task:

$$J_1(x) = \min_i z_i(x) \to \max_x$$

It is convex and can be reduced to an equivalent formulation that allows achieving non-convexity:

$$J_2(x) = \max_i(\exp(-z_i(x))) \to \min_x$$

The last task with a sufficiently large ν is equivalent to the following:

$$J_3(x) = \sum_{i=1}^{m} \exp(-\upsilon z_i(x)) \to \min_x$$

This is the main target functional in solving systems of inequalities. It is non-convex and has the form (4).

Functionals (4) also arise in other formulations of optimization tasks of real systems, therefore the problem (4) has a rather general character.

Further, we will consider methods for problem solving (4) with the following additional assumptions:

- solving analyzing task of the optimized system requires significant computational costs. Therefore, in the optimization process, it is required to minimize the number of references to the calculation of values $J(x)$;
- fill coefficient γ of matrix $G(x) = J''(x)$ is small. We can usually assume $\gamma \sim 1/q$.

It is easy to establish that the structure of the matrix G (x) does not depend on the point x:

$$G(x) = \begin{bmatrix} G_{11} & & 0 & & G_{1q} \\ & G_{22} & & & G_{2q} \\ 0 & & G_{33} & & G_{3q} \\ & & & \ddots & \vdots \\ G_{q1} & G_{q2} & G_{q3} & \dots & G_{qq} \end{bmatrix}.$$

The submatrices G_{ij} have dimensions $n_i \times n_j$, and the total number of nonzero elements is

$$\sum_{i=1}^{q} n_i^2 + 2n_q \sum_{i=1}^{q-1} n_i.$$

Thus, taking into account the symmetry of the matrix G (x), it is necessary to store in the computer's memory

$$\sum_{i=1}^{q} \left(n_i^2 + n_i\right)\Big/2 + n_q \sum_{i=1}^{q-1} n_i$$

nonzero elements. The necessary information about the storage schemes of sparse matrices are widely presented in the literature describing large data [6].

3 Methods with Chebyshev relaxation functions

We take the eigenvalues $\lambda_i(G_k) \in [-m, M]$, $M >> m > 0$. According to above assumptions and requirements for relaxation functions formulated in [1], the most rational method should have a relaxation function R (λ), the values of which sharply decrease from R = 1 at λ = 0, remaining small throughout the entire range [0, *M*]. And opposite, if λ<0, the function *R* (λ) should increase intensively. In addition, the matrix function *H* corresponding to R (λ) must be constructed without matrix multiplications in order to preserve the sparsity property of the matrix $G_k = J''(x^k)$.

Основной текст должен быть написан в одну колонку. Размер шрифта – 10, межстрочный интервал – одинарный. Рекомендуется использовать шрифт Times New Roman.

We show that as such R (λ) with accuracy up to multiplier, offset Chebyshev polynomials of the 2nd kind $P_s(\lambda)$ can be used, satisfying the known recurrence relations:

$$P_1(\lambda) = 1,\ P_2(\lambda) = 2(1 - 2\lambda);\ P_{s+1}(\lambda) = 2(1 - 2\lambda)P_s(\lambda) - P_{s-1}(\lambda). \quad (5)$$

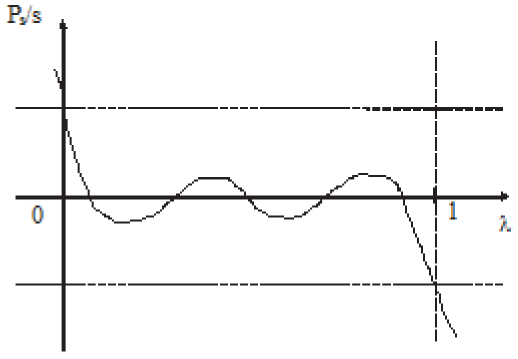

Figure 2 – Chebyshev relaxation function

Dependency graph $P_s(\lambda)/s$ for some s is shown in Fig. 2. Supposing $R(\lambda) = P_L(\lambda)/L$ with a sufficiently large value of L, we get an arbitrarily fast relaxation of any addend in the presentation of approximating paraboloid [1]

$$f\left(x^{k+1}\right) = \frac{1}{2}\sum_{i=1}^{n} \xi_{i,k}^{2}\lambda_i R^2\left(\lambda_i\right), \tag{6}$$

where

$$x^k = \sum_{i=1}^{n} \xi_{i,k} u^i$$

is a decomposition of the current vector into eigenvectors of the matrix G_k .

