ALGORITM OPTIMAL PENTRU PROBLEME DE OPTIMIZARE CU RESTRICŢII SPECIALE

Dr. Eugeniu GÂRLĂ, ASEM
e-mail: eugeniugarla@yahoo.com

*În acest articol, este analizată o clasă de probleme de optimizare neliniară cu restricţii speciale, se propune o metodă directă de rezolvare pentru problema auxiliară, pentru care este calculată complexitatea, se evaluează numărul maxim de operaţii elementare, este descris algoritmul optimal pentru efectuarea calculelor numerice. În studiu, se construieşte un algoritm optimal de rezolvare a problemei auxiliare a modelului **PG**, complexitatea acestui algoritm este **O(nm², N)**, numărul de operaţii elementare este minimal. Matricea supusă inversării nu depinde de dimensiunea problemei n şi are permanent dimensiunea constantă – m × m, m << n. Astfel, modelul PG este absolut funcţional şi, practic, „imun" la dimensiunea problemei de optimizare.*

Cuvinte-cheie: *complexitatea algoritmului, metode de optimizare*
JEL: C0, C6

1. Consideraţii generale

În lucrul cu structuri de date, metode de optimizare, se folosesc modele complexe care presupun un consum foarte mare de resurse pentru rezolvare. Există mai mulţi algoritmi de rezolvare a uneia şi aceleiaşi probleme şi atunci ar trebui să se stabilească algoritmul care este mai performant. Se impune astfel a găsi o măsură a gradului de performanţă sau de eficienţă al algoritmilor. Două criterii stabilesc măsura performanţei unui algoritm: timpul în care se obţine soluţia problemei şi spaţiul de memorie utilizat pentru obţinerea

OPTIMAL ALGORITHM FOR OPTIMIZATION PROBLEMS WITH SPECIAL RESTRICTIONS

PhD Eugeniu GÂRLĂ, ASEM
e-mail: eugeniugarla@yahoo.com

*The present paper analyses a class of nonlinear optimization problems with special restrictions, we propose a direct method for solving the auxiliary problem, for which we calculate complexity, we also assesses the maximum number of elementary operations and describe the optimal algorithm for performing numerical calculations. The study builds an optimal algorithm for solving the auxiliary problem of **PG** model; the complexity of this algorithm is **O(nm², N)**, the number of elementary operations is minimal. Matrix inversion does not depend on the size of problem **n** and always has constant size - m × m, m << n. Thus, the model PG is fully functional and practically "immune" to the size of the optimization problem.*

Key words: *algorithm complexity, optimization methods*
JEL: C0, C6

1. General considerations

Complex models are used when working with data structures and optimization methods; they involve the consumption of a lot of resources. There are several algorithms for solving one and the same problem and then there should be settled an algorithm that would be more efficient. It is therefore necessary to find a measure of performance or efficiency of algorithms. Two criteria establish the measure of performance of an algorithm: the time during which the solution of the problem is found and the memory

ei. Dintre cele două resurse de calcul, spaţiu şi timp, cel mai des, cea critică este timpul de execuţie. Analiza acestor parametri de eficienţă a algoritmilor este cunoscută sub numele de analiza complexităţii algoritmilor. În majoritatea algoritmilor, volumul resurselor necesare depinde de dimensiunea problemei de rezolvat – n. Funcţia f_0, care dă ordinul de mărime al numărului de operaţii elementare şi implicit al timpului de execuţie, se va nota cu $f_0(n)$, iar algoritmul respectiv este notat $O(f_0(n))$. Această notaţie comportă un caracter asimptotic şi determină o clasificare a algoritmilor după ordinul de complexitate. Un algoritm cu $(f_0(n))$, $O(n))$ se numeşte liniar, cu $O(n^2)$ – pătratic, cu $O(n^3)$– cubic, cu $O(n^k)$– polinomial, cu $O(2^n)$– exponenţial. Nici algoritmii exponenţiali, nici cei polinomiali cu grad mare nu pot fi utilizaţi în practică. De aici, algoritmii aplicabili pentru probleme de dimensiuni mari sunt doar cei din clasa $O(n^k)$ ($k << n$), a căror durată de execuţie se consideră rezonabilă. Se cere menţionat că, pentru dimensiunea mare a problemei, performanţa hardware este insuficientă, mai esenţială fiind îmbunătăţirea ordinului algoritmului. La general, nu există nicio formulă universală pentru determinarea complexităţii. Aceasta se face de la caz la caz, ţinând cont de particularităţile problemei. Pentru cele ce urmează: diferiţi algoritmi care soluţionează aceeaşi problemă formează clasa algoritmilor; operaţia cea mai des repetată se numeşte operaţia de bază, iar determinarea teoretică a numărului de repetări permite alegerea celui mai bun algoritm; algoritmul optimal este acel algoritm care efectuează cel mai mic număr de operaţii de bază dintre toţi algoritmii clasei sale; o tehnică cunoscută de

space used to obtain it. Of the two calculation resources, space and time, most often, the runtime is critical. The analysis of these efficiency parameters of the algorithm is known as the complexity of the analysis algorithms. In most algorithms, the amount of resources needed depends on the size of the problem to be solved – n. Function f_0, which gives the size of the number of elementary operations and the execution time, shall be noted as $f_0(n)$, while the respective algorithm is noted as $O(f_0(n))$. This notation has an asymptotic character and determines a classification of algorithms by order of complexity. An algorithm with $(f_0(n))$, $O(n))$ is called linear with $O(n^2)$ – quadratic, with $O(n^3)$ – cubic, with $O(n^k)$ – polynomial with $O(2^n)$ – exponenttial. Neither the exponential algorithms, nor the polynomial with high degree can be used in practice. From here, algorithms applicable to large problems are just of the class $O(n^k)$ ($k << n$), whose execution time is considered reasonable. It is worth mentioning that for the large size of the problem, the hardware performance is insufficient, more essential being the improvement of the algorithm order. Generally, there is no universal formula for determining the complexity. This is done from case to case, taking into account the peculiarities of the problem. For the following: different algorithms that solve the same problem form the class of algorithms; the most often repeated operation is called basic operation; the theoretical determination of the number of repetitions allows selecting the best algorithm; the optimal algorithm is the one that performs the smallest number of basic operations of all algorithms for its class; a well-known technique in solving recurrence

rezolvare a recurenţei complexităţii este metoda iteraţiei, care transformă recurenţa în sumă de termeni dependenţi doar de n şi condiţiile iniţiale, iar suma, la rându-i, se obţine, deseori, prin operaţia de logaritmare; sunt utile relaţiile:

