High performance computing for global optimization problems

K.A. Barkalov, V.P. Gergel

Lobachevsky State University of Nizhni Novgorod, Gagarin avenue 23, Nizhn Novgorod, Russia, 603950

E-mail: konstantin.barkalov@itmm.unn.ru

Abstract. In the present work, the multiextremal optimization problems and a high-performance parallel algorithm for solving these ones are considered. The investigation of the algorithm scalability has been carried out on the problem class, in which the computation costs of the function value evaluations can be varied at different iteration points. The algorithm proposed in the present work can utilize the CPUs (for solving more complex subproblems) as well as the GPUs (for solving the simple subproblems). The results of numerical experiments demonstrating the speedup when solving a series of multiextremal constrained problems are presented.

1. Introduction

In the present paper, the constrained multiextremal optimization problems and the parallel methods for solving these ones are considered. The fact that the global extremum is an integral characteristic of a problem is an important feature of the multiextremal problems. Thus, its finding is related to the construction of a coverage of the search domain and the computing the function values at all points of this coverage. The dimensionality affects the difficulty of solving the problems of considered class crucially: the computational costs grow with increasing this one exponentially, therefore, for solving such problems, the methods are required, which generate a nonuniform grid in the search domain, denser near the global minimizer and more sparse far away from this one. In the present paper, the results of scalability investigation of the parallel index global optimization algorithm developed in Lobachevsky State University of Nizhni Novgorod [1, 2] and implemented in the Globalizer system [3, 4] are presented.

Within the framework of the discussed approach, the solving of the multidimensional problems is reduced to the solving of a set of the connected subproblems of less dimensionality. The corresponding dimensionality reduction is based on the use of the evolvents of a unit interval on the real axis onto a hypercube. The continuous unambiguous mappings like Peano curves, called also the space-filling curves, play the role of such evolvents. The nested optimization scheme is one more mechanism used for reducing the dimensionality of the problem being solved. The numerical methods allowing the efficient utilization of such reduction schemes had been developed in details and substantiated in [1, 2].

In the algorithms considered in the present paper the objective function and constraints are assumed to be Lipschitzian that is typical for many other approaches to the development of the parallel optimization algorithms (see, for example, [5, 6, 7, 8]). This assumption is natural for

many applied problems since the relative variations of the functions generally cannot exceed a certain threshold determined by the bounded energy of changes occurring in the system under study. Also, a standard assumption is that the execution of a single trial (the computing of the values of the objective function and constraints at a point in the search domain) is a time-consuming operation since it utilized the results of numerical modeling.

An assumption on different costs of computing the problem functions subject to various components of the vector of parameters is a novel element investigated in the present work. It is assumed that it is possible to select the "difficult" and "easy" parts of the functions connected with each other. An example of a problem of this type has been considered in [12]. Then, the solving of the "difficult" part of the problem (for which the execution of time-consuming algorithms is required to conduct the trials) can be performed on CPU using an efficient index global optimization algorithm. The "easy" part of the problem (the execution of which doesn't require the complex algorithms and can be transferred to an accelerator easily) can be solved on GPU using the uniform grid technique. Obviously, in this case a major portion of the trials will be carried out on GPU, and the minor one will be carried out on CPU. However, because of the difference in the difficulties of the subproblems solved on different units, a speedup of the algorithm as a whole can be expected. The proposed approach has been implemented in Globalizer parallel software system for solving the global optimization problem developed in Lobachevsky State University of Nizhni Novgorod.