This statement follows from the known fact of uniform convergence of the sequence $\{P_s(\lambda)/s\}$ to zero when $s \to \infty$ in the open space (0, 1). Further, we assume that the G_k matrix eigenvalues are normalized to the interval (0, 1). In this case, it suffices to consider the matrix G / || G || instead of the matrix G, and the vector g / || G || instead of the gradient vector g.

Correlating to the adopted R (λ), H (λ) dependence has the form

$$H(\lambda) = [1 - R(\lambda)]/\lambda = [1 - P_L(\lambda)/L]/\lambda. \tag{7}$$

The methods design (1) directly with function (7) is possible, but it makes it necessary to solve at each step k large linear systems of equations with sparse matrix. Below it is shown that there are more effective implementation techniques.

According to (7) it follows that H (λ) is a polynomial of L–2 degree, while R (λ) has L–1 degree. Therefore, to implement the matrix gradient method with the indicated function H (λ), generally speaking, there is no need to solve linear systems. The method will be as follows.

$$x^{k+1} = x^k - \left(\alpha_1 E + \alpha_2 G_k + \ldots + \alpha_{L-1} G_k^{L-2}\right) g^k = x^k - H\left(G_k\right) g^k. \tag{8}$$

The implementation of method (8) can be based on the techniques of calculating the coefficients α_i for various L degrees. In this case, the number L should be chosen from the condition of the most rapid decrease of $J(x)$. An alternative, more economical approach based on other considerations is discussed below.

For function:

$$H_s\left(\lambda\right) \triangleq \alpha_1 + \alpha_2\lambda + \ldots + \alpha_{s-1}\lambda^{s-2},\ s = 2, 3, \ldots$$

from (5) we can get the recurrence relation

$$(s + 1)H_{s+1} = 2s(1 - 2\lambda)H_s - (s - 1)H_{s-1} + 4s;\ H_1 = 0,\ H_2 = 2,\ s \in [2: L - 1]. \tag{9}$$

Consequently, we have

$$x^{k+1}\left[s+1\right] \triangleq x^k - H_{s+1}g^k = x^k - \frac{2s}{s+1}\left(E - 2G_k\right)H_s g^k + \frac{s-1}{s+1}H_{s-1}g^k - \frac{4s}{s+1}g^k,$$
$$s \in \left[2 : L-1\right]$$

or

$$\vartheta_{s+1} \triangleq x^{k+1}[s+1] - x^k = \frac{2s}{s+1}(E - 2G_k)\vartheta_s - \frac{s-1}{s+1}\vartheta_{s-1} - \frac{4s}{s+1}g^k;$$

$$\vartheta_1 = 0,\ \vartheta_2 = -2g^k,\ s \in [2 : L-1]. \tag{10}$$

$x^{k+1}[s]$ is s-e approximation to vector $x^{k+1} = x^{k+1}[L]$.

Thus, with a fixed quadratic approximation *f(x)* of the functional *J(x)* in the neighborhood of $x = x^k$, we have the opportunity to move from P_s to P_{s+1} due to one multiplication of the matrix $E - 2G_k$ by the vector, fully using the sparsity property of the matrix G_k and without additional gradient calculations. The efficiency of algorithm (9) with large values of the stiffness coefficientη [1,2] is determined by the relaxation factors for small eigenvalus of the matrix G_k. Consider the positive part of the spectrum ($\lambda > 0$), which is especially important in the neighbourhood of the optimum, where the matrix $G(x)$ is positively defined. The main advantage of the method with $R_s(\lambda) = P_s(\lambda)/s$ is that already at small s there is a noticeable suppression of the addends from (5) in a wide range of λ values . Below there are the R_s values for the internal maximum of $R_s(\lambda)$ and the boundaries of the ranges $\alpha_s \le \lambda \le \beta_s$, where $|R_s(\lambda)| \le R_s$:

s	…	3	4	5	6	7	8
R_s	…	0,333	0,272	0,250	0,239	0,233	0,230
α_s	…	0,147	0,092	0,061	0,044	0,033	0,025
β_s	…	0,853	0,908	0,939	0,956	0,967	0,975
$-R'_s(0)$	…	5,30	10,0	16,0	23,3	32,0	42,0

