ISBN 978-1-257-96027-9
90000
9 781257 960279
ID: 11033595
www.lulu.com

Solution of some problems on multicore processors. New methods and modeling in the MATLAB.

Solution of some problems on multicore processors

New methods and modeling in the MATLAB

Solomon I. Khmelnik

Khmelnik S.I.

Solution of some problems on multicore processors.

New methods and modeling in the MATLAB

Israel 2011

Published by Mathematics in Computer Computers "MiC", Israel
BOX 15302, Bene-Ayish, Israel, 60860
Fax: ++972-8-8691348
solik@netvision.net.il
Printed in United States of America, Lulu Inc.
Content ID: 1nq22kmk
Created: 08/03/2011
ISSN 978-1-257-96027-9

Annotation

We are presenting a new method and algorithm for solving several common problems of linear algebra and optimization, where **only** vectors multiplication is used (matrices inversion and division is absent). The solution of these problems is reduced to consequently performed multiplication of a matrix by a vector. It leads to significant simplification of the corresponding programs for multi-core processors, and the solution time is very much reduced.

Open codes of the programs in MATLAB system are presented. In these programs the above mentioned method is realized for a single-core processor.

Programs can be purchased from the author to the address: solik@netvision.net.il.

Contents

Preface

It is well known that 75% of all numerical mathematical problems are essentially the problems of linear algebra [1]. It will not be an exaggeration to say that the super processors and multi-core processors owe their upraise exactly to these problems. But in these problems only the multiplication of matrices harmonizes ideally with the with the possibility of parallel computations in matrix processors. Other operations, necessary for bringing the linear system to a form suitable for iterations or for matrix inversion [2], are ill fitted to parallelizing. This problem, along with the high cost of matrix processors, is an obstacle to their expansion.

In the most part the above mentioned problems are as follows

1. Solving an equations system with complex variables, including underdetermined and overdetermined equations systems.
2. Solving the systems of equations and inequalities
3. Solving a quadratic programming problem with constraints in the form of equalities and inequalities.

These problems, aside from their direct use, are included as sub-problems into numerous applied mathematical problems. Their importance is stressed also because it is a test problem for comparison between multi-core systems and super processors. At the same time these problems are hard to parallelize (it is caused by the fact that for their solution it is anyway necessary to find the inverse matrix, and each element of the inverse matrix in general is significantly depending on all the elements of the original matrix). Thus, it is for these problems that the advantage of multi-cored systems is poorly expressed.

We are presenting a method (and MATLAB-programs for its realization) for solving the above named problems; this method consists in consequently performed multiplications of matrix by vector. Such operation is easily parallelized (and there exist several methods for doing it [5, 6, 7]).

Thus, the proposed method [3, 4] for solving these problems consists **only** of the multiplication of matrix by vector. There is no matrix inversion and division operations. Such programs are much simpler for multi-core processor and their speed is much higher (we can assume that the solution time for these problems is reduced 5-10 times).

The method is based on a principle found by the author – variational optimum principle in linear alternating current electric circuits. This means that for every such circuit there exists a sole optimum of a certain

functional. This sole optimum may be found with the aid of a high-speed gradient descent method, which is very attractive for applications. There exists an inverse proportionality between the accuracy and the solution time. On practice it means that the user may quickly look through approximate solutions, and then compute a chosen variant with more accuracy. The method may be employed not only for computing complex electric circuits, but also for solving listed tasks.

Thus,

- In presented method only algebraic addition and multiplication of vectors are being used.
- The inverse matrix computation is absent
- Such processor should contain **only** summators.
- The number of summators S should be in proportion with the processor's volume and in inverse proportion with the performance time of these operations.
- The solution of linear equations system for an proposed matrix processor is performed iteratively. Here on every iteration the multiplication of a squarte matrix by a vector is being performed.
- The solution time for an proposed matrix processor is proportional to N^3/S, where N - the vector dimension.

1. The First Method for Solution of Linear Algebraic Equations System

1.1. Introduction

In this and in the following section we are describing a new method and algorithm for solving the systems of linear equations (including the underdetermined and overdetermined ones) with complex coefficients. This method is based on the author's variational optimum principle which may be observed in linear AC electric circuits [1]. It means that for every such circuit there exists a unique optimum of a certain functional. Therefore, there exists a unique optimum of a functional whose optimal value is a solution of a linear equations system with complex coefficients. This unique optimum may be found by a fast gradient descent method, which is very attractive for the applications. Also, there exists an inverse relationship between the accuracy and the solution time. On practice it means that the user may promptly sort through the approximate variants, and then compute the chosen variants with greater accuracy.

1.2. Description of the Method

The system being solved should have the following form

$$(a + jb)x = c\,, \tag{1}$$

where

x – the complex variables vector,

$a,\ b$ – given squarte matrixes,

d - a given vector.

On each iteration the new value of charge is found from the formula

$$x =: x - \frac{p \otimes p}{(pa + jpb) \otimes p}\, p\,. \tag{2}$$

Here symbol $\otimes$ denotes the operation of scalar multiplication of complex vectors and the summation of these products. The result of such operation is a real number. It is easy to see that

$$a \otimes b = \mathrm{Re}(a) \cdot b^T + \mathrm{Im}(a) \cdot b^T\,. \tag{2a}$$

Here and further superscript «T» denotes the operation of conjugation.

Notice that formula (2) **cannot be** presented in the form:

$$x =: x - \frac{p \otimes p}{p(a + jb) \otimes p} p,$$

Notice also that using the formula

$$x =: x - \frac{p \Theta p}{p(a + jb) \Theta p} p \qquad (2в)$$

instead of formula (2) leads (as the experiments show) to the same results and affects not very significantly the accuracy and speed of calculation. Here the symbol Θ denotes the component-wise multiplication of complex vectors and summation of the obtained products. The result of such operation will be a complex number (compare with the operation $\otimes$). It is easy to see that

$$a \Theta b = a \cdot b^T . \qquad (2c)$$

The gradient by which we move to the defined functional's optimum, has the following form

$$p = (a + jb)x - c . \qquad (3)$$

Thus, the computations are conducted according to the following gradient descent algorithm.

Algorithm 1

1. fix $x = 0$;
2. compute gradient *p* using formula (3);
3. determine the norm $\|p\|$ of gradient *p;*
4. if $\|p\| < \varepsilon$ the computation is terminated with the value of *x,* which has been fixed previously;
5. otherwise the new value is determined using the formula (2);
6. points 2-5 are repeated.

During the movement to an optimum the norm $\|p\|$ of this gradient (3) is decreased.

The following fig. 1 shows the typical graph of $\|p\|$ as a function of the iteration number. The following fig. 2 shows for the same case the graph of $\log\|p\|$ as a function of the iteration number.

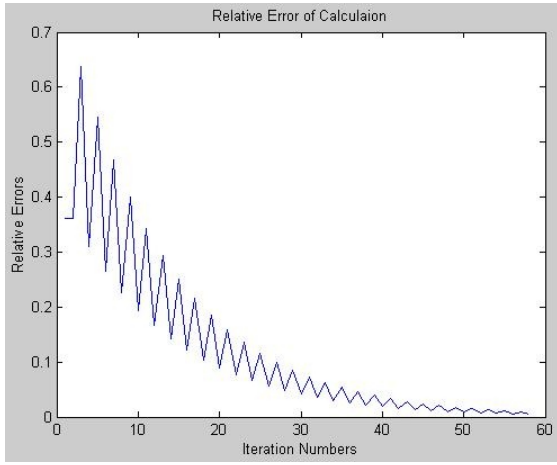

Fig. 1.