2. Problem statement

Let us consider the N-dimensional global optimization problem

$$\varphi(y^*) = \min\{\varphi(y) : y \in D,\ g_i(y) \le 0,\ 1 \le i \le m\} \tag{1}$$

with the search domain

$$D = \left\{y \in R^N : a_j \le y_j \le b_j, 1 \le j \le N\right\}.$$

The objective function $\varphi(y)$ (henceforth denoted by $g_{m+1}(y)$) and the left-hand sides $g_i(y),\ 1 \le i \le m$, of the constraints satisfy the Lipschitz conditions with constants L_i, $1 \le i \le m+1$, respectively, and may be multiextremal. It is assumed that the functions $g_i(y),\ 1 \le i \le m+1$, are defined and computable only in the corresponding domains

$$Q_1 = D,\ Q_{i+1} = \{y \in Q_i : g_i(y) \le 0\},\ 1 \le i \le m.$$

These conditions allow for the introduction of a classification of the points $y \in D$ according to the number $\nu(y)$ of the constraints computed at this point.

Thus, a *trial* at a point $y^k \in D$ executed at the k-th iteration of the algorithm will consist in computing the values $g_1(y^k), ..., g_\nu(y^k)$, where the index $\nu \le m$ is determined by the conditions

$$g_i(y^k) \le 0,\ 1 \le i < \nu,\ g_\nu(y^k) > 0,\ \nu \le m.$$

The occurrence of the first violation of the constraint terminates the trial. In the case when the point y^k is a feasible one, i.e. when $y \in Q_{m+1}$, the trial includes the computation of the values of all the functions of the problems and the index is assumed to be $\nu = m+1$. The pair of values

$$\nu = \nu(y^k),\ z^k = g_\nu(y^k)$$

is a *trial result*.

The main idea of the index algorithm which can be applied to solving such problems with partially defined functions consists in the reduction of the initial multidimensional problem to a set of the subproblems of less dimensionality and the solving of these ones in parallel. The index method and dimensionality reduction schemes are described briefly in the next sections.

3. Parallel index method

In this section we consider a one-dimensional multiextremal optimization problem

$$\varphi(x^*) = \min\{\varphi(x) : x \in [a,b],\ g_i(x) \le 0,\ 1 \le i \le m\}.$$

Let us describe a rules of parallel index method for the case when the algorithm carries out during one iteration $p > 1$ trials simultaneously (each trial in a separate thread or process). Let $k(n)$ be the total number of trials, carried out after n parallel iterations.

Suppose $n \ge 1$ iterations of the method have been carried out, in which trials were performed at $k = k(n)$ points $x^i, 1 \le i \le k$. Then points $x^{k+1}, \ldots, x^{k+p}$ of search trials of the next $n+1$ iteration are determined according to the following rules.

Rule 1. Renumber the points $x^1, ..., x^k$ of the preceding trials by the lower indices in ascending order of coordinate values, i.e.

$$0 = x_0 < x_1 < \cdots < x_k < x_{k+1} = 1,$$

and juxtapose to them the values $z_i = g_\nu(x_i),\ \nu = \nu(x_i),\ 1 \le i \le k$, computed at these points. The points $x_0 = 0$ and $x_{k+1} = 1$ are introduced additionally, while the values z_0 and z_{k+1} are not defined.

Rule 2. Classify the indices $i,\ 1 \le i \le k$, of the trial points according to the number of the problem constraints fulfilled at these points, by constructing the sets

$$I_\nu = \{i : 1 \le i \le k,\ \nu = \nu(x_i)\},\ 1 \le \nu \le m+1,$$

containing the numbers of all the points $x_i,\ 1 \le i \le k$, with the same values of ν. The end points $x_0 = 0$ and $x_{k+1} = 1$ are interpreted as the ones having indices equal to zero. An additional set, $I_0 = \{0, k+1\}$, corresponds to them.

Determine the maximum value of the index:

$$M = \max\{\nu(x_i),\ 1 \le i \le k\}.$$

Rule 3. Compute the current lower estimates

$$\mu_\nu = \max\left\{\frac{|z_i - z_j|}{x_i - x_j},\ i,j \in I_\nu,\ i > j\right\},\ 1 \le \nu \le m+1, \tag{2}$$

for the unknown Lipschitz constants L_ν of the functions $g_\nu(x), 1 \le \nu \le m+1$. If a set I_ν contains less than two elements, or if μ_ν is equal to zero, then assume $\mu_\nu = 1$.