In the left part of the spectrum ($\lambda < 0$) we have

$$R_s(\lambda) > 1 + R'_s(0)\lambda,$$

therefore, the values of the derivatives $R'_s(0)$ in the last row of the table characterize the relaxation multipliers for the negative terms in (6). Calculation of derivatives $R'_s(0)$ can be performed on the basis of the following recurrence relations:

$$P'_1 = 0,\ P'_2 = -4;\ P'_{s+1} = 2P'_s - 4s - P'_{s-1};\ R'_L(0) = P'_L/L.$$

α_s, β_s values for $s > 8$ ($\lambda > 0$) can be calculated using the asymptotic formula

$$\alpha_s = 1{,}63/s^2,\ \beta_s = 1 - \alpha_s; \tag{11}$$

when $R_s < 0{,}22$.

The relation (11) is obtained from the following representation of Chebyshev polynomials

$$P_L(\lambda) = \frac{\sin L\zeta}{L\sin\zeta},\ \lambda = \sin^2\frac{\zeta}{2},\ \lambda, \zeta \in [0,1].$$

Indeed, for sufficiently small ζ we have:

$$P_L(\lambda) \cong \Phi(\xi) = \frac{\sin\sqrt{\xi}}{\sqrt{\xi}},\ \xi \triangleq 4L^2\lambda.$$

If we consider $x = sqrt(\zeta)$, we get

$$F(\xi) = \varphi(x) = \sin x / x.$$

We have

$$F(\xi) \le \Phi/F(\xi_{\kappa p}) \text{ when } \xi \ge \xi_{\kappa p},$$

where

$$\xi_{\kappa p} = x^2_{\kappa p} = 6{,}523;\ \Phi(\xi_{\kappa p}) = \varphi(x_{\kappa p}) \cong 0{,}22.$$

Thus, if we suppose $\xi_{\kappa p} = 4L^2\lambda_{\kappa p}$, we get the following statement: for the smallest (positive) eigenvalue m the inequality is fulfilled

$$\xi_{min} = 4L^2 m \ge \xi_{\kappa p} = 6{,}523,$$

it means, if

$$m \ge 6{,}523/(4L^2) = 1{,}63/L^2, \tag{12}$$

for all $\lambda > m$ we'll have

$$|R_L(\lambda)| \le 0{,}22.$$

From (12) follows (11).

The enlarged scheme of the algorithm based on the relation (10) can be implemented using the following sequence of steps. It is assumed that all task variables are properly normalized. We also suppose that the variables are numbered in some optimal way, ensuring efficient storage of the sparse matrix ($E - G_k$) in the computer's memory.

4 RELCH Algorithm

Step 1. Set the starting point x; calculate $J := J(x)$; set L, which determines the number of recalculations using the formula (10) (about the prior choice of L, see below).
Step 2. Calculate $g := J'(x)$, $G := J''(x)$; lay $g := g/\ ||G||$, $G := G/\ ||G||$; $\alpha : = 1$.
Step 3. According to the formula (10) build ϑ_L ; put $x^t := x + \vartheta_L$.
Step 4. Calculate $J_t := J(x^t)$. If $J_t > J$, go to step 5, otherwise go to step 6.
Step 5. Put $\alpha := \alpha/2$, $x^t := x + \alpha\vartheta_L$ and go to step 4.
Step 6. Put $x:= x^t$, $J := J_t$ and go to step 2.

The criterion of the end of the process is not specified here. As a rule, the calculations end when the specified number of calculations of the functional has been exhausted or when the algorithm is explicitly stopped. The number of recalculations L by the formula (10) is a parameter set by the user. According to (11), it is initially advisable to assume

$$L \cong \sqrt{1{,}63/\alpha_L} \cong 1{,}3\sqrt{\eta}$$

where η - assessment of the ravine degree of the minimized functional. With this choice of L, the relaxation multipliers in the positive part of the spectrum will be guaranteed to be less than 0.23. When designing algorithmic methods for L tasks, it is necessary to take into account that the sequence $\{J_s\}$, where $J_s \triangleq J\left(x^k + \vartheta_s\right)$ will not decrease monotonically with $s \to \infty$. In step 5 of the algorithm, the regulation of the advance vector norm was applied in order to prevent the local quadratic model of the functional from leaving the space of validity.