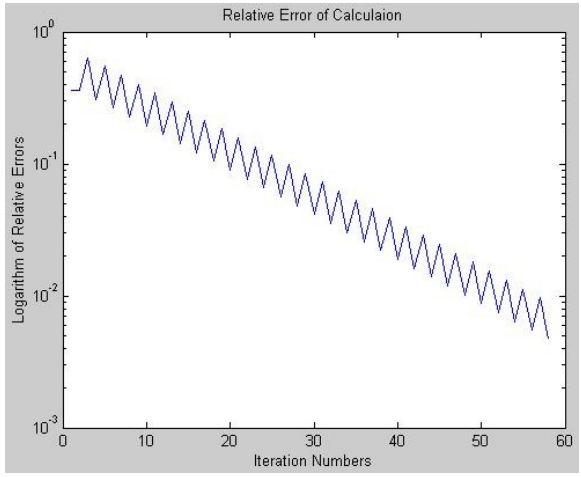

Fig. 2.

Example 1. One branch. Let us assume that the system (1) consists of only one equation. On the first iteration $x = 0, \quad p = -c$ and from (2) follows that $x = -\dfrac{p}{(a + jb)}$.

The norm may be computed by different formulas. For instance, the following formula may be used:

$$\left\| p = \begin{vmatrix} p_1 \\ \dots \\ p_k \\ \dots \end{vmatrix} \right\| = \max_k |p_k| \cdot \qquad (3a)$$

As the norm is needed only for comparison, the following formula may be applied instead of (3a):

$$\|p\| = \max_k (|p_k|)^2 . \qquad (3в)$$

The latter formula is preferable for hardware realization, as

$$\|p\| = \max_k (|p_k|)^2 = (\mathrm{Re}\, p)^T \cdot \mathrm{Re}\, p + (\mathrm{Im}\, p)^T \cdot \mathrm{Im}\, p . \qquad (3c)$$

Simultaneously with the solution of equation (1) two functions written below are being optimized:

$$F_1(q) = \frac{\pi}{\omega} \left(x^T a - 2c \right) \otimes x , \qquad (4)$$

$$F_2(q) = -\pi \left(\omega x^T b + 2jc \right) \otimes x . \qquad (5)$$

1.3. Main program

M-function for the calculation is as follows:

```
function [res,erMin,k,q,er] = ...
        SinLin(E,Z,maxEr,maxK)
```

In this program the following parameters are used:

input parameters:

E – vector c, defined in (1),

Z = a + j*b – see (1),

maxEr = ε - relative mean square residua,

maxK - allowed number of iterations,

output parameters:

q = x - calculation result,

k - reached number of iterations,

er – array of errors at each iteration,

erMin – received error of result,

res – a sign of the result:

- res=0 – result was obtained,
- res=1 – exceeded allowed number of iterations,
- res=2 - exceeded the permissible error.

1.4. Tests

In the tests additionally made the following notation:

qt=xt=Z\c – result of the traditional calculation,

ert = Z*xt - c – error of the traditional calculation,

comp – relative error of calculations results for this and the traditional algorithms.

Example 4. Program for solving one equation with complex coefficients in the MATLAB:

```
function [xt,x,erMin,k] = test1
```

Example 5. Program for solving two equations with complex coefficients in the MATLAB:

```
function [res,qt,q,erMin,k] = test2
```

Displays graphs in the form of Fig. 1 and Fig. 2.

Example 6. Program for solving three equations with complex coefficients in the MATLAB:

```
function [res,qt,q,erMin,k] = test3
```

Displays graphs in the form of Fig. 1 and Fig. 2.

Example 7. Program for solving N equations with complex coefficients in the MATLAB:

```
function [comp,erMin,k] = testN(N, maxEr)
```

Displays graphs in the form of Fig. 1 and Fig. 2.

Graphs of the form shown in Fig.1 and Fig. 2 are depicted. The parameter comp serves to compare the calculations results using the considered and the traditional algorithms.

1.5. About convergence

The iterative process *converges*, if the matrices *M* and *R* are *of fixed sign* (positive or negative).

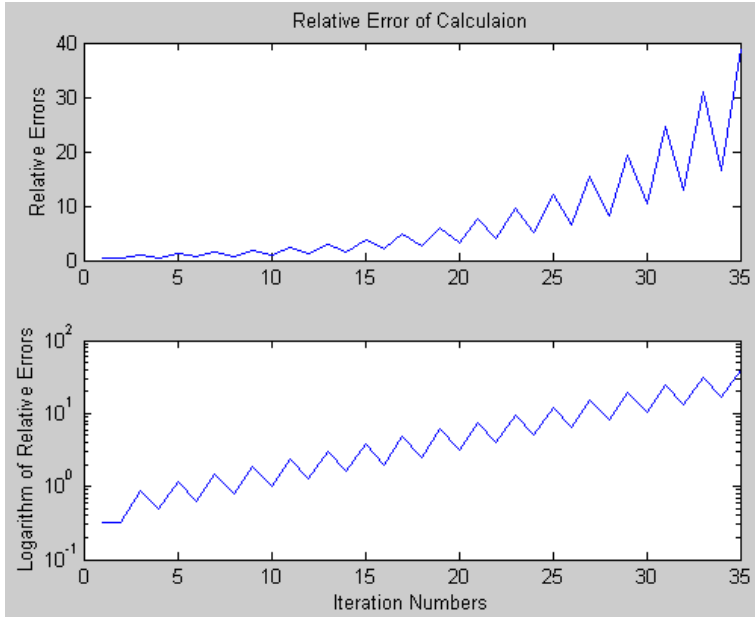

Fig. 3.

Example 8. The program for solution of three equations with complex coefficients in the MATLAB system:

```
function [res,qt,q,erMin,k] = test3r
```

Here the matrix *M* **is not** *of fixed sign,* the process does not converge and is terminated when the error becomes 100 times higher than at the beginning of the iteration. Graphs of the form shown on Fig. 3 are depicted. The user may experiment with matrices of fixed sign or not of fixed sign, namely matrices a and в, indicated in commentaries.

1.6. About speed and accuracy

Experiments show that

- comp ~ maxEr,
- comp is proportional to maxEr,
- the iterations number is proportional to maxEr,
- the iterations number is proportional to the dimension N.

Example 9. Program for solving N equations with complex coefficients in the MATLAB:

```
function testNv
```

Parameter comp serves for precision comparison for this and the traditional algorithms. Fig. 4 shows graphs of the iteration number and the error comp as functions of dimension number N with fixed value maxEr. One can see that, first, comp < maxEr and, second, the iteration number is proportional to the dimension number N.

