International Institute for Applied Systems Analysis
IIASA www.iiasa.ac.at

On the Use of Matrix Factorization Techniques in Penalty Function Methods for Structured Linear Programs

Chebotarev, S.

IIASA Research Memorandum
May 1977

RM-77-21

ON THE USE OF MATRIX FACTORIZATION TECHNIQUES IN PENALTY FUNCTION METHODS FOR STRUCTURED LINEAR PROGRAMS

Spartak Chebotarev

May 1977

PREFACE

The penalty function method (PFM) has long been one of very few techniques which were successful in solving nonlinear mathematical programs. Its main advantage is that it helps to obtain a rough approximation to a solution very quickly and requires almost no additional memory.

However, when applied to linear programs, it has proved to be incompatible with direct methods, such as the Simplex-Method, with respect to speed and accuracy. It is not surprising, however, for unlike the Simplex-Method, no effort was made to try to deeply understand the structure of unconstrained semi-quadratic optimization problems arising when PFM is applied.

In this paper, it is shown that the traditional PFM, with quadratic penalty function, is, in fact, finite and when applied together with matrix factorization schemes, possesses some nice features, which allow us to solve large-scale problems.

The application of the proposed algorithm to structured linear programs, especially to dynamic linear programs, which arise in different IIASA areas, is described.

ABSTRACT

An algorithm converging to an optimal solution of a linear program in a finite number of steps is proposed. The algorithm is based on the use of smooth penalty functions as well as on matrix factorization techniques. It consists of finding corner points of the piece-wise linear unconstrained minima trajectory.

The application of the algorithm to dynamic linear programs and block-angular programs is described.

ACKNOWLEDGEMENT

The author is indebted to R. Mifflin for his valuable comments and criticism.

On The Use of Matrix Factorization Techniques in Penalty Function Methods for Structured Linear Programs

1. INTRODUCTION

Since the very beginning of linear programming applications, direct numerical methods such as the Simplex-Method and its different modifications became most popular in both theoretical and applied research. The use of triangular factorization schemes in the Simplex-Method together with different "tricks" in pivoting strategies have led to very efficient direct algorithms for linear programming. A review of such methods is given in [1]. It would not be a big mistake to say, that now the main steps of the Simplex-Method are carried out in an almost optimal way, so the main direction of current research activity in this field is the modification of general-purpose direct algorithms for solving specially structured linear programs.

On the other hand, many books and articles are devoted to iterative schemes for linear programs solutions, and many of them deal with penalty function techniques. The main reason for their development is that, in principal, these techniques provide an approximate solution to the problem much faster than any one of the direct methods. Unfortunately, the refinement of the approximate solution takes such a long time that it turns out to be only a waste of time and money.

However, recently a number of articles have been published, which describe methods using the modified Lagrange functions, and in principal, provide exact solutions to linear programs, (see, i.g., [2]). A review of such methods is given in [1].

The main purpose of this paper, however, is to show that a traditional quadratic penalty function scheme reinforced by the use of triangular factorization also gives the exact solution to linear programs, and is free of known penalty-function drawbacks, such as poor convergence in the vicinity of an optimal solution, which it usually was supposed to have.

Although being somewhat inferior to the Simplex-Method when solving general LP problems, the algorithm proposed here seems to be more effective in the case of so-called "staircase" problems. So the main field of application of the algorithm is in solving dynamic linear problems. Another class of LP problems where the algorithm has proved to be effective is block-angular linear programs with coupling columns.

2. A SUFFICIENT CONDITION FOR UNIQUENESS OF THE UNCONSTRAINED MINIMA TRAJECTORY

Let A be an $m \times n$ matrix, and let b and p be column vectors with m and n components respectively. We consider the linear programming (LP) problem in the canonical form

$$\min_x p^T x = f \quad , \tag{1}$$

subject to

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

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

where "T" denotes the transpose.

In what follows it is supposed that $m < n$, and that there exists a unique solution $\bar{x}$ to the problem (1)-(3). Let us introduce the function $F(q,x)$ as follows:

$$F(q,x) = qp^T x + \tfrac{1}{2}(Ax-b)^T(Ax-b) + \tfrac{1}{2}x^T\theta(-x)x \quad , \tag{4}$$

where $q > 0$ is an arbitrary scalar and the elements of the diagonal matrix $\theta(-x)$ are defined by the relation

$$\theta_{ii}(-x) = \begin{cases} 1, & \text{if } x_i < 0 \\ 0, & \text{otherwise} \end{cases} . \tag{5}$$

As is well-known [1, Chapter 9], the following relation holds

$$\lim_{q \to +0} [\min_{x} F(q,x)] = f \quad , \tag{6}$$

provided that the minimum exists for any $q > 0$. This relation allows us to find an approximate solution to the problem (1)-(3) by solving a sequence of unconstrained optimization problems. Each of the problems consists of minimization of the nonlinear function $F(q,x)$ for a fixed value of q. Usually the rate of convergence depends very much on the choice of the sequence $\{q_k\}$. One can find the discussion of the question and references in the chapter written by D.M. Ryan in [1].