5 Convergence Characteristics

We give an estimate of the efficiency of the method (10) in comparison with the methods of conjugate gradients (SG methods). For tasks of large dimension (when the number of iterations is less than the dimension), one can guarantee the convergence of SG-methods only with the speed of a geometric progression even for strongly convex quadratic functionals.

We'll consider the case

$$f(x) = 1/2\langle Gx, x\rangle,\ G > 0$$

and estimate the rate of convergence of the SG method to the extreme point x = 0.

The iteration x^k, obtained by the SG method can be represented as

$$x^k = (E + c_1G + c_2G^2 + \ldots + c_kG^k)x^0 = P_k(G)x^0,$$

where $P_k(G)$ - matrix polynomial of degree k. Moreover, it follows from the properties of the SG method that the coefficients $c_1, \ldots, c_k$ of the polynomial $P_k(G)$ at each iteration take such values to minimize the value $f(x^k)$, which differs only by a multiplier from the error function. In other words k-e approximation minimizes $f(x^k)$ among vectors $x^0 + V$, where vector V is the element of a subspace stretched by vectors Gx^0, G^2x^0, …, G^kx^0.

We suggest

$$x^0 = \sum_{i=1}^{n} \xi_{i,0} u^i,$$

where $\{u^i\}$ - orthonormal basis of eigenvectors of the G matrix, therefore we get

$$x^k = P_k(G)\sum_{i=1}^{n} \xi_{i,0}u^i = \sum_{i=1}^{n} \xi_{i,0}P_k(\lambda)u^i,\ P_k(0) = 1,$$

$$f\left(x^k\right) = 1/2\left\langle Gx^k, x^k\right\rangle = 1/2\sum_{i=1}^{n} \xi_{i,0}^2 P_k^2(\lambda_i)\lambda_i. \tag{13}$$

Hence, we have

$$\left\|x^0\right\|^2 = \sum_{i=1}^{n} \xi_{i,0}^2,$$

$$\left\| x^k \right\|^2 = \sum_{i=1}^{n} \xi_{i,0}^2 P_k^2(\lambda_i) \le \max_i P_k^2(\lambda_i) \left\| x^0 \right\|^2 . \tag{14}$$

As a polynomial $P_k(\lambda)$ we choose the closest to optimal polynomial, the least deviating from zero on the interval $[m, M]$, containing all the eigenvalues of the positive-definite G matrix and normalized so that $P_k(0) = 1$.

By linear change of variables

$$\lambda = \frac{M+m}{2} - \frac{M-m}{2} t$$

the task is reduced to constructing a polynomial least deviating from zero on the interval $t \in [-1, 1]$ and taking at the point $t_0 = (M + m) / (M - m)$, (corresponding $\lambda = 0$) value 1. The solution of the last problem is given by the polynomial.

$$\tilde{T}_k(t) = \frac{T_k(t)}{\cos(\mathrm{k}\arccos t_0)} = \frac{T_k(t)}{T_k(t_0)},$$

where $T_k(t) = \cos(k\arccos t)$ is the Chebyshev polynomial. At the same time

$$\max_{-1\le t\le 1} \left|\tilde{T}_k(t)\right| = \frac{1}{\left|T_k(t_0)\right|} \max_{-1\le t\le 1} \left|T_k(t)\right|.$$

It is obvious

$$\max_{-1\le t\le 1} \left|T_k(t)\right| = 1,$$

that is why

$$L_k = \max_{\lambda} \left|P_k(\lambda)\right| = \max_{t} \left|\tilde{T}_k(t)\right| = \frac{1}{T_k(t_0)}, \lambda \in [m, M], t \in [-1, 1].$$

Since the conception is fair

$$T_k(t) = 0,5\left[\left(t + \sqrt{t^2-1}\right)^k + \left(t - \sqrt{t^2-1}\right)^k\right],$$

$$L_k = 2\Bigg/\left[\left(\frac{M+m}{M-m} + \sqrt{\left(\frac{M+m}{M-m}\right)^2 - 1}\right)^k + \left(\frac{M+m}{M-m} - \sqrt{\left(\frac{M+m}{M-m}\right)^2 - 1}\right)^k\right] =$$

$$= 2\Bigg/\left[\left(\frac{\sqrt{M}+\sqrt{m}}{\sqrt{M}-\sqrt{m}}\right)^k + \left(\frac{\sqrt{M}-\sqrt{m}}{\sqrt{M}+\sqrt{m}}\right)^k\right].$$