Rule 4. For all nonempty sets $I_\nu,\ 1 \le \nu \le m+1$, compute the estimates

$$z_\nu^* = \begin{cases} -\epsilon_\nu, & \nu < M, \\ \min\{g_\nu(x_i) : i \in I_\nu\}, & \nu = M, \end{cases}$$

where the nonnegative numbers $(\epsilon_1, ..., \epsilon_m)$ are parameters of the algorithm.

Rule 5. For each interval $(x_{i-1}, x_i),\ 1 \le i \le k+1$, compute the *characteristics* $R(i)$:

$$R(i) = 2\Delta_i - 4\frac{z_i - z_\nu^*}{r_\nu \mu_\nu},\ \nu = \nu(x_i) > \nu(x_{i-1}),$$

$$R(i) = \Delta_i + \frac{(z_i - z_{i-1})^2}{r_\nu^2 \mu_\nu^2 \Delta_i} - 2\frac{z_i + z_{i-1} - 2z_\nu^*}{r_\nu \mu_\nu},\ \nu = \nu(x_i) = \nu(x_{i-1}), \tag{3}$$

$$R(i) = 2\Delta_i - 4\frac{z_{i-1} - z_\nu^*}{r_\nu \mu_\nu},\ \nu = \nu(x_{i-1}) > \nu(x_i),$$

where $\Delta_i = x_i - x_{i-1}$. The values $r_\nu > 1,\ 1 \le \nu \le m+1$, are parameters of the algorithm. An appropriate selection of r_ν allows to consider the product $r_\nu \mu_\nu$ as an estimate of the Lipschitz constants $L_\nu,\ 1 \le \nu \le m+1$.

Rule 6. Arrange characteristics $R(i), 1 \le i \le k+1$, in decreasing order

$$R(t_1) \ge R(t_2) \ge \cdots \ge R(t_k) \ge R(t_{k+1}) \tag{4}$$

and select p maximum characteristics with interval numbers $t_j, 1 \le j \le p$.

Rule 7. Carry out new trials at points $x^{k+j} \in (x_{t_j-1}, x_{t_j}), 1 \le j \le p$, calculated using the formulae

$$x^{k+j} = \frac{x_{t_j} + x_{t_j-1}}{2},\ \text{if } \nu(x_{t_j-1}) \ne \nu(x_{t_j}),$$

$$x^{k+j} = \frac{x_{t_j} + x_{t_j-1}}{2} - \frac{z_{t_j} - z_{t_j-1}}{2r_\nu \mu_\nu},\ \text{if } \nu = \nu(x_{t_j-1}) = \nu(x_{t_j}). \tag{5}$$

The algorithm terminates if the condition $\Delta_{t_j} < \epsilon$ is satisfied at least for one number $t_j, 1 \le j \le p$; $\epsilon > 0$ is the predefined accuracy.

This method of parallelization has the following justification. The characteristics of intervals (3) used in the index method can be considered as probability measures of the global minimum point location in these intervals. Inequalities (4) arrange intervals according to their characteristics, and trials are carried out in parallel in the first p intervals with the largest probabilities.

4. Dimensionality reduction

4.1. Dimensionality reduction using space-filling curves

The use of Peano curve $y(x)$

$$\left\{y \in R^N : -2^{-1} \le y_i \le 2^{-1}, 1 \le i \le N\right\} = \{y(x) : 0 \le x \le 1\}$$

unambiguously mapping the interval of real axis $[0,1]$ onto a N-dimensional cube is the first of the dimensionality reduction methods considered. To implement this method of dimensionality reduction a numerically constructed curve (*evolvent*) is used. The evolvent is 2^{-m} accurate approximation of the theoretical Peano curve in L_∞ metric, where m is an evolvent construction parameter. Problems of numerical construction of the evolvents and the corresponding theory are considered in detail in [1]. Examples of the evolvent with different m in two dimensions are given in figure 1.