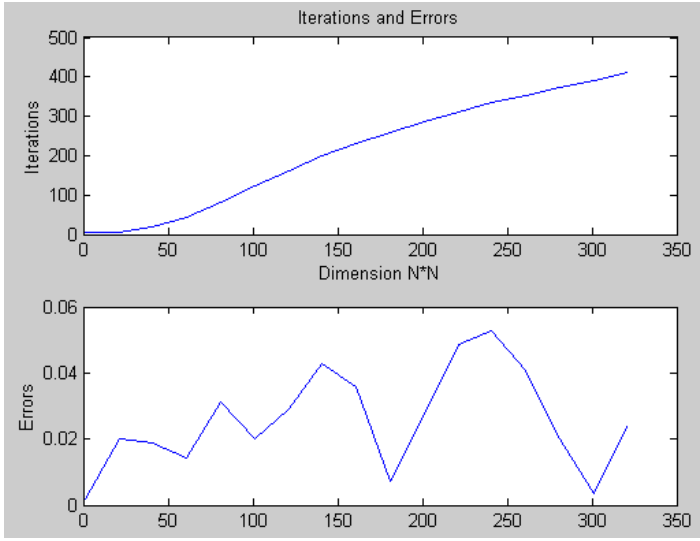

Fig. 4.

Example 10. Program for solving N equations with complex coefficients in the MATLAB:

```
function testNe
```

Parameter comp serves for precision comparison for this and the traditional algorithms. Fig. 5 shows graphs of the iteration number and the error comp as functions of dimension number N with fixed value maxEr. One can see that, first, comp < maxEr and, second, the iteration number is proportional to the dimension number N.

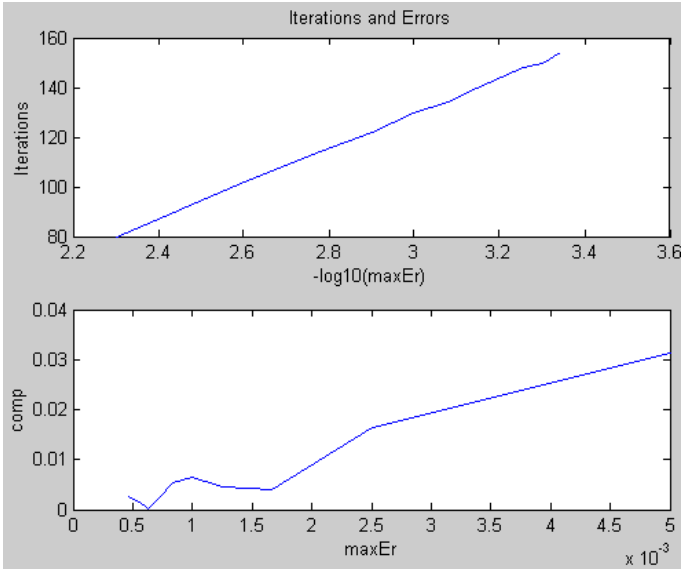

Fig. 5.

Let us consider now the solution of ill-determined systems of the form (1).

1.7. Underdetermined system

In such system the number of equations is less than the number of variables. In this case the system (1) may be complemented by an equation

$$\left(jn^T + m^T\right)x = 0, \tag{4}$$

where n, m – matrices of given weight coefficients. The system (1) is transformed into the system:

$$\begin{vmatrix} a \\ m \end{vmatrix} x + j \begin{vmatrix} b \\ n \end{vmatrix} x = \begin{vmatrix} c \\ 0 \end{vmatrix}. \tag{5}$$

Matrices n, m are going to complement the matrices a, b to squarte matrices $\begin{vmatrix} a \\ m \end{vmatrix}$, $\begin{vmatrix} b \\ n \end{vmatrix}$. The solution of this system correspond to minimization of a weighted sum of squared variables.

The functionals (4) and (5) for relatively large weight coefficients will take the form of the following functions:

$$F_1(x) = x^T m \otimes x, \tag{8}$$

$$F_2(x) = -x^T n \otimes x \quad . \tag{9}$$

These functions correspond to the minimization of weighted sum of variables' squares.

1.9. Overdetermined System

In such system the number of equations is larger than the number of variables. In this case the system (1) may be transformed to the form:

$$\left| a \begin{vmatrix} 0 \\ m \end{vmatrix} \right| \cdot \begin{vmatrix} x \\ y \end{vmatrix} + j \left| b \begin{vmatrix} 0 \\ n \end{vmatrix} \right| \cdot \begin{vmatrix} x \\ y \end{vmatrix} = c \,, \tag{10}$$

where y – the vector of complementary variables n, m – matrices of given weight coefficients of the complementary variables.

matrices n, m are going to complement matrices a, b to squarte matrices $\left| a \begin{vmatrix} 0 \\ m \end{vmatrix} \right|$, $\left| b \begin{vmatrix} 0 \\ n \end{vmatrix} \right|$. The solution of this system correspond to the minimization of a weighted sum of squared residuals.

The functionals (4) and (5) for relatively large weight coefficients will take the form of the following functions

$$F_1(x) = y^T m \otimes y \,, \tag{11}$$

$$F_2(x) = -y^T n \otimes y \quad . \tag{12}$$

These functions correspond to the minimization of weighted sum of residuals' squares.

2. The Second Method for Solution of Linear Algebraic Equations System

2.1. Description of the method

Let us consider now an equation of the form

$$\left(aa^T + bb^T\right)x + (a - jb)c = 0. \qquad (13)$$

Evidently, equation (13) differ from equation (1) by the factor $(a - jb)$. Simultaneously with the solution of system (13) two functions of the following form are being optimized

$$F_1(x) = \left(x^T a^T b + (b - ja)c\right) \otimes x, \qquad (14)$$

$$F_2(x) = \left(x^T \left(aa^T + bb^T\right) + 2(a + jb)c\right) \otimes x. \qquad (15)$$

For well determined system of equations (1), this system, as well as the system (13) have only one solution. For an ill-determined system (1) the solution of system (13) is equivalent to the solution of system (1) with minimization of functions (14) и (15).

Notice, that in equations with real variables and coefficients $b=0$, equation (13) takes the form

$$aa^T x + ac = 0, \qquad (16)$$

And the minimized function (15) will be

$$F_2(x) = \pi\left(x^T aa^T + 2ac\right) \otimes x. \qquad (17)$$

2.2. Main program

M-functions for solving a linear equations system with complex coefficients of the form (a+j*b)*q+c=by the second method are as follows:

```
function [res,erMin,k,q,er] = ...
        SinLin2(E,Z,maxEr,maxK)
```

The parameters used in it are described in Section 1.2.

2.3. Underdetermined System

In such a system the number of equations is less than the number of variables, and there exists a set of solutions. But the solution obtained by the presented method, minimizes the quadratic forms (14) and (15).

Example 12. The program for solution of underdetermined linear equations system with complex coefficients (a+j*b)*q+c=0 in the MATLAB system:

```
function [res,k,q] = testNedo
```

2.4. Overdetermined System

In such a system the number of equation is larger than the number of variables, and the system has no solution. But in our method a solution with a certain residual is found, and it minimizes the quadratic forms (14) and (15).

Example 14. The program for solution of overdetermined linear equations system with complex coefficients of the form (a+j*b)*q+c=0 in the MATLAB system:

```
function [res,k,q] = testPere
```

Thus, to solve the system (1) by the presented method, the latter should be transformed into system (13). This rule is applicable for any system (1) – well-determined, underdetermined or overdetermined.

3. The Method for Solving Linear Equations and Inequalities Systems

3.1. Introduction

In the book [1] the method for solving the problems indicated in the headline of this section is described (in particular). Here we are giving the open codes of the MATLAB program for such computations. The main features of these program are as follows:

1. The main matrix may be singular.
2. The equations and inequalities may be incompatible.
3. The system of equations and inequalities may be underdetermined or overdetermined.
4. A solution with a certain residual is always found (with standard MATLAB the solution may not exist at all, for example if the matrix is singular, or no active inequalities are found, or the system is underdetermined or overdetermined).
5. We are using an iterative method that has no operations of division or matrix inversion.
6. The solution algorithm is suitable for parallelized processing.
7. The problem's size is limited only by the computer's resources.