iteration complexity is the method of iteration, that converts recurrence in the sum of terms dependent only on *n* and initial conditions, while the sum is obtained through the logarithm operation; in this respect the following relations are useful:

$$\lim_{n\to\infty}\frac{n^k}{2^n}=0;\ \ \mathrm{O}(n^k)\subset\mathrm{O}(2^n),$$

la calcularea complexităţii modelului contează doar ordinul superior al operaţiilor efectuate în procesul algoritmizării, deci, se reţine doar termenul care creşte cel mai repede odată cu creşterea lui n, deoarece acest termen are impactul cel mai mare asupra timpului de execuţie sau al spaţiului ocupat al implementărilor algoritmului, ceilalţi termeni devenind neglijabili pentru valori mari ale lui n; pentru operaţia de multiplicare a două matrice pătrate, complexitatea este $O\left(n^3\right)$, respectiv, pentru înmulţirea a două matrice dreptunghiulare (una *m* × *l*, alta *l*× *n)*, este $O\left(mln\right)$; inversarea matricială nu este mai complexă din punct de vedere al numărului de operaţii decât înmulţirea de matrice; frecvent se cere parantezarea optimală a produsului de matrice $A_1\times A_2\times...\times A_n$ de dimensiuni $d_0\times d_1,\ d_1\times d_2,\ ...,d_{n-1}\times d_n$ pentru care numărul total de operaţii elementare să fie minim, în particular, parantezarea optimală se rezolvă pornind de la dreapta spre stânga: fie $A_1(n\times m),\ A_2(m\times m),\ A_3(m\times n), A_4(n\times 1)$, atunci, pentru produsul $A_3(m\times n)\ \Pi, A_4(n\times 1)$, se vor efectua înmulţiri şi adunări respectiv

while calculating the model complexity only higher order operations counts, that are carried out in the process of algorithmisation, therefore having retained only the term which has the fastest growth together with the growth of *n*, because this term has the largest impact on execution time or space occupied in the implementation of the algorithm, the other terms becoming negligible for larger values of *n*; for the operation of multiplication of two square matrices, the complexity is $O\left(n^3\right)$, respectively, for the multiplication of two rectangular matrix (one *m* × l, another *l*×*n*), is $O\left(mln\right)$; matrix inversion is not more complex in terms of the number of operations than multiplication of matrices; frequently requires bracketing of optimal product matrix $A_1\times A_2\times...\times A_n$ of size $d_0\times d_1,\ d_1\times d_2,\ ...,d_{n-1}\times d_n$ for the total number of elementary operations to be minimal, in particular, bracketing optimal solved from right to left: let, $A_1(n\times m),\ A_2(m\times m),\ A_3(m\times n),\ A_4(n\times 1)$ then the product $A_3(m\times n)\ \Pi, A_4(n\times 1)$, will perform multiplication and congregation like

$$\rightarrow 2\times n\times m-m,$$

pentru produsul $A_2(m\times m)\ \Pi\ A_3(m\times n)\ \Pi\ A_4(n\times 1)$, se vor efectua înmulţiri şi adunări respectiv $\rightarrow\ 2\times n\times m-m+2\times m\times m-m,$

for product $A_2(m\times m)\ \Pi\ A_3(m\times n)\ \Pi\ A_4(n\times 1)$, will perform multiplication and gatherings that $\rightarrow\ 2\times n\times m-m+2\times m\times m-m,$

în rezultat per total, pentru produsul

resulting in overall product for product

$$A_1(n\times m)\ \Pi\ A_2(m\times m)\ \Pi\ A_3(m\times n)\ \Pi$$

se vor efectua înmulţiri şi adunări respectiv

will perform multiplication and sums respectively

$$\rightarrow\ 2\times n\times m-m+\ 2\times m\times m-m+2\times n\times m-n\,.$$

2. Modelul de optimizare

Problemele de optimizare neliniară, în control optimal şi economie, sunt probleme de valori extreme cu condiţii suplimentare, caracterizate prin aceea că numărul variabilelor este foarte mare. Este binecunoscut faptul că majoritatea modelelor de optimizare operează cu inversa pentru unele matrice. Dacă inversa s-ar afla, atunci soluţia s-ar găsi în urma unor operaţii matematice elementare, dar efortul de calcul necesar inversării este foarte mare, există enorme probleme de stabilitate, totodată, nu există soluţia în formă analitică, de aceea, majoritatea metodelor sunt metode iterative, adică de la iteraţie la iteraţie se construieşte consecutivitatea

2. The optimization model

Nonlinear optimization problems, as well as those in optimal control and economy are matters of extreme values with additional conditions, characterized in that the number of variables is very high. It is well known that most optimization models operate with the reverse for some matrix. If the reverse is found out, then the solution would find the following basic mathematical operations, but the calculation effort required for reverse is very high; there is enormous stability issues, while there is no solution in analytical form, therefore, most methods are iterative methods, i.e. iteration to iteration builds the following sequence

$$f\left(x^0\right)\geq,\ldots,\geq\ f\left(x^{k-1}\right)\geq\ f\left(x^k\right)\geq\ f\left(x^{k+1}\right),\geq\ldots,$$

$$x^{k+1}=x^k+\alpha_k p_k\ ,$$

unde α_k – număr, p_k – vector,

iar procesul de calcul se încheie, dacă diferenţa dintre două valori consecutive ale lui x satisfac inegalitatea $\left|x^{k+1}-x^k\right|<\varepsilon$, ε – număr mic dat, iar iteraţia aici incluzând rezolvarea problemei auxiliare – problemă-cheie, de care depinde esenţial şi eficienţa problemei iniţiale. Pornind de la aceste considerente, cercetările făcute de autor [1] s-au axat pe ideea selectării claselor de probleme de dimensiuni mari, care determină nişte cazuri particulare, dar care acoperă o mare parte din problemele practice şi pentru care, totodată, pot fi propuse scheme eficiente de calcul numeric. Fie problema iniţială de optimizare neliniară în forma (modelul *PG*)

where α_k – number, p_k – vector,

and the calculation process ends, if the difference between two consecutive values of x satisfy the inequality $\left|x^{k+1}-x^k\right|<\varepsilon$, ε – given small number and iteration here including ancillary problem solving – key issue, and effectiveness which depends heavily on the initial problem. Based on these considerations, the research made by the author [1] focused on the idea of selecting classes of large problems that causes some cases, but covering much of the practical problems and which also may be the proposed effective numerical computation schemes. Whether the initial nonlinear optimization problem in the form (model *PG*)

$$min\ f(x)\ (1)$$
$$\begin{cases} g_j(x) = 0,\ j = \overline{1,m}.....(2) \\ 0 \le x_i \le a_i, \mathrm{i} = 1,n.....(3), \end{cases}$$