If q varies continuously then there is a trajectory of minimizers of $F(q,x)$ which terminates at the solution $\bar{x}$ of the problem (1)-(3). In what follows we suppose that the trajectory is unique, i.e. that $F(q,x)$ has a unique minimizer $x(q)$ for any $q > 0$. A sufficient condition for uniqueness, which reminds me of the Haar condition, is given in Theorem 1 below.

It is evident that $x(q)$ satisfies the equation

$$qp + A^T(Ax-b) + \theta(-x)x = 0 \quad . \tag{7}$$

Using the notation

$$\Phi(x) = \tfrac{1}{2}(Ax-b)^T(Ax-b)$$

the function $F(q,x)$ may be written in the form

$$F(q,x) = qp^T x + \Phi(x) + \tfrac{1}{2}x^T\theta(-x)x \quad .$$

Let an arbitrary $q > 0$ be given. Suppose, also, that there are two minimizers of $F(q,x)$: x_1' and $x_2' \neq x_1'$. Since $F(q,x)$ is convex, any point of the segment $[x_1', x_2']$ also minimizes $F(q,x)$. It is evident that one may choose a segment $[x_1, x_2]$ $x_1 \neq x_2$ embedded in $[x_1', x_2']$ and such that the corresponding coordinates of x_1 and x_2 have the same signs, i.e. the vectors x_1 and x_2 belong to the same orthant in the euclidean space E^n.

Lemma 1. Let $x \in [x_1, x_2]$, and the set of indices J be defined by $J = \{i/x_i < 0\}$. Let $y^1, y^2 \neq y^1$ be two arbitrary vectors such that $y^1, y^2 \in [x_1, x_2]$. Then $y^1_i = y^2_i$ for $i \in J$, i.e. the segment $[x_1, x_2]$ lies on a hyperplane orthogonal to the unit vectors e_i, $i \in J$.

Proof: Suppose the contrary. Let $x_i < 0$, and the hyperplane containing $[x_1, x_2]$ is not orthogonal to e_i. Then $F(q,x)$ takes the form

$$F(q,x) = qp^T x + \Phi(x) + \tfrac{1}{2} \sum_{j \neq i} x_j^2 \theta(-x_j) + \tfrac{1}{2} x_i^2 \quad . \qquad (8)$$

The first three terms in (8) constitute a convex function while the last one describes a strictly convex function with respect to any direction not orthogonal to e_i. Hence $F(x,q)$ is strictly convex with respect to such directions, which implies the uniqueness of the minimizer: a contradiction. ||

Lemma 2. Suppose that any m columns of A are linearly independent. Then the number of nonnegative coordinates of $x \in [x_1, x_2]$ is less than or equal to $n - m - 1$.

Proof: Denote the number of negative coordinates of $x \in [x_1, x_2]$ by $|J|$. As was shown in Lemma 1, $y^1_i = y^2_i$ for y^1, $y^2 \in [x_1, x_2]$ and $i \in J$. Hence the function $F(q,x)$ equals

$$\hat{F}(q,x) = q\hat{p}^T\hat{x} + \tfrac{1}{2}(\hat{A}\hat{x}-b)^T(\hat{A}\hat{x}-b)$$

to within an additive constant. Here the matrix $\hat{A}$ is the restriction of A to the columns A^i, $i \notin J$, $\hat{p}^T$ and $\hat{x}$ are the analogous restrictions of p^T and x respectively. If $|J| \geq n - m$ then, since any m columns of A (and consequently of $\hat{A}$) are linearly independent, the function $(\hat{A}\hat{x}-b)^T(\hat{A}\hat{x}-b)$ is strictly convex, which again contradicts the nonuniqueness assumption. ||

Let us add the row p^T to A and denote the new matrix by $\tilde{A}$. Now we may state a theorem.

Theorem 1. The function $F(x,q)$ *has a unique minimizer for any* $q > 0$ *if any* $m+1$ *columns of* $\tilde{A}$ *are linearly independent.*

Proof: Let $x \in [x_1, x_2]$. Let also x, $\hat{p}$ and $\hat{A}$ be defined as in the proof of Lemma 2. Then $\hat{x}$ is the solution to

$$q\hat{p} + \hat{A}^T(\hat{A}\hat{x}-b) = 0 \quad ,$$

hence

$$\hat{p} = \hat{A}^T\left(\frac{b-\hat{A}\hat{x}}{q}\right) \quad .$$

Making use of the notation $\alpha = (b-\hat{A}\hat{x})/q$ we get $\hat{p} = \hat{A}^T\alpha$. If $\alpha = 0$ for all $\hat{x} \in [\hat{x}_1, \hat{x}_2]$ then it means (from Lemma 1) that the solution is unique. Else there is a $\hat{x}_0$ such that $(b-\hat{A}\hat{x}_0)/q \neq 0$, so $\hat{p}$ is a linear combination of columns of $\hat{A}^T$, i.e. $\hat{p}^T$ is a linear combination of rows of $\hat{A}$.

So we have shown that if there are two different minimizers then a vector $\hat{p}