3.2. Basic Problem

We shall begin with a certain basic problem (using the notations from [1]), and then we shall formulate a number of problems in the terms of the basic problem.

So, we are considering a system of equalities

$$T^T \cdot I + C = n_1 \tag{1}$$

and a system of inequalities

$$T \cdot \varphi + \phi - U \geq n_2, \tag{2}$$

where

$$\phi = R \cdot I, \tag{3}$$

T as a superscript is the sign of transposition,

I, C, U, ϕ, n_1, n_2 - are k -dimensional vectors,

φ - is a m -dimensional vector,

T - is a matrix of dimensions $k \cdot m$,

R - is a diagonal matrix of dimensions $k \cdot k$.

The unknown variables are the vectors I, φ. The values n_1, n_2 are the residuals of corresponding equations appearing in the solution process.

> It is important to note that here and further the inequality sign may belong only to a subset of equations, and in the remaining equations it should be replaced by the sign of strict equality.

In [1] it is shown that the solution of this problem is at the same time the solution of two dual problems of quadratic programming. The first of them is as follows:

$$\begin{aligned} &\frac{1}{2} \cdot I^T \cdot R \cdot I + \frac{\rho}{2} \cdot {n_1}^T \cdot n_1 - U^T \cdot I = \min, \\ &T^T \cdot I + C - n_1 = 0, \\ &I \le 0, \end{aligned} \tag{4}$$

and the second is

$$\begin{aligned} &\frac{1}{2} \phi^T \cdot R^{-1} \cdot \phi + \frac{1}{2\rho} \cdot \varphi^T \cdot \varphi - C^T \cdot \varphi = \min, \\ &T \cdot \varphi + \phi - U - n_2 \ge 0. \end{aligned} \tag{5}$$

Also

$$C^T \cdot \varphi = T \cdot \varphi \cdot I. \tag{6}$$

Besides, according to the complementary slackness condition,

$$I^T \cdot \left(T \cdot \varphi + \phi - U - n_2\right) = 0. \tag{7}$$

Здесь ρ – величина, определяемая пользователем. В [1] показано, что при $\rho \to \infty$ величины n_1, n_2 стремятся к нулю и сагаемое $\frac{\rho}{2} \cdot {n_1}^T \cdot n_1$ в (4) также стремится к нулю.

Here ρ is a user determined value. In [1] it is shown that for $\rho \to \infty$ the values n_1, n_2 tend to zero, and the summand $\frac{\rho}{2} \cdot {n_1}^T \cdot n_1$ in (4) also tends to zero.

The method is based on moving along the gradient of the minimized functions. The computation is performed in iterations, and the computation result will as a rule be an approximate value. So we are getting the above mentioned residuals n_1, n_2. Corresponding to them

are the relative errors $\varepsilon_1, \varepsilon_2$. The value ε_1 is determined by computation, and ε_2 is determined by the user.

The number of iterations (i. e. the duration of computation) and the value of ε_1 are regulated by the value ρ. The larger is the value ρ, the less is ε_1, but the longer is computation time.

3.3. The Solution of Linear Equations System

For this in the basic problem we assume that $R = 0,\ U = 0$. Then (2.4) takes the form

$$T^T \cdot x + C - n_1 = 0, \qquad (1)$$

$$n_1{}^T \cdot n_1 = \min. \qquad (2)$$

The system (1) may be incompatible, underdetermined or overdetermined. We search for a solution corresponding to minimum (2).

M-function for solving this problem is:

```
function [x,eps1,k,n1,mini]=...
    anySLAE2(A,B,r,eps2,kmax)
```

A – the matrix $A = T'$ – see (1),

B – the vector $(-C)$– see (1),

r – the value ρ – see section 2,

eps2 – the value ε_2 – see section 2,

kmax – maximal iterations number,

The output values here are:

x – the vector I – see (2.4, 1),

eps1 – the relative residual value ε_1 – see section 2,

k – iterations number,

n1 – the residuals vector in equation (1),

mini –value of the minimum (2).

The M-functions for test problems looks as :

```
function test_anySLAE2()
```

The test includes the solution for various types of systems. For control the same system is being solved by MATLAB means. The test performs also

```
function [m_x, m_eps1, m_n1,
m_minx, tmatlab] = anySLAE2m(A,B)
```

Here the values similar to the output values of the function **anySLAE2** are computed by traditional methods. Parameter **mode** defines the test's number. Let us consider these tests. .

1. Well determined small dimension system.
2. Underdetermined small dimension system,
3. Overdetermined small dimension system,
4. Overdetermined system for the computation of such vector **x**, with respect to which the components of vector **B** have minimal dispersion.
5. Underdetermined large dimension problem.
6. Overdetermined large dimension problem.
7. Poorly determined small dimension problem. In this case the problem cannot be solved by MATLAB means: a message "Warning: Matrix is singular to working precision" is displayed.

3.4. The Solution of Linear Equations and Inequalities System

The system we are going to solve has the form (see 2.2):

$$T \cdot x - U \geq 0. \qquad (1)$$

In the basic system we assume: $R = 0,\ C = 0.$ System (1) may be incompatible, underdetermined, overdetermined. We are searching for such solution, that corresponds to the minimum for the (2.5) or

$$x^T \cdot x = \min. \qquad (2)$$

The M-function for solving this problem has the form:

```
function [k,x,er2K,n2eq,n2neq,minf]...
   =anySLAE3(A,B,D,r,eps2,kmax)
```

The input arguments here are:

A – is the matrix $A = T$ – see (1),

B – vector (U) – see (1),

D – the equation type indicators vector: if the m-equation is an equality, then $D_m = 0$, in the opposite case $D_m = 1$,

r – the value ρ – see Section 2,

eps2 – the value ε_2 – see Section 2,

kmax – maximal iterations number.

The output arguments here are:

k – iterations number,

x – vector φ – see (2.5, 1),

ep2K – the reached value ε_2 – see Section 2,

n2eq – vector of residuals in the equalities of (1),

n2neq – vector of residuals in the inequalities of the (1), which by the problem's conditions may have any positive value.

minf – the minimum value (2).

The M-function for test problems is:

```
function testqw3
```

The test may solve various versions of the system. In some cases for the sake of control the same system is solved by MATLAB means. The parameter **eq** determines the number of such solution method In the test parameter **mode** determines the test number. Let us consider these tests.

1. Well determined small dimension system. The control is performed according to formula **y=A\B.**
2. Underdetermined small dimension system. The control is performed according to formula **y=A\B.**
3. Overdetermined small dimension system. The control is performed according to formula **y=A\B.**
4. Well determined equalities system **(A,B)**. The control is performed according to formula **y=A\B.**
5. The inequalities system **(A,B)**, coinciding in its left parts with the equalities of the system from p. 4. This system cannot be solved by MATLAB means as a quadratic programming problem A message "`No active inequalities`" is displayed.
6. The inequalities system **(A,B)**, coinciding in its left parts with the equalities of the system from p. 4The equality and inequality signs may be determined by the user in the vector **D**. The control of this problem by MATLAB means is not performed.
7. Underdetermined inequalities system. When trying to solve it by MATLAB means as a quadratic programming problem a message "`A must have 2 column(s)`" is displayed.