problema auxiliară (*PA*) în model este o problemă de programare pătratică cu matrice unitară

the auxiliary problem (*PA*) in the model is a quadratic programming problem with unitary matrix

$$min\left(f_1(p) = 1/2\|p\|^2 + (d,p)\right) \quad (4)$$

$$Hp = h, H = \begin{bmatrix} h_{11}.....h_{1n} \\ ..... \\ h_{m1}.....h_{mn} \end{bmatrix} \quad (5)$$

$$0 \le p_i \le a_i, i = 1,\ldots,n. \quad (6)$$

unde $f(x), g_j(x)$, – funcţii de x, continue, împreună cu derivatele lor, *d,h* – vectori, *H* – forma matricială a restricţiilor Jacobianului *(2),* în care $h_j(x) \equiv g_j'(x)$, însemnând derivata, iar componenta *i* a gradientului $h_j(x)$, notată respectiv cu – h_{ji}. Nu se distinge aici separat cazul unui sistem de ecuaţii neliniare, la fel nu se abordează în detalii condiţiile necesare de extremum, convergenţa modelului *PG,* ca atare, stabilitatea soluţiei, problema acumulării erorilor, tehnicile computeriale etc. (pentru rezultate teoretice şi practice a se vedea, de exemplu [3;1]). În continuare, se examinează doar complexitatea modelului, în speţă a celei mai dificile operaţii – complexitatea problemei auxiliare, celelalte operaţii ale modelului nefiind esenţiale ca timp şi spaţiu. Trebuie să se remarcate că, în *(2),* se presupune $m << n$, altfel spus, restricţii de tip " =" sunt foarte puţine, adevărata dimensiune mare a problemei depinzând esenţial de n. Pentru *(2)-(3)* se defineşte operatorul de proiecţie:

where $f(x), g_j(x)$ – functions of *x*, continue along with their derivatives, *d, h* – vectors, H – matrix form of Jacobean restrictions *(2)* where $h_j(x) \equiv g_j'(x)$ meaning the derivative and the gradient component *i* of gradient $h_j(x)$, denoted respectively with h_{ji}. We cannot separately distinguish here a system of linear equations, the same does not address in detail the conditions necessary of extremum, *PG* convergence model, as such, the stability of the solution, the problem of accumulation of errors, computer techniques etc. (for theoretical and practical results see, for example [3, 1]). Next, we shall examine the complexity of the model and in case of the most difficult operations – Auxiliary complexity of the problem, other operations of the model is not essential as time and space. It should be noted that in *(2),* it is assumed that $m << n$, in other words, restrictions type "=" are very few true size of the problem depends essentially on *n*. For *(2)* – *(3)* is defined the projection operator:

$$P = G^T\left(GG^T\right)^{-1}G,$$

care alcătuieşte matricea de proiectare din restricţii după un algoritm stabilit. Operatorul

which forms the matrix design of restrictions set by an algorithm. The operator *P* has a

P are o serie de proprietăţi remarcabile, fapt pentru care, deseori, constituie metoda de bază a rezolvării problemei auxiliare. Dar în cazul problemelor de dimensiuni mari examinate aici, dacă s-ar proceda frontal, atunci în G s-ar include şi restricţiile paralelipipedice de tip „≤„ fapt ce-ar îngreuna la maximum inversarea lui $\left(GG^T\right)^{-1}$, matrice de dimensiuni foarte mari $\left(m+2\times n, m+2\times n\right)$. Pentru operatorul de proiecţie al modelului *PG*, însă, s-a reuşit reducerea inversării la calculul matricei

number of remarkable properties, for which often constitutes the basic method of auxiliary problem solving. But if large issues examined here, if you would proceed front, then it would include restrictions rectangular type " ≤" which would make it more difficult to reverse of $\left(GG^T\right)^{-1}$ than his very large matrix $\left(m+2\times n, m+2\times n\right)$. Operator *PG* projection model, however, has managed to reduce to calculation matrix inversion

$$R^{-1} = \left(HH^T\right)^{-1},$$

unde H este compusă doar din restricţii de tip " =" – matrice bandă, iar restricţiile de tip „≤„ se iau H în considerare adăugând în H o coloană după o formulă, sau scoţând din H o coloană conform unei formule similare, altfel spus, s-a demonstrat că luarea în considerare a unei restricţii în procedura de optimizare echivalează cu adăugarea unei coloane sau cu scoaterea coloanei respective din matricea restricţiilor de tip "=". Pentru problema auxiliară, s-a demonstrat că la pasul k există: întotdeauna o <u>direcţie</u> p nenulă:

where *H* is composed only of restrictions type "=" – matrix tape and restrictions of "≤" are *H* considered adding to *H* a column after the formula, or taking out from *H* a column according to a formula similar, in other words, it was shown that consideration of a restriction optimization procedure is equivalent to a column or removing restrictions that column matrix type "=". For the auxiliary problem, it has been demonstrated that in step *k* there is:
always a non-null *p* <u>direction</u>:

$$p^{k+1} = p^k - \alpha^k \left(I-P\right)\left(p^k + d\right)$$

- formulă de calcul al <u>multiplicatorilor Lagrange</u>:

- calculation formula of <u>Lagrange multipliers</u>:

$$\lambda^{m+j}{}_k = \left\{ p^k + d \right\}_i + \left(H^T{}_j,\ \lambda_k\right)$$

- soluţie după un număr <u>finit</u> de paşi ≤ *N*, *N* – constantă
- număr α^k, găsit după o <u>formulă analitică</u>
- metodă de <u>inversare</u> a $\left(HH^T\right)$.

Pentru evitarea calculării excesive a $\left(HH^T\right)^{-1}$, în procesele iterative, se aplică frecvent formula generală de recurenţă a lui Woodbury, sau cazul particular al acesteia –

- solution after a <u>finite</u> number of steps ≤ *N*, *N* – constant
- number α^k, found after an <u>analytical formula</u>
- method of <u>reverse</u> of $\left(HH^T\right)$.

In order to avoid calculating excessive $\left(HH^T\right)^{-1}$, in iterative processes frequently applied a common recurrence formula of Woodbury, or the particular case of it -

formula Sherman-Morrison cu modificaţii, care permit calcularea inversei noii matrice, la care s-a adăugat o coloană (vectorul H_j) şi un rând (vectorul $H^T{}_j$), fără a purcede la inversarea acesteia, ci folosind matricea iniţială R^{-1} şi operaţia produsului vectorial.

3. Rigurozităţi matematice

Rigurozităţile modelului de optimizare au la bază următoarele considerente.