By using this kind of mapping it is possible to reduce the multidimensional problem (1) to a univariate problem

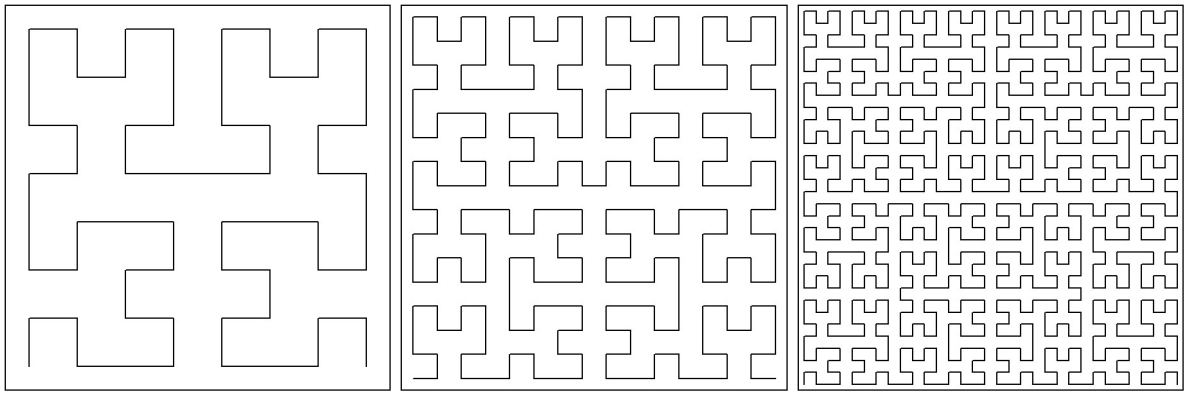
$$\varphi(y(x^*)) = \min\left\{\varphi(y(x)) : x \in [0,1],\ g_i(y(x)) \le 0,\ 1 \le i \le m\right\}.$$

The considered dimensionality reduction scheme juxtaposes a multidimensional problem with lipschitzian functions to a univariate problem where the corresponding functions satisfy the uniform Hölder condition (see [1]), i.e.

$$\left|g_i(y(x')) - g_i(y(x''))\right| \le H_i \left|x' - x''\right|^{1/N},\ x', x'' \in [0,1],\ 1 \le i \le m+1. \tag{6}$$

Here N is the dimensionality of the initial multidimensional problem and the coefficients H_i are related with the Lipschitz constants L_i of the initial problem by the inequalities $H_i \le 2L_i\sqrt{N+3}$.

Relation (6) allows modification of the parallel index algorithm described in previous section for solving multidimensional problems reduced to one-dimensional ones. For this purpose the



(a) (b) (c)

Figure 1. Evolvents in two dimensions with (a) $m = 3$, (b) $m = 4$ and (c) $m = 5$

intervals lengths Δ_i from the rules (2) and (3) of the index method are replaced with the lengths in a new metric

$$\Delta_i = (x_i - x_{i-1})^{1/N},$$

and formula (5) is replaced by the expression

$$x^{k+j} = \frac{x_{t_j} + x_{t_j-1}}{2} - \frac{\mathrm{sign}(z_{t_j} - z_{t_j-1})}{2r_\nu} \left[\frac{|z_{t_j} - z_{t_j-1}|}{\mu_\nu} \right]^N, \ \nu = \nu(x_{t_j-1}) = \nu(x_{t_j}).$$

Various modifications of this algorithm and the corresponding theory of convergence are given in [1, 2, 15].