8. Overdetermined inequalities system. When trying to solve it by MATLAB means as a quadratic programming problem a message `"A must have 6 column(s)"` is displayed.
9. Overdetermined system of equalities and inequalities (in its left part the same as in p. 8). The vector **D** is such that the obtained solution has large residuals in the equalities.
10. Overdetermined system of equalities and inequalities (in its left part the same as in p. 8). The vector **D** is such that the obtained solution has large residuals in the equalities.

4. A Method for Solving Quadratic Programming Problems

4.1. The First Problem of Quadratic Programming

In the book [1] there was described (in particular) a method for solving the problem of quadratic programming. The following problem is considered.

To find minimum of the function

$$F(x) = 0.5 \cdot x^T \cdot R \cdot x - U^T \cdot x \tag{1}$$

under the restraints

$$T^T \cdot x + C = 0, \tag{2}$$

$$x \geq 0, \tag{3}$$

where

T is a superscript - the transposition sign,

x, C, U - k-dimensional vectors,

T - $k \cdot m$ matrix

R - a $k \cdot k$ square positive definite matrix

The unknown variable here is vector x. The equations set (2) must not be empty. The constraint (3) may relate only to certain variables, or not exist at all. Further in this problem we shall use vector **D** – the equation type indicator in (3): if an m-equation is an equality, then $D_m = 0$, in an opposite case $D_m = 1$.

The solution method is based on movement along the gradient of the minimized function (1). The computation is going in iterations, and the result as a rule is approximate. As a result there appear residuals in the equations (2, 3). The relative residual in the equation (2) is determined as

$$\varepsilon = \left(T^T \cdot x + C\right)^T \left(T^T \cdot x + C\right) \Big/ \left(x^T x\right) \tag{4}$$

The permissible value of this residual $\varepsilon_{\max}$ is given by the user.

The number of iterations (i.e. the computation time) and the value $\varepsilon_{\max}$ are regulated by the value ρ. The larger is the value ρ, the less is the value $\varepsilon_{\max}$, but the longer is computation time.

The M-functions for this problem solution is:

```
function [x,n1,k,ero,Fmin,...
                kk,erok,Fmink,xx]=...
squ2 (R,T,C,U,D,r,erd,kmax)
```

The input arguments here are:

R,T,C,U, D – matrices and vectors defined in (1, 2),

r – the value ρ,

erd – the value $\varepsilon_{\max}$,

kmax – maximal iterations number,

The output arguments here are:

x – the unknown vector,

n1 – the vector of residuals in the equations (2),

k – iterations number,

ero – value of relative residual ε,

Fmin – minimum of the function (1).

The following output values are used for creating the graphs:

kk – the vectors of iterations numbers,

erok - the vector of relative residuals ε in each iteration,

Fmink – the vector of function (1) minimum value on each iteration,

xx - the matrix of **x** vectors on each iteration.

4.2. The Second Problem of Quadratic Programming

Let us consider now the following problem. We are searching for the minimum of function (1)

$$F_1(x) = 0.5 \cdot x_1^T \cdot R \cdot x_1 - U_1^T \cdot x_1 \qquad (5)$$

under the restraints

$$T_1^T \cdot x_1 + C_1 \geq 0. \qquad (6)$$

Here the unknown variable is vector x_1. The set of equations (6) should not be empty. The sign "$\geq$" in (6) may refer only to certain equations, and in the remaining equations it may be replaced by the sign of strict equality. Further in this problem we shall use the vector D_1 –

the equation type indicator in (6): if m-equation is an equality, then $D_{1m} = 0$, and in the opposite case $D_{1m} = 1$.

The second problem may be transformed into first problem in the following way. We shall present the restraint (6) in the form

$$T_1^T \cdot x_1 + C_1 - x_2 = 0, \tag{7}$$

$$x_2 \geq 0. \tag{8}$$

Let us consider the vector

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{9}$$

and rewrite the formulas (5, 7, 8) in the form (1, 2, 3) accordingly, where

$$R = \begin{bmatrix} R_1 & Q_{21} \\ Q_{12} & Q \end{bmatrix}, \tag{10}$$

$$T = \begin{bmatrix} T_1 \\ -E \end{bmatrix}, \tag{11}$$

$$U = \begin{bmatrix} U_1 \\ V_u \end{bmatrix}, \tag{12}$$

$$C = C_1, \tag{13}$$

$$D = \begin{bmatrix} V_d \\ D_1 \end{bmatrix}, \tag{14}$$

E identity matrix,

Q, Q_{12}, Q_{21} - zero vectors,

V_u, V_d - zero vectors

The M-function for the transformation of the second problem into the first problem has the form:

```
function [R,T,U,C,D]=squ21(R1,T1,U1,C1,D1)
```

4.3. Test for the First Problem

The M-function **`function testsqu2()`** solves the first problem for a two-dimensional vector **x**. here we are building the graphs shown on Figure 1, where

- the window `'Error (testsqu2)'` shows the error ε change depending on iteration number ,
- the window `'Log of error'` shows the error $\ln(\varepsilon)$ change depending on iteration number.
- the window `'Minimum'` shows the function (1) change depending on iteration number.

Let us consider the two-dimensional vector **x** as a point on the plane. The window `'Trajectory'` shows the movement of this point with iteration number growth beginning from the initial point (0, 0) to the final point – the problem's solution.

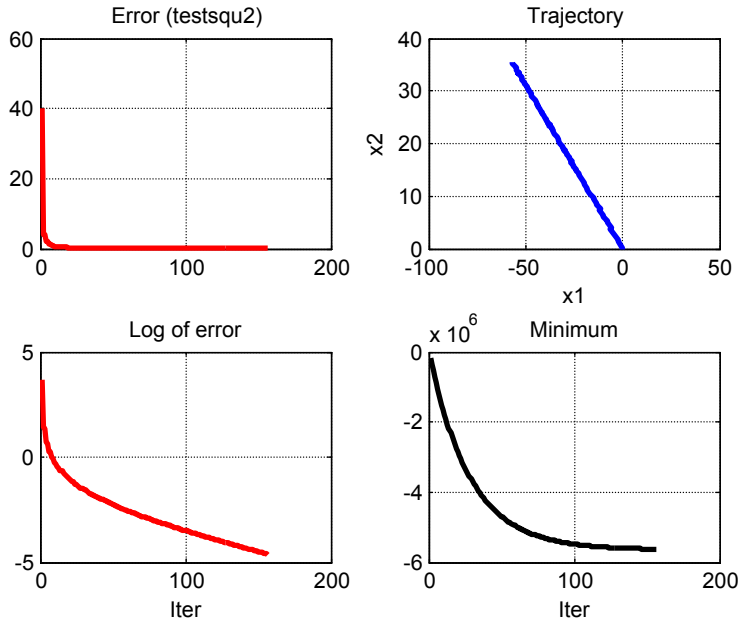

Fig. 1.

4.4. Test for the Second Problem

The M-function **function testsqu4()** solves the second problem for a four-dimensional vector x. Here the graphs shown on Figure 2 are built, where

- the window 'Error (testsqu2)' shows the error ε change depending on iteration number ,
- the window 'Log of error' shows the error $\ln(\varepsilon)$ change depending on the iteration number,
- the window 'Minimum' shows the function (1) change depending on iteration number,
- the window 'Trajectory' shows the change of four components of vector **x** with iteration number growth beginning from the initial value (0) to the final point, that is the problem's solution.