3.1. Fie $p^0{}_k$ punctul iniţial, care satisface condiţiile *(5) – (6)* şi h_j liniar independenţi. Având în vedere specificul restricţiilor problemei iniţiale se presupune că un astfel de punct $p^0{}_k$ de fiecare dată există, mai mult decât atât, deoarece *n* >> *m*, la fiecare pas se pot lua în considerare toate *m* restricţii de tip "=", iar cele de tipul „≤„ se iau în evidenţă conform algoritmului descris mai sus. După cum s-a menţionat, matricea *G* va conţine, în rândurile corespondente restricţiilor de tipul "=", doar ±1 sau 0, ceea ce sugerează idei de simplificare la calcularea operatorului de proiecţie *P*. Cu adevărat, dacă, de exemplu, o componentă $p_j{}^0{}_k$ a vectorului $p^0{}_k$ iese la suprafaţă, adică $p_j{}^0{}_k = 0$, bunăoară (aici şi în continuare, cazul $p_j{}^0{}_k = a_i$, se examinează analogic), atunci aceasta **echivalează** cu fixarea coloanei respective *j* în matricea *H*, corespunzătoare restricţiilor de tipul "=".

formula Sherman-Morrison with modifications that allow the calculation of the inverse of the new matrix to which was added a column (vector H_j) and a row (vector $H^T{}_j$) without proceeding to its reverse, but using the original matrix R^{-1} and vector product operation.

3. Mathematical rigors

Stringencies of the optimization model are based on the following considerations.

3.1. Let $p^0{}_k$ be the starting point, satisfying the conditions *(5) – (6)* and h_j linearly independent. Considering the specific restrictions of original problem, supposedly such a point $p^0{}_k$ every time there is more than that, since n >> m, every step can consider all *m* restrictions of "=" and those of type "≤" are taken out according to the algorithm described above. As mentioned above, the matrix *G* will contain in the corresponding rows restrictions of the type "=" only ± 1 or 0, which suggests ideas to simplify the calculation of projection operator *P*. Indeed, if, for example, a component $p_j{}^0{}_k$ of vector $p^0{}_k$ comes out, that is $p_j{}^0{}_k = 0$, for example (here and hereinafter, the case $p_j{}^0{}_k = a_i$ is examined analogue), then this is **equivalent** to determining that column *j* of the matrix *H*, corresponding to restrictions such as "=".

$$\begin{bmatrix} h_{11}.....0.....h_{1n} \\ .....0..... \\ h_{m1}.....0.....h_{mn} \end{bmatrix} = H \qquad (7)$$

Pentru aceasta, este suficient să comparăm operatorii de proiecţie respectivi P_L, $P_{L\text{-}1}$. $P_{L\text{-}1}$ corespunde matricei H_{L-1}=$(H_{L,1},\ldots, H_{L,j-1}, H_{L,j+1},\ldots, H_{L,n})$, în care lipseşte exact coloana *j*, iar P_L matricei *(5)* şi care va avea coloana respectivă

For this reason it is sufficient to compare the respective projection operators P_L, $P_{L\text{-}1}$. $P_{L\text{-}1}$ corresponding to matrix H_{L-1}=$(H_{L,1},\ldots, H_{L,j-1}, H_{L,j+1},\ldots, H_{L,n})$, where it misses column *j*, while P_L of matrix (5) and that will have the respective column

alcătuită din 0. La fel şi întreaga (*PA*) arată respectiv

composed of 0. Similarly, the entire *(PA)* looks as follows

$$min\ \left(1/2\ \|\left(z-p_k\right)\|^2 : H_L p_k = h\right)$$
$$min\ \left(1/2\ \|\left(z-p_k\right)\|^2 : H_{L-1} p_k = h\right).$$

Altfel spus, se efectuează succesiv trecerea de la o problemă de minimizare la alta cu o unitate mai mică a dimensiunii (subspaţiu), adică trecerea restricţiei în rândul celor pasive a echivalat cu apariţia coloanei din 0 în *(7)*. Având P_L calculat în acest mod şi micşorând cu mult dimensiunea matricei iniţiale se poate găsi un nou punct

In other words, sequentially is performed the switch from one problem to another with a smaller unit size (subspace), that is passage of restriction among those liabilities amounted to 0 in appearance of the column (7). With P_L calculated in this way and reducing a lot the initial matrix size can find a new point

$$p^0{}_{k+1} = p^0{}_k - \alpha^o{}_k (I - P_L) f_1^{'}\left(p^0{}_k\right) \qquad (8)$$

care este desemnat drept iniţial şi cu care se procedează analogic. Situaţia se schimbă, dacă, la un moment dat, o deplasare din punctul iniţial este imposibilă, cu alte cuvinte, s-a atins *min.* Atunci se pot formula următoarele condiţii necesare pentru extremum, referitor la problema *(4) – (6)*, pe componente

that is designated as the original and proceed with the analogue. The situation changes if, at a given moment, a shift of the starting point is not possible, in other words, it has reached *min*. Then the following conditions can be formulated for the extremum, on the problem *(4) – (6)*, on components

$$\left[f_1^{'}\left\{(p^o{}_k)\right\}^j \right] = -\left(H_{L,j}{}^T, \lambda_k\right) - \ldots - \lambda_k{}^{m+j}\{-1\}\ , \qquad (9)$$

$\lambda^i{}_k$ – multiplicatorii Lagrange sunt împărţiţi în două grupuri: $\lambda_k = (\lambda^1{}_k \ldots \lambda^m{}_k)$, cei ce corespund restricţiilor de tip "=", pentru care semnul nu contează şi restricţiilor „≤ „ şi cei care se deduc din *(9)* după <u>formula</u>:

$\lambda^i{}_k$ – Lagrange multipliers are divided into two groups: $\lambda_k = (\lambda^1{}_k \ldots \lambda^m{}_k)$, those corresponding to restrictions of type "=", to which the sign does not matter and restrictions "≤" and those which are deducted from *(9)* by the <u>formula:</u>

$$\lambda^{m+j}{}_k = \left\{ f_1^{'}\left(p^0{}_k\right)\right\}^j + \left(H_{L,j}{}^T,\ \lambda_k\right). \qquad (10)$$

Dacă toţi $\lambda^{m+j}{}_k$ sunt nenegativi, atunci s-a găsit soluţia pentru *(4)- (6)*, în caz contrar, dacă cel puţin unul din aceştia, de exemplu $\lambda^{m+j}{}_k$, care corespunde $p_j{}^0{}_k = 0$, este negativ, atunci se efectuează operaţia inversă, în matricea *H* se readuce coloana *j* şi se construieşte deja operatorul de proiecţie P_{L+1},

If all $\lambda^{m+j}{}_k$ are non-negative, then we found the solution for *(4) – (6)*, otherwise, if at least one of them, for example $\lambda^{m+j}{}_k$, which corresponds to $p_j{}^0{}_k = 0$ is negative, then is performed the inverse operation, in matrix *H* is brought back the column *j* and projection operator P_{L+1} is build, following the same

urmând aceeaşi procedură de calcul a *min* în noul subspaţiu.