4.2. Nested optimization scheme

Nested optimization scheme is based on relation (see [2])

$$\min_{y \in D} \{\varphi(y) : \ g_i(y) \le 0, \ 1 \le i \le m\} = \min_{u_1 \in D_1} \min_{u_2 \in D_2} ... \min_{u_M \in D_M} \{\varphi(y) : \ g_i(y) \le 0, \ 1 \le i \le m\},$$

which allows replacing the solving of multidimensional problem (1) by solving a family of subproblems related to each other recursively. Here we consider vector y as a vector of block variables

$$y = (y_1, y_2, ..., y_N) = (u_1, u_2, ..., u_M),$$

where the i-th block variable u_i is a vector of N_i components of vector y, taken serially, i.e. $u_1 = (y_1, y_2, ..., y_{N_1})$, $u_2 = (y_{N_1+1}, y_{N_1+2}, ..., y_{N_1+N_2})$,..., $u_M = (y_{N-N_M+1}, y_{N-N_M+2}, ..., y_N)$, and $N_1 + N_2 + ... + N_M = N$. The subdomains $D_i, 1 \le i \le M$, are projections of initial search domain D onto the subspaces corresponding to the variables $u_i, 1 \le i \le M$.

The number of vectors M and the quantity of components in each vector $N_1, N_2, ..., N_M$ are the parameters of the nested optimization scheme and can be used to form the subproblems with necessary properties. For example, if $M = N$ i.e. $u_i = y_i, 1 \le i \le N$, each nested subproblem is a one-dimensional one. If $M = 1$, i.e. $u = u_1 = y$, the solving of the problem is equivalent to solving this one using a single evolvent mapping $[0, 1]$ into D; the nested subproblems are absent.

In the present study, this scheme has been applied for $M = 2$

$$\min_{y \in D} \{\varphi(y) : \ g_i(y) \le 0, \ 1 \le i \le m\} = \min_{u_1 \in D_1} \min_{u_2 \in D_2} \{\varphi(y) : \ g_i(y) \le 0, \ 1 \le i \le m\},$$

i.e. only one nesting level has been used.

5. Organization of parallel computing

The organization of parallel computing with the use of the recursive optimization scheme has been considered in details for the shared/distributed memory as well as for the accelerators in [13, 14]. However, in these works the problems were considered, in which the time of the trial execution didn't depend on the trial point. Here, the problem is considered assuming that in the function $\varphi(y_1, ..., y_N)$ it is possible to select more difficult part $f(y_1, ..., y_s)$ (depending on a part of the parameters only) and a simpler part $g(y_1, ..., y_N)$ (depending on all problem parameters), for example, $\varphi(y_1, ..., y_N) = f(y_1, ..., y_s)g(y_1, ..., y_N)$.

The difficult part of the function implies performing some computations related to the numerical simulation, which can be performed on the CPUs only. The simple part doesn't imply the complex computations and can be computed on an accelerator, for example, GPU. For solving a problem with such a structure, one can apply the recursive optimization scheme in parallel with the use CPU at the upper nesting level and GPU at the lower one.

6. Numerical experiments

The recursive scheme of solving the global optimization problems has been implemented in the Globzlizer solver developed in Lobachevsky State University of Nizhni Novgorod [3, 4]. The global search methods and various dimensionality reduction schemes make the algorithmic base for the Globalizer. The numerical experiments, the results of which are presented in [16, 17] demonstrate these methods, at least, are not worse than the well known methods applied for similar purposes, and even overcome these ones with respect to some parameters.

Let us conduct the investigation of the scalability of the parallel algorithm by solving a series of 100 test problems of the constrained global optimization. In work [18] the approach has been proposed, which allowed generating the constrained global optimization problems with the following properties:

- one could control the size of the feasible domain Q_{m+1} with respect to the whole search domain D;
- the global minimizer of the objective function is known a priori taking into account the constraints;
- the global minimizer of the objective function without accounting for the constraints is out of the feasible domain Q_{m+1} (with the purpose of simulating the behavior of the constraints and the objective function in the applied constrained optimization problems).