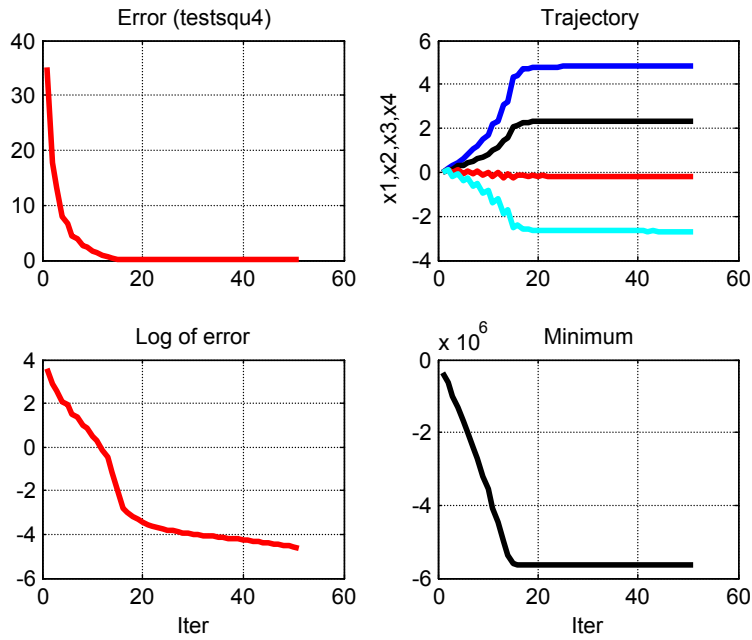

Fig. 2.

5. Algorithms

Here we are considering block diagrams for the algorithms for solving the above enumerated problems. We are using the following notations:

$a \cdot B$ - multiplication (of a vector of a number B) by a number a,

$A \bullet B$ - multiplication of a matrix A by a matrix or a vector B,

$A \otimes B$ - operation of componentwise multiplication of complex vectors A, B and summation of the obtained products,

A' - transposition of a matrix A.

5.1. Algorithm for Solving a Linear Algebraic Equations System

Let u consider the block diagrams of the algorithms for solving a linear algebraic equations system – see Fig. 1-4.

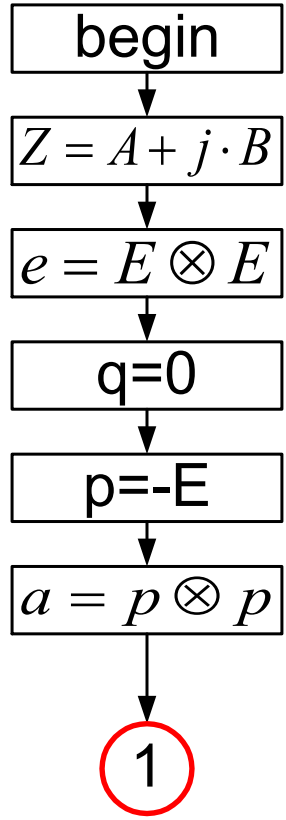

Fig. 1.

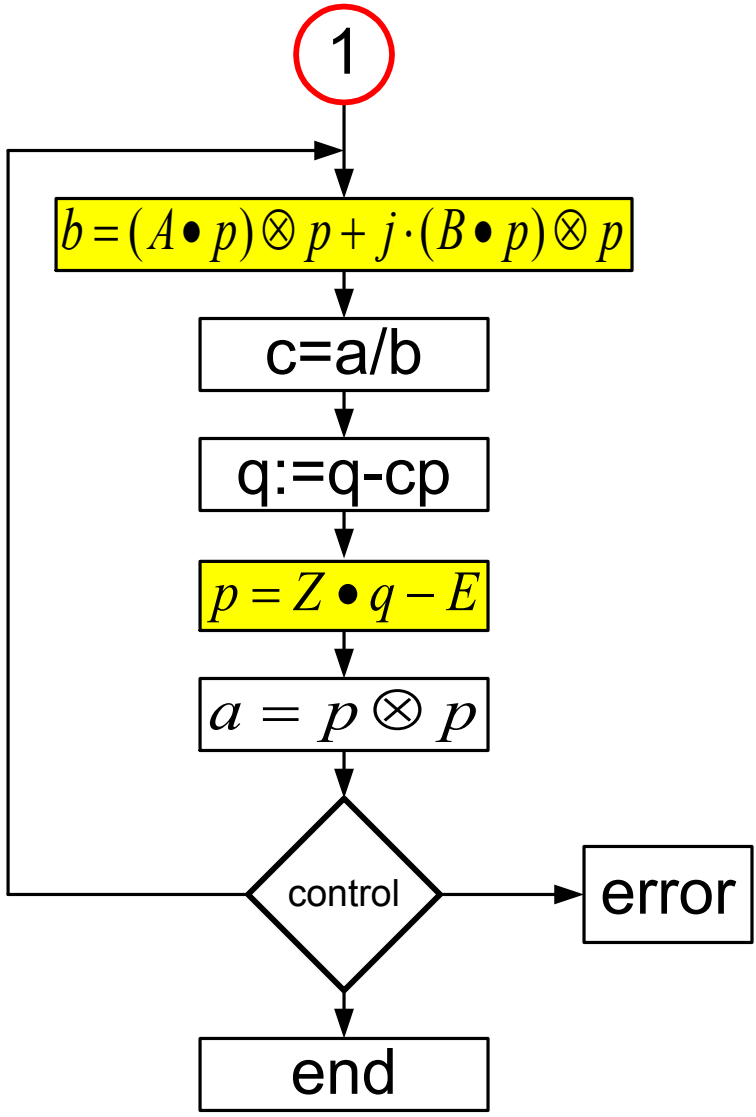

Fig. 2.

Figures 1 and 2 show the block diagram of computation algorithm by the first method a (see also the SinLin function), and Figures 3 and 4 – the block diagram of computation algorithm by the second method (see also the SinLin2 function). Figures 2 and 4 show the blocks realizing the iterations cycle in these algorithms. The differing blocks in these algorithms are shown by different coloring

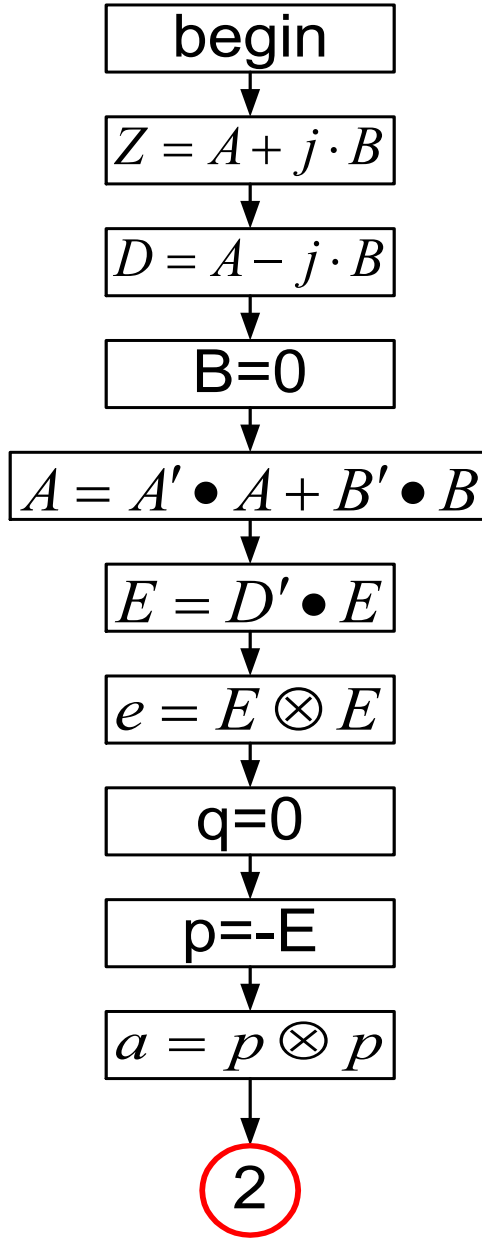
begin
$Z = A + j \cdot B$
$D = A - j \cdot B$
B=0
$A = A' \bullet A + B' \bullet B$
$E = D' \bullet E$
$e = E \otimes E$
q=0
p=-E
$a = p \otimes p$
2

Fig. 3.