3.2. Este uşor de dedus că $\alpha^o{}_k$ din *(4)-(6)* are următoarea **formulă analitică** de calcul

procedure for calculating the new subspace *min*.

3.2. It is easy to deduce that $\alpha^o{}_k$ from *(4-(6)* has the following **analytical formula**

$$\alpha^o{}_k = min\left(p_j{}^0{}_k \,/\, \left(\left(I - P_L\right) f_1^{'}\left(p^0{}_k\right)\right)^j > 0, \left(p_j{}^0{}_k - a_j \right) / \left(\left(I - P_L\right) f_1^{'}\left(p^0{}_k\right)\right)^j < 0,1\right), \qquad (11)$$

parcurgând toţi $j = \overline{1,n}$. Evident că cel puţin pentru componenta *j*, pentru care $\lambda^{m+j}{}_k < 0$ a vectorului $\left(I - P_L\right) f_1^{'}(p^0{}_k)$, este posibilă o direcţie nenulă, deoarece produsul componentei *j* a vectorului unitar $e = \left(0,\ldots 1..,0\right)$ cu această componentă va fi: $-\left(\left(I - P_L\right) f_1^{'}\left(p^0{}_k\right)^j = \; - \; \lambda^{m+j}{}_k > 0\right.$.

Suplimentar, se observă uşor că $\alpha^o{}_k > 0$, deoarece valoarea $\alpha^o{}_k = 0$ a fost exclusă pentru componenta pentru care $p_j{}^0{}_k = 0$.

3.3. Fie *J(p⁰ₖ)* mulţimea de indici pasivi. În noul punct $p^0{}_{k+1}$, valoarea funcţionalei f_1 strict va descreşte, iar mulţimea indicilor $J(p^0{}_{k+1})$ va include strict mulţimea $J(p^0{}_k)$. Aceasta înseamnă că *min*, în noul subspaţiu, va fi atins în mod obligatoriu, deoarece mulţimea *J* nu se poate lărgi la infinit, ceea ce demonstrează că, **cel mult în** N **paşi**, procesul de lărgire a mulţimii de indici se va întrerupe. Astfel, se observă lejer că schema generală de rezolvare conţine o singură operaţie matematică complicată şi voluminoasă în calcule numerice – alcătuirea operatorului de proiecţie P_L, în care intră operaţia de inversare a matricei $\left(H_L H^T{}_L\right)^{-1}$. Pornind de la specificul restricţiilor problemei iniţiale, schema generală de rezolvare poate fi eficientizată în continuare, în primul rând, prin evitarea operaţiei de inversare a matricei $\left(H_L H^T{}_L\right)^{-1}$ la fiecare iteraţie, utilizând formule de recurenţă din algebra liniară. Pentru aceasta, este suficient de reprezentat R_L sub forma

going through all $j = \overline{1,n}$. Obviously, at least for the component *j*, where $\lambda^{m+j}{}_k < 0$ of the vector $\left(I - P_L\right) f_1^{'}(p^0{}_{k)})$ is possibly a non-zero direction because the product of component *j* of the unit vector $e = \left(0,\ldots 1..,0\right)$ with this component will be: $-\left(\left(I - P_L\right) f_1^{'}\left(p^0{}_k\right)^j = \; - \; \lambda^{m+j}{}_k > 0\right.$.

Additionally, it is easy to notice that $\alpha^o{}_k > 0$, as the value $\alpha^o{}_k = 0$ was excluded for component for which $p_j{}^0{}_k = 0$.

3.3. Let $J(p^0{}_k)$ be the crowd of passive indices. The new point $p^0{}_{k+1}$, the functional value f_1 will strictly decrease, and the crowd indices $J(p^0{}_{k+1})$ will include strict set $J(p^0{}_k)$. This means that *min* in the new subspace will necessarily be achieved, as set *J* cannot expand indefinitely, which demonstrates that in **at most** N **steps** the process of set enlarging of indices will pause. Thus, there is a general scheme for resolving a single mathematical operation which is complicated and full of numerical calculations – P_L projection operator composition into that contains the matrix inversion operation $\left(H_L H^T{}_L\right)^{-1}$. Starting from the specific restrictions of the initial problem, the general scheme of solving the problem can be streamlined further, primarily by avoiding matrix inversion $\left(H_L H^T{}_L\right)^{-1}$ at each iteration operation, using recursion formulas from linear algebra. For this, it is sufficient to represented R_L in the form

$$R_L = H_{L,1}H^T{}_{L,1} + \ldots + H_{L,n}H^T{}_{L,n},$$

de unde rezultă imediat că orice permutare a coloanelor în R_L nu schimbă această matrice. fie acum că se adăugă o coloană în H_L, anume $H_{L,j}$, notat $H_{L+1} = \left(H_L,, H_{L,j}\right)$. Atunci

from which results immediately that any permutation of columns in R_L does not change this matrix. Event that now is added a column in H_L, namely $H_{L,j}$, noted $H_{L+1} = \left(H_L,, H_{L,j}\right)$. Then

$$R_{L+1} = H_L H^T{}_L + H_{L,j}H^T{}_{L,j}.$$

Aplicând acum pentru cazul particular R_{L+1} formula generală se obţine:

So, applying for the particular case R_{L+1} the general formula, is obtained:

$$R^{-1}{}_L = R^{-1}{}_{L+1} - (1 + H^T{}_{L,j}R^{-1}{}_{L+1}H_{L,j})^{-1}R^{-1}{}_{L+1}H_{L,j}H^T{}_{L,j}R^{-1}{}_{L+1}.$$

Analogic, la scăderea din H_{L+1}, a coloanei $H_{L,j}$:

Analogically, decreasing from of H_{L+1} the column $H_{L,j}$:

$$R^{-1}{}_L = R^{-1}{}_{L+1} + (1 - H^T{}_{L,j}R^{-1}{}_{L+1}H_{L,j})^{-1}R^{-1}{}_{L+1}H_{L,j}H^T{}_{L,j}R^{-1}{}_{L+1}.$$

Ca rezultat, excelenta formulă recurentă din algebra liniară îşi găseşte locul cuvenit în schema de optimizare sporind eficienţa algoritmului de minimizare şi evitând operaţia de inversare. Atunci însă, când inversarea este inevitabilă, bunăoară, la prima iteraţie, sau la acumularea erorii de calcul după un anumit număr de paşi, în ipoteza că funcţiile ce figurează sunt derivabile până la ordinul doi inclusiv, fapt ce implică practic şi simetria matricei $R_L = R^T{}_L$, schema de optimizare poate fi dezvoltată în continuare. Pentru astfel de matrice, în algebra liniară există metode de inversare şi mai eficiente, cu economii esenţiale de memorie operativă şi volum de operaţii elementare. De exemplu, metoda lui Cholesky permite descompunerea R_L în produs de două matrice triunghiulare, una superior triunghiulară şi alta inferior triunghiulară $R_L = S_L S^T{}_L$, cu inversarea prin factorizare a uneia din ele.