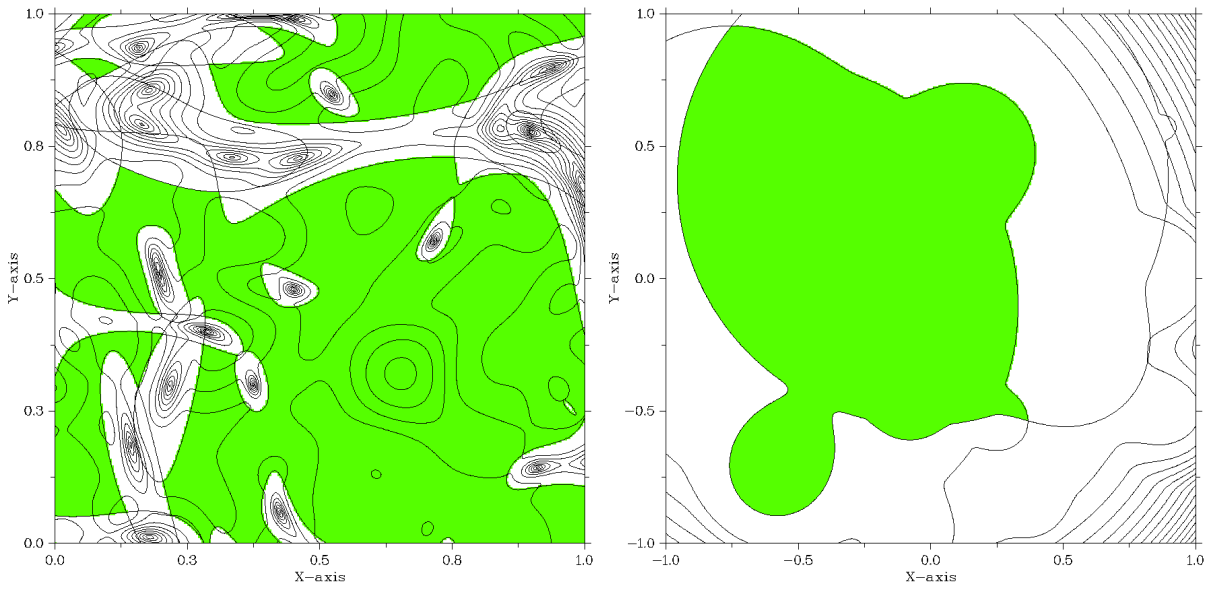
In order to simulate the applied problems with various computational difficulty, we will use a combination of the function classes of the kind

$$\varphi(y_1, ..., y_N) = p(y_1, ..., y_N)(f(y_1, y_2) + g(y_3, ..., y_N)).$$

Here $f(y_1, y_2)$ is a two-dimensional function from the class described in [19], $g(y_3, ..., y_N)$ is a function of the dimensionality $N-2$ from the class described in [20], and $p(y_1, ..., y_N)$ is a second order polynomial. The multiplication by $p(y_1, ..., y_N)$ excluded the possibility of separable search of the minima of the functions $f(y_1, y_2)$ and $g(y_3, ..., y_N)$.

The level lines of the two-dimensional subproblems based on the functions $f(y_1, y_2)$ and $g(y_3, y_4)$ are shown in figure 2. The feasible domains are highlighted by color. One can see that the subproblem (a) has more complex structure as compared to the subproblem (b). At the same time, computing the values of the function $f(y_1, y_2)$ is more time-consuming than of $g(y_3, y_4)$.

The numerical experiments were carried out using two classes of problems (*Simple* ans *Hard*) with $N = 5$. The problem was considered to be solved if the algorithm generates a trial point y^k in δ-vicinity of the global minimizer, i.e. $\left\|y^k - y^*\right\| \leq \delta$. The size of the vicinity was selected



(a) (b)

Figure 2. The level lines of (a) hard and (b) simple subproblems

Table 1. Average time for solving the problem on one node

Problems	Time (sec)
Hard	52
Simple	51

Table 2. Average time for solving the problem on p nodes

Problems	$p = 8$	$p = 16$	$p = 32$
Hard	11.51	5.53	0.54
Simple	2.04	1.50	0.47

as $\delta = 0.01\,\|b - a\|$, where a and b are the boundaries of the search domain D. The maximum allowable number of iterations was $K_{max} = 10^6$.