With sufficiently large k ($k \ge k_0$) we have

$$L_k \cong 2\Bigg/\left(\frac{\sqrt{M}+\sqrt{m}}{\sqrt{M}-\sqrt{m}}\right)^k = 2\left(\frac{\sqrt{\eta}-1}{\sqrt{\eta}+1}\right)^k \cong 2\left(1 - \frac{2}{\sqrt{\eta}}\right)^k, \eta \triangleq \frac{M}{m}. \tag{15}$$

From (13) and (14) we get

$$||x^k|| \le L_k ||x^0||$$

or

$$||x^k|| \le 2q^k ||x^0||, k \ge k_0, \tag{16}$$

where $q \cong \left(1 - 2/\sqrt{\eta}\right)$. Thus, the convergence of the SG method with the speed of a geometric progression is proved. The exact value of L_k, valid for any k, will be equal to

$$L_k = 2\Big/\left[\left(1+\frac{2}{\sqrt{\eta}}\right)^k + \left(1-\frac{2}{\sqrt{\eta}}\right)^k\right], L_0 = 1.$$

From (16) it follows that when $\eta >> 1$, convergence can be very slow.

SG method «finiteness» is the exact solution to the problem of minimizing a quadratic function (paraboloid) in n steps, where n is the dimension of the search space. It appears only with a sufficiently large number of iterations. The degree of the polynomial $P_k(\lambda)$ in (13) will be equal to n, and the optimal choice of this polynomial reduces to localizing its n roots at the $\lambda_1, \lambda_2, \ldots, \lambda_n$, points that will lead to an exact solution of the task ($f(x^n) = 0$).

It is easy to see that the estimate (16) for a quadratic function of a general form

$$f(x) = 1/2\langle Gx, x\rangle - \langle G, x\rangle + c$$

converted to

$$||x^k - x^*|| \leq 2q^k||x^0 - x^*||, \quad (17)$$

where x^* is the optimal point that does not coincide in the general case with the start of the coordinates.

An important feature of RELCH-type algorithms is that the corresponding relaxation multipliers will be determined only by the number of iterations L and η degree of the rigidity of the problem, regardless of the dimension n. At the same time, in the schemes of SG methods n iteration order is required to complete each cycle of descent; otherwise, according to (17), the rate of convergence may be very small. In addition, each iteration of the SG method, even for a quadratic case, requires a new gradient calculation, that is, additional computational costs for calculating the target functional. We will further assume that the RELCH algorithm is implemented with a constant $L = 1,3\sqrt{\eta}$, having relaxation multipliers in the area of $\lambda > 0$, not exceeding the value of 0.23.

We will consider the problem of minimizing a quadratic functional $f(x) = 1/2\langle Gx, x\rangle$ with a positively defined matrix G. Let us estimate the number of calculations $f(x)$ required to achieve the control vector x' with the norm $||x'|| \leq 0,23$ by the SG method and the RELCH algorithm from the starting point x^0 с $||x^0|| = 1$. When reaching the x' point the whole situation repeats, therefore, the comparative efficiency estimates obtained below are of a rather general character.

We will assume that two-sided finite difference relations are used to calculate the derivatives, which in the following analysis provides additional advantages to the SG method.

To achieve the vector x', the RELCH algorithm needs to calculate at the point x^0 a weakly filled Hesse matrix and a gradient vector $f'(x^0)$. With fill coefficient γ, it would require about $2\gamma n^2$ calculations of f. Further, we iterate $L = 1,3\sqrt{\eta}$ using formula (10), which do not require additional calculations of the objective functional f.

To obtain the vector x' the SG method will need N iterations, where N is calculated this way:

$$||x^N|| = 2q^N = 0,23,$$

it means $N \cong -2,2/\ln q$. To perform each iteration, it is necessary to update the gradient vector, that in the general case of the application of two-sided finite-difference approximations of derivatives is associated with $2n$ calculations of $f(x)$. The total number of calculations f is $-4,4n/\ln q$. The relative benefit in the number of f calculations by the RELCH method compared to the SG method is given by the function $\Psi(\eta) \cong -2,2/(\gamma n \ln q)$. It is obvious that when $\eta \to \infty$ we have $q(\eta) \to 1$ and $\Psi(\eta) \to \infty$. Typical values Ψ for $\gamma = 0,01$ and $n = 1000$ are given below:

η	…	100	1000	1500	10^4	10^5
Ψ	…	1,0	3,4	4,0	11,0	35,0

Thus, to obtain comparable results when $\eta = 10^4$, the RELCH algorithm will require approximately 11 times less f calculations than with the SG method. However, it should be taken into account that as η increases, the number of L recalculations by the formula (10) grows. This can lead to an increase in the influence of computational errors in the calculation of ϑ_s with large s numbers.