As a result, the excellent formula recurring from linear algebra finds its rightful place in the scheme of increasing efficiency optimization algorithm for minimizing and avoiding reversal operation. Then, when the reversal is inevitable, for example, at the first iteration, or at the accumulation of error calculation after a certain number of steps, assuming that the functions contained are differentiable up to order two inclusive, which basically involves matrix symmetry $R_L = R^T{}_L$, the optimization scheme can be further developed. For such a matrix, in linear algebra, there are reversal methods even more efficient, with essential memory savings and volume of elementary operations. For example, the Cholesky method of decomposition R_L in product of two triangular matrices, one upper and one lower triangular $R_L = S_L S^T{}_L$ reversing through factorization of one of them.

3.4. Foarte des în model se formulează problema de extremizare pe subspaţiu în forma *(4)-(5)*. Minimumul pentru *(4)-(5)* se obţine **dintr-un pas**. Într-adevăr, dacă punem

3.4. Very often in the model is formulated a problem of extremity in the form of *(4)-(5)*. The minimum for *(4)-(5)* is obtained in **one step**. Indeed, if we have

$P = H^T\left(HH^T\right)^{-1}H, x=\left(I - P\right)p$, I – matrice unitară şi $x_0 = x + \left(I - P\right)p$ în *(4)-(5),* atunci derivând după *p* şi luând în considerare proprietăţile lui *P*, se obţine în coordonatele iniţiale

$P = H^T\left(HH^T\right)^{-1}H, x=\left(I - P\right)p$, I – unitary matrix and $x_0 = x + \left(I - P\right)p$ in *(4)-(5),* then when derived after *p* and taking into account the properties of *P* can be obtained in the initial coordinates

$$f'_1(x) = f'_1\left(\left(I - P\right)p\right) = \left(I - P\right)p' = \left(I - P\right)\left(x_0 + d\right),$$

adică

$$x_1 - x_0 = -\left(I - P\right)f'_1(x_0),$$

iar pe de altă parte, utilizând metoda gradienţilor conjugaţi pentru problema formulată după primul pas, vom avea la fel

namely

iar pe de altă parte, utilizând metoda gradienţilor conjugaţi pentru problema formulată după primul pas, vom avea la fel

$$x_1 - x_0 = -\left(I - P\right)f'_1(x_0),$$

de unde rezultă afirmaţia enunţată.

hence the stated statement.

4. Complexitatea algoritmului

Din cele demonstrate, rezultă că operaţia de bază este calcularea matricei *P,* dimensiunile acesteia fiind date de

4. The complexity of the algorithm

From the above mentioned results that the basic operation in calculating the matrix *P*, its size is given by

$$\begin{bmatrix} P \\ n\times n \end{bmatrix} = \begin{bmatrix} H^T \\ n\times m \end{bmatrix} \mathrm{X} \left(\begin{bmatrix} H \\ m\times n \end{bmatrix} \mathrm{X} \begin{bmatrix} H^T \\ n\times m \end{bmatrix} \right)^{-1} \mathrm{X} \begin{bmatrix} H \\ m\times n \end{bmatrix}. \qquad (12)$$

În procesul de calcul, însă, se profită din plin de descompunerea lui *P* în produs de matrice, efectuând succesiv de la dreapta spre stânga operaţia de înmulţire a matricei la vector, numărul operaţiilor elementare micşorându-se simţitor. Pe lângă acest mare avantaj, mai există cel legat de stabilitate, deoarece anume operaţia inversării matricei generează multă instabilitate. Or, matricea supusă inversării are dimensiuni constante şi mult mai mici, comparativ cu cele care se înmulţesc. În prealabil, se cuvine subliniat că, pentru calcularea produsului matricei $\left(HH^T\right)$, se vor utiliza $m^2 \times (2\times n - 1)$ operaţii elementare. Mai sus, s-a arătat că, pentru multiplicarea a 4 matrice, parantezarea optimală se face prin $2\times n\times m - m + 2\times m\times m - m + 2\times n\times m - n$ înmulţiri şi adunări, ori componentele lui *P* din *(12)* au exact

In the process of calculation however, is taken full advantage of the decomposition of *P* into the matrix product performing sequentially from the right to left matrix multiplication operation of the vector, the number of elementary operations decreasing signifycantly. Besides this big advantage, there is one related to stability, because only matrix inversion operation generates more instability. Or, matrix subject to inversion is subject to constant and much smaller dimensions compared to those that are multiplying. Initially, it should be noted that for the calculation of the matrix product $\left(HH^T\right)$ we shall use $m^2 \times (2\times n - 1)$ elementary operations. Above we have shown that, for multiplying 4 matrixes, optimal bracketing is done by $2\times n\times m - m + 2\times m\times m - m + 2\times n\times m - n$ multiplications and sums, or

aceleaşi dimensiuni, deci, la minimizarea pe subspaţiu este nevoie de acelaşi număr de operaţii. Atunci, per total, la rezolvarea *(PA)* se vor utiliza cel mult

the components of *P* in *(12)* have exactly the same size, thus, minimizing the subspace requires the same number of operations . So, overall, in solving the *(PA)* will be used at most

$$N\times(\ m^2\times(2\times n-1)+2\times m\times m-m+2\times n\times m-n+n\times(2\times m-1)\ ) \qquad (13)$$

operaţii elementare. Evident $N = 1$, dacă restricţiile de tipul „ ≤ „ lipsesc. Din motivele expuse mai sus în *(13)*, nu se are în vedere inversarea matricei, care nu depinde de parametrul *n*, ci numai de *m*, aceasta are permanent aceeaşi dimensiune constantă – $m \times m$, $m << n$, iar complexitatea operaţiei este $O(m^3)$, totodată, multiplicările la vector din aceleaşi considerente au complexitatea liniară $O(mn)$, echivalentă cu $O(n)$. Atunci *(13)*, finalmente, poate fi scrisă în forma:

elementary operations. Obviously $N = 1$ if restrictions of the type "≤" are missing. For the reasons stated above in *(13)* is not taken into account the matrix inversion, which does not depend on the parameter *n*, but only *m*; it always has the same constant size – $m \times m$, $m << n$ and the complexity of the operation is $O(m^3)$, while the vector multiplications of the same considerations have linear complexity $O(mn)$, equivalent to $O(n)$). Then *(13)*, finally, can be written in the form

$$N\times\left[2\times m^2\times n+4\times m\times n-2\times n-m-m^2\right]. \qquad (14)$$