Let us conduct the first experiment by solving the *Simple* and *Hard* problem series on a single node employing both available CPUs in full (i.e. $p = 16$ cores available). In table 1 the averaged time (in seconds) required to solve the problems of the series is presented.

Let us conduct the next experiment employing p nodes each including three graphic accelerators. Thus, at $p = 32$ total 96 GPU accelerators were employed each including 2688 CUDA cores; 258048 CUDA cores in all. In tables 2 and 3 the average time and the speedup with respect to the run on a single node are presented.

The results of experiments demonstrate that the parallel index algorithm combined with the nested optimization scheme provides a good speedup on the considered problem class.

Table 3. Speedup with respect to one node

Problems	$p = 8$	$p = 16$	$p = 32$
Hard	5	9	96
Simple	25	34	109

Acknowledgments

This research was supported by the Russian Science Foundation, project No 16-11-10150 "Novel efficient methods and software tools for the time consuming decision making problems with using supercomputers of superior performance".

References

[1] Strongin R G and Sergeyev Ya D 2000 *Global Optimization with Non-Convex Constraints. Sequential and Parallel Algorithms* (Dordrecht: Kluwer Academic Publishers)
[2] Strongin R G, Gergel V P, Grishagin V A and Barkalov K A 2013 *Parallel Computing in Global Optimization Problems* (Moscow: Moscow State University Press) (in Russian).
[3] Sysoyev A V, Barkalov K A, Sovrasov V V, Lebedev I G and Gergel V P 2017 *Lect. Notes Comput. Science* **10421** 492–499
[4] Gergel V, Barkalov K and Sysoyev A 2018 *Numer. Algebra Contr. Optim.* **8(1)** 47–62
[5] Floudas C A and Pardalos P M 2009 *The Encyclopedia of Optimization* (Heidelberg: Springer) p 725–735
[6] Paulavičius R, Žilinskas J and Grothey A 2011 *Optim. Meth. Soft.* **26(3)** 487–498
[7] Evtushenko Yu G, Malkova V U and Stanevichyus A A 2009 *Comput. Math. Math. Phys.* **49(2)** 246–260.
[8] Evtushenko Yu G and Posypkin M A 2014 *Autom. Remote Control* **75(6)** 1025–1040
[9] Kovartsev A N 2017 *Computer Optics* **41(1)** 118–125 (In Russ.)
[10] Malysheva S A and Golovashkin D L 2016 *Computer Optics* **40(2)** 179–187 (In Russ.)
[11] Vorotnikova D G and Golovashkin D L 2017 *Computer Optics* **41(1)** 134–138 (In Russ.)
[12] Barkalov K A and Strongin R G 2017 *Lect. Notes Comput. Science 10556* 18–33
[13] Barkalov K, Gergel V and Lebedev I 2016 *AIP Conf. Proc.* **1738** 400006
[14] Barkalov K A and Lebedev I G 2016 *Commun. Comp. Inf. Science* **687** 224–235
[15] Gergel V P and Strongin R G 2003 *Lect. Notes Comput. Science 2763* 76–88
[16] Gergel V and Lebedev I 2015 *Proc. Comput. Science* **66** 53–62
[17] Gergel V and Sidorov S 2015 *Lect. Notes Comput. Science* **9251** 505–515
[18] Gergel V 2017 *Lect. Notes Comput. Science* **10556** 314–319
[19] Gergel V, Grishagin V and Gergel A 2016 *J. Glob. Optim.* **66(1)** 35–51
[20] Sergeyev Y D and Kvasov D E 2015 *Commun. Nonlinear Sci. Numer. Simul.* **21(1-3)** 99–111