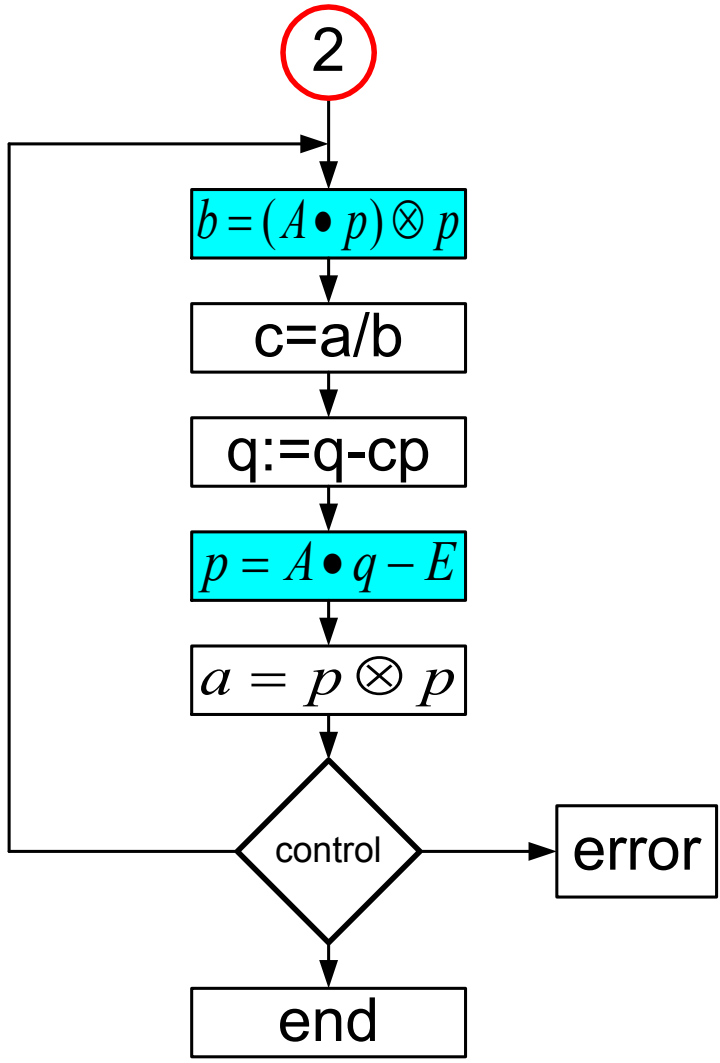

Fig. 4.

The **CONTROL** block in these algorithms on each iteration checks the condition for ending the iteration process - see **`SinLin`** and **`SinLin2`** functions.

The comparison of the two algorithms shows that the second algorithm is more attractive for technical realization, as it contains less operations of matrix by vector multiplication.

5.2. Minimization Algorithms

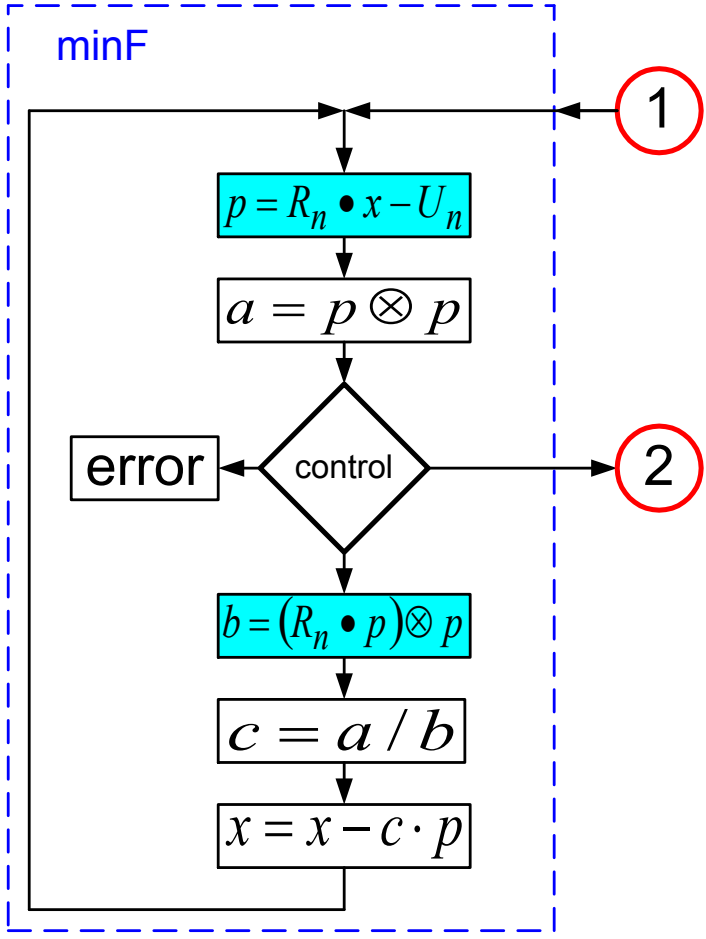

Fig. 5.

In the considered problems of the Sections 3 and 4 a quadratic functional of the following type is being minimized:

$$F(x) = 0.5 \cdot x^T \cdot R_n \cdot x - U_n^T \cdot x \qquad (1)$$

under linear and nonlinear constraints:

$$T^T \cdot x + C = 0, \qquad (2)$$

$$x \geq 0, \qquad (3)$$

where R_n, U_n^T, T are determined by the problems conditions. The minimization is performed by the gradient descent method [4]. Without

constraints (3) the block diagram of the algorithm is of the form shown on Figure 5, where

p – is the functional's gradient,

a, b, c- scalar values.