Revenind acum la (*PA*), *(4)-(6)* formal *N* depinde de numărul de restricţii de tipul „≤„, dar, deoarece acestea se referă la componentele variabilei *x*, urmează că sunt nu mai mult de $2\times n$ şi *N*, în consecinţă, depinde de *n*. La evaluarea lui *N*, se disting câteva cazuri. Din cele menţionate anterior, rezultă că, în problema auxiliară, este raţional de introdus iniţial toate restricţiile, în acest caz, la pasul *k* prezintă interes doar situaţia când câteva dintre acestea devin pasive, ieşind la suprafaţa paralelipipedului. Dacă se constată că multiplicatorul lui Lagrange corespunzător este nenegativ, se îndeplinesc condiţiile de minimum şi procesul de calcul numeric s-a încheiat. Dacă însă multiplicatorul lui Lagrange este negativ, atunci componenta respectivă se readuce în matricea *H*. Important este că acest proces este monoton, funcţionala descreşte de la pas la pas, iar fiecare pas se termină neapărat după un număr finit de operaţii de aflare a minimumului. Aceste concluzii decurg din teorema fundamentală formulată în [5], care demonstrează că operatorul de proiecţie minimizează o formă pătratică cu

Turning now to *(PA) (4)-(6)* formal *N* depends on the number of restrictions such as "≤" but because they relate to parts of the variable *x*, it follows that there are no less than $2\times n$ and *N*, therefore, depends on *n*. In the evaluation of *N*, there can be distinguished several cases. From the foregoing, it follows that the auxiliary problem is rational to introduce initially restrictions; in this case, in step *k* of interest is only the situation when some of them become passive, then coming out to the surface of the cuboid. If it turns out that corresponding Lagrange multipliers are non-negative, they fulfil the minimum conditions and numerical computation process has ended. But if the Lagrange multiplier is negative, then the matrix component is brought back to matrix *H*. It is important that this process is monotonous, functional, decreasing step by step, and each step necessarily terminates after a finite number of operations of finding the minimum. These conclusions arise from the fundamental theorem formulated in [5], demonstrating that the operator projection minimizes a quadratic form with

matrice simetrică şi pozitiv definită în cel mult *n* paşi. Însă rigurozităţile matematice de mai sus îmbunătăţesc esenţial aceste rezultate, demonstrând că, în cazul matricei unitare, minimumul se atinge dintr-un singur pas, numărul de operaţii elementare reducându-se considerabil, ceea ce a făcut ca algoritmul elaborat să fie declarat optimal. Astfel, complexitatea acestui algoritm se poate scrie sub forma:

symmetric matrix and defined positively in most *n* steps. But mathematical rigors above essentially improve these results, demonstrating that in the case of unitary matrix, the minimum is reached in one step, reducing the number of elementary operations considerably, which made the developed optimal algorithm to be declared as optimal. Thus, the complexity of this algorithm can be written as:

$$O(nm^2, N). \qquad (15)$$

Suplimentar la aceste afirmaţii, componenta respectivă, odată readusă în matrice, determină multiplicatorul lui Lagrange corespunzător să-şi păstreze pe întreg parcursul iteraţiei semnul nenegativ în procesul de calcul, fapt dovedit şi de formulele analitice *(10)-(11),* ceea ce exclude revenirea a doua oară la suprafaţă a componentei respective, menţinând, totodată, valoarea numerică a acestei componente în interiorul paralelipipedului, graţie alegerii adecvate a pasului α^k. Aşadar, $N \leq 2\times n$ pentru restricţiile de jos asupra variabilelor, analogic, aceeaşi estimare este valabilă şi pentru restricţiile de sus, în consecinţă, per total $N \leq 4\times n$. De observat comportamentul formării mulţimii de indici pentru această situaţie

In addition to these statements, the respective component, once returned to the matrix determines the Lagrange multiplier appropriate to retain throughout iteration the mark of non-negative in the calculation, which has been proven also by analytical formulas *(10)-(11),* which excludes returning a second time to the surface of the respective component, while maintaining the numerical value of this component within the cuboid, through appropriate choice of step α^k. So, $N \leq 2\times n$ for lower restrictions on variables, analogically, the same estimation applies to the above restrictions, consequently, overall $N \leq 4\times n$. The observed trend in setting of indices for the situation

$$J_{activ}(p^0_k) \setminus \{j\}\ ,\ J_{pasiv}(p^0_k) \cup \{j\},$$

anume mulţimea indicilor pasivi mereu se lărgeşte şi includerea este strictă:

namely passive indices crowd always broadens and inclusion is strict

$$J_{pasiv}(p^0_{k+1}) \supset J_{pasiv}(p^0_k),$$

proces care poate dura numai un număr finit de ori, cumulat cu descreşterea funcţionalei, aceasta înseamnă că mulţimile de indici $J(p^0_k)$ nu se pot repeta, ca rezultat – convergenţa algoritmului rezolvării problemei auxiliare într-un număr finit de operaţii.

a process which can take only a finite number of times, combined with decreased functionality, it means that the sets of indices $J(p^0_k)$ cannot be repeated; as a result – convergence algorithm in solving the problem in a finite number of auxiliary operations.

5. Compararea cu alţi algoritmi

Desigur, o astfel de evaluare optimală a complexităţii algoritmului *PG* porneşte de la forma specială a restricţiilor. De aceea, pentru compararea cu alţi algoritmi, sunt necesare

5. Comparison with other algorithms

Of course, such an assessment *PG* optimal algorithm complexity starts to form special restrictions. Therefore, comparing it with other algorithms requires some

câteva generalizări şi estimări asimptotice. Este ştiut, bunăoară, că pentru problemele de optimizare de dimensiuni mari, încercarea de a le rezolva utilizând cea mai populară metodă – metoda simplex, deseori, se confruntă cu dificultăţi, deoarece numărul de iteraţii, în acest caz, creşte exponenţial faţă de dimensiunea problemei. Ulterior, au fost propuşi alţi algoritmi, polinomiali în sensul complexităţii [4], dar departe de a fi şi funcţionali, de aceea, metoda – simplex rămâne, în continuare, cea mai des utilizată. Fie în continuare problema *(4)-(6).* Algoritmii direcţi de rezolvare conduc la soluţia exactă a sistemului. Metodele directe aduc sistemul prin transformări de echivalenţă, la un sistem particular (diagonal, triunghiular etc.), care se rezolvă prin metode elementare. Este cunoscut, la fel, că, în acest caz, complexitatea metodelor directe de rezolvare este de ordinul $O(n^3)$, spre deosebire de cele iterative pentru care complexitatea este de $O(n^2)$, dar ultimele construiesc doar un şir de aproximaţii pentru x, $\nu = 1,2,3,...$ convergent în anumite condiţii la soluţia exactă, iar procesul iterativ se opreşte când aproximaţia de ordin ν se încadrează între limitele unei precizii stabilite iniţial, considerente din care acestea nu sunt examinate aici. În atare situaţie, se pot compara doar metode finite de rezolvare. Creşterea dimensiunii problemei şi trecerea acesteia în clasa de probleme de dimensiuni mari, automat o transformă în problemă foarte dificilă de rezolvat, în speţă la efectuarea calculelor numerice. De aceea, este foarte important de propus metode şi modele, care ar fi, într-un anumit sens „imune" la dimensiunea problemei. Desigur că astfel de metode şi modele sunt excepţie de la regulă, căci majoritatea depind esenţial de acest „blestem" al problemelor de optimizare. Modelul *PG* este anume un astfel de caz fericit, iar numărul minim de operaţii elementare necesare găsirii soluţiei permite declararea acestuia drept algoritm optimal pentru probleme de opti-

generalizations and asymptotic estimations. It is known, for instance, that for large optimization problems, trying to solve them using the most popular method – the simplex method often faces difficulties because the number of iterations in this case, increases exponentially compared to the size of the problem. Subsequently, other algorithms have been proposed, within the meaning of polynomial complexity [4], but far from being functional and, therefore, the simplex method still remains the most widely used. Let us consider the problems *(4)-(6).* The direct algorithms lead to finding the exact solution of the system. Direct methods bring equivalence transformations to the system, to a particular system (diagonal, triangular etc.), which are resolved by elementary methods. It is also known that in this case, the complexity of direct methods of resolution is of the order $O(n^3)$, as opposed to the iterative complexity for which the complexity is $O(n^2)$, but the last build only a number of approximations to x, $\nu = 1,2,3,...$ convergent under certain conditions to the exact solution and the iterative process stops when the approximation ν order falls within the limits of accuracy set initial considerations of which are not examined here. In such a situation can be compared only finished solving methods. Increasing the size of the problem and its passage in large class of problems automatically turns very difficult problem to solve, namely the numerical calculations. It is therefore very important to propose methods and models that would be in some sense "immune" to the extent of the problem. Of course such methods and models are an exception, as most depend essentially on the "curse" of optimization problems. The *PG* model is exactly a happy event, and the minimum number of elementary operations needed to find its solution allows the declaration as optimal algorithm for nonlinear optimization problems with special

mizare neliniară cu restricţii speciale, numărul de operaţii elementare fiind

restrictions, the number of elementary operations being

$$4 \times n \times \left[2 \times m^2 \times n + 4 \times m \times n - 2 \times n - m - m^2 \right]. \quad (16)$$

De menţionat că, în cazul când în *(4)-(6)* *m* tinde către *n*, modelul practic se transformă într-unul clasic liniar, complexitatea algoritmului devenind $O(n^4)$. Dacă se compară acest rezultat cu complexitatea din [4] – un record în domeniu, dată în forma $O(nm^5log^2m)$, *m,n* – însemnând dimensiunile matricei restricţiilor şi care în condiţii similare creşte până la $O(n^6log^2n)$, se constată o diminuare a complexităţii pentru algoritmul descris mai sus cu peste două ordine, suplimentar, în algoritmul construit, se iau în evidenţă uşor şi restricţiile asupra semnului variabilelor.

6. Concluzii

În studiu, s-a construit un algoritm optimal de rezolvare a problemei auxiliare a modelului *PG*, funcţional din toate punctele de vedere. Complexitatea acestui algoritm este $O(nm^2, N)$, dată de *(15)*, numărul de operaţii elementare este limitat de *(16)*. Matricea supusă inversării nu depinde de dimensiunea problemei *n*, ci numai de *m*, altfel spus, matricea inversabilă are permanent aceeaşi dimensiune constantă – $m \times m$, $m << n$, iar complexitatea operaţiei este $O(m^3)$. Pornind de la aceste evaluări, se poate concluziona că *PG* este absolut funcţional şi practic „imun" [2] la dimensiunea problemei de optimizare.

It should be noted that if *(4)-(6)* *m* tends to *n* , the model practically turns into a classic linear algorithm, the complexity becomes $O(n^4)$. If we compare this result with the complexity of [4] – a record in this field, given in the form $O(nm^5log^2m)$,), *m,n* – meaning the matrix dimensions restrictions and similar conditions of increase up to $O(n^6log^2n)$, there is a decrease in the complexity of the algorithm described above by more than two orders with additional built in algorithm that take out easily as well as the restrictions on the sign variables.

6. Concluzions

In the study, we constructed an algorithm for solving the problem of auxiliary optimal model *PG*, functional in all respects. The complexity of this algorithm is $O(nm^2, N)$, given by *(15)*, the number of elementary operations is limited by *(16)*. The matrix subject to the reversal of the matrix does not depend on the size of the problem *n*, but only by *m*, that is, the inverse matrix that always has the same constant size – $m \times m$, $m << n$, and the operation complexity is $O(m^3)$. Based on these assessments, it can be concluded that *PG* is fully functional and practically "immune" [2] to the size of the optimization problem.

Referinţe/References:

1. GÂRLĂ E. *Ştiinţificitatea strategiilor de dezvoltare economică*. Academia de Studii Economice a Moldovei, 2015, p. 255. ISBN 978-9975-75-762-1.
2. GÂRLĂ E. Model imun la dimensiunea problemei de optimizare, *Analele ASEM*, Ediţia a IX-a, 2011, p. 255-259. ISBN 978-9975-75-567-2.
3. PSHENICHNIY B. *Nonsmooth optimization and nonlinear programming*. Pergamon Press,1978.
4. KHACHIYAN L.G. *Polynomial algorithms in linear programming*. Computational Mathematics and Mathematical Physics, volume 20, Issue 1, 1980, pages 53-72.
5. PEARSON J. D. *Variable metric methods of minimization. Research Analysis Corporation*, McLean, Virginia, 1968.