Example. We will consider the model task of the quadratic functional minimization f(x) c n = 200, h = 1500, g = 0,025. For definiteness, we assume that the time of a single calculation of f(x) is equivalent to performing 102n multiplication operations with floating point. The execution time of a single multiplication operation for some device for definiteness conditionally we will assume equal to ty = 3·10– 5sec. The calculation of the f(x) value takes at the same time tf = 0,6 sec of processor time. To calculate f′ и f″ using the general finite-difference formulas, we need, respectively, t′ = 2ntf = 4 min, t″ = 2γn2t = 20 min. The number of recalculations by the formula (10) is equal to $L = 1,3\sqrt{\eta} = 50$. At each recalculation, a slightly filled matrix E – 2Gk is multiplied by a vector ϑ_{50}, that requires γn2ty ≅ 3·10– 2 sec of computer time. The vector construction time ϑ_{50} without taking into account the calculation of f′, f″ will be about 50·3·10– 2 = 1,5 seconds and may not be taken into account.

The result is that to build the control vector x' when $||x'|| < 0{,}23$ by using the RELCH method, it will take about $t'' = 20$ min of machine time. The SG method will cost, respectively, $\Psi(1500)\cdot 20 \cong 1{,}3$ hours. When the RELCH algorithm is re-applied to the constructed vector x', we will get the vector x'' с $||x''|| \leq 0{,}23||x'||$ etc. Therefore, if we denote the corresponding sequence of vectors by $\{x^m\}$, the norm of the vector x will decrease according to the law of geometric progression $||x^m|| \leq d^m||x^0||$, where d <0.23 regardless of the η and n values.

An important additional advantage of the RELCH algorithm in comparison with the SG method is its rather high efficiency in the non-convex case, since the relaxation function of the method in the left half-plane is entirely located in the allowed area and the relaxation multipliers for $\lambda < 0$ grow rapidly by absolute value when going from ϑ_s to ϑ_{s+1}. Growth characteristics were given before.

6 Conclusion

The described class of matrix gradient methods has shown in practice a sufficiently high performance in conditions of high stiffness and non-convexity of objective functionals. When optimizing large systems, it is possible to use efficient "packed" storage forms for matrices of second derivatives, that significantly reduces the requirements for the necessary computer memory. However, the method retains its main characteristics for small-sized systems, competing with the main optimization procedures of nonlinear programming.

Acknowledgements

The work was financially supported by the Ministry of Education and Science of the Russian Federation in the framework of the Federal Targeted Program for Research and Development in Priority Areas of Advancement of the Russian Scientific and Technological Complex for 2014-2020 (No. 14.584.21.0022, ID RFMEFI58417X0022).

References

[1] I. Chernorutskiy, P. Drobintsev, V. Kotlyarov and N. Voinov. A New Approach to Generation and Analysis of Gradient Methods Based on Relaxation Function. Proceedings - 2017 UKSim-AMSS 19th International Conference on Modelling and Simulation, UKSim 2017:83-88, May 2018.

[2] I. G. Chernorutsky. Algorithmic problems of stiff optimization. St. Petersburg Polytechnic University Journal of Engineering Science and Technology, №6:141-152, 2012.

[3] Yu. E. Nesterov and B. T. Polyak. Cubic regularization of Newton method and its global performance. Mathematical Programming, 108(1):177-205, August 2006.

[4] M. Avriel. Nonlinear programming: analysis and methods - Mineola, NY: Dover Publishing, 2003.

[5] P. Deuflhard. Newton methods for nonlinear problems. Affine invariance and adaptive algorithms - Volume 35 / Springer Series in Computational Mathematics, Springer, 2004.

[6] S. Pissanetski.Technology of sparse matrices. - Mir, 1988.