The iterations process goes on till the value

$$\varepsilon_2 = p \otimes p \tag{4}$$

will not reach the preset value $\varepsilon << 1$. The **CONTROL** block in these algorithms on each iteration checks the condition of iteration process ending, which is

$$\varepsilon_2 / u \le \varepsilon, \tag{5}$$

where

$$u = U_N \otimes U_N, \tag{6}$$

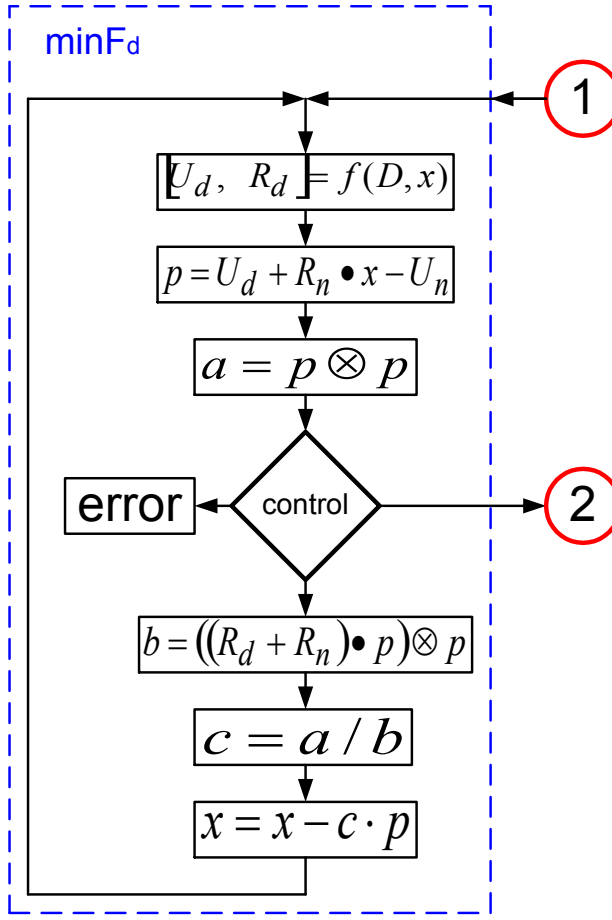

Fig. 6.

If the constraints (3) are present, the block diagram becomes more complicated and take the form shown on Figure 6, where **D** is the vector of indicators for different types of conditions in (3): if $x_m \geq 0$ is present in (3), then $D_m = 1$, in the opposite case $D_m = 0$. The function is determined in M-function **Rdiod.**

5.3. Algorithms for Solving Linear Equations and Inequalities System

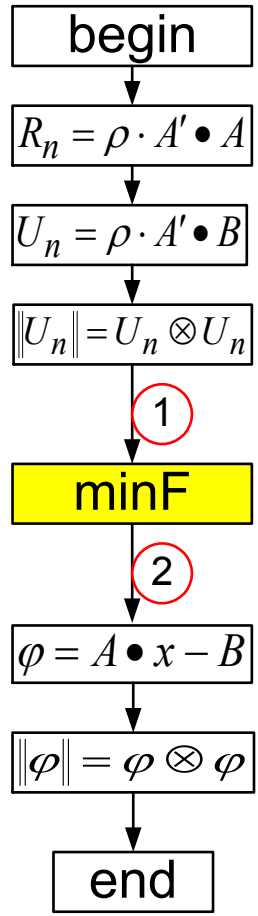

Fig. 7.

Figure 7 presents the block diagram of the algorithm for solving a linear equations system **A*x-B=0**, where the computation error is:

Figure 8 presents the block diagram for solving the linear equations and inequalities system **A*x-B>=0**.

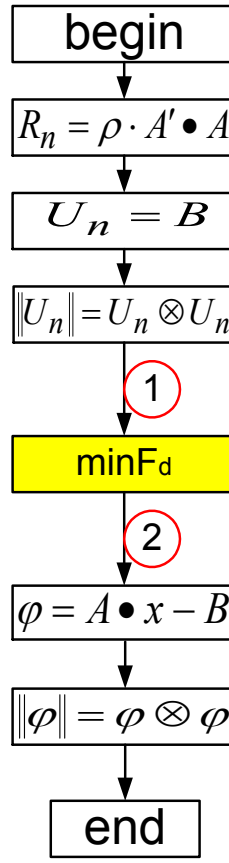

Fig. 8.

5.4. Algorithms for Solving a Quadratic Programming Problem

Figure 9 presents the block diagram of the algorithm for solving a quadratic programming problem considered in Section 4.

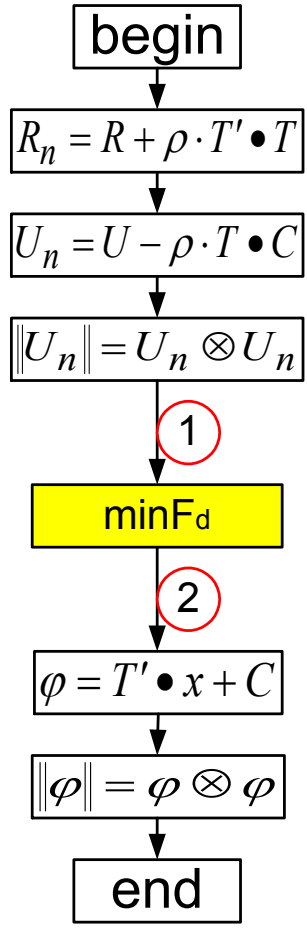

Fig. 9.

5.5. Conclusion

When viewing the presented algorithms, one may note that they contain only the operations of multiplication by a number *a*, multiplication of matrix by matrix or by vector, scalar multiplication of vectors and matrix transposition, Q.E.D. Let us consider now a function $U_d, R_d = f(D, x)$, used in the algorithm on Figure 6 and determined in M-function **Rdiod:**

$$R_{dk} = \begin{bmatrix} \left\{ \begin{matrix} d_{\min}, & if\ x_k > 0, \\ d_{\max}, & if\ x_k \le 0, \end{matrix} \right\} & if\ D_k = 1, \\ d_{\min}, & if\ D_k = 0, \end{bmatrix} \quad (1)$$

$$U_{dk} = R_{dk} \cdot x_k . \qquad (2)$$

Here $d_{\min}$, $d_{\max}$, D_k are constants unchanged from iteration to iteration. So the function (1) may be computed in parallel for all vector x. We can reach additional acceleration if we complement the arithmetic unit by two simple operations:

1. placing the constants $d_{\min}$, $d_{\max}$, D_k into the arithmetic units registers, where they will be kept during all computation process.
2. computing R_{dk}, U_{dk} as functions depending only on x_k.

References

1. Vallach Y. Alternating Sequential/Parallel Processing. Springer-Verlag. Berlin - Haidelberg - New York, 1982, 456p.
2. Shevtsov G.S. Linear Algebra. M.: Gardariki, 1999, 360p. (in Russian).
3. Khmelnik S.I. Variational principle of extremum in electromechanical and electrodynamic Systems, second edition. Publisher by "MiC", printed in USA, Lulu Inc., ID 1142842, Israel, 2010, 355 p., ISBN 978-0-557-08231-5.
4. Khmelnik S.I. Direct Current Electric Circuit for Simulation and Control. Algorithms and Hardware. Published by "MiC" - Mathematics in Computer Comp., printed in USA, Lulu Inc., second edition, ID 293359, Israel-Russia, 2006, 177 p., ISBN 978-1-4452-0994-4 (in Russian).
5. Kumar V., Grama, A., Gupta, A., Karypis, G. (1994). Introduction to Parallel Computing. – The Benjamin/Cummings Publishing Company, Inc. (2nd edn., 2003)
6. Quinn, M. J. (2004). Parallel Programming in C with MPI and OpenMP. – New York, NY: McGraw-Hill.
7. Fox, G.C., Otto, S.W. and Hey, A.J.G. (1987) Matrix Algorithms on a Hypercube I: Matrix Multiplication. Parallel Computing. 4 H. 17-31.

Programs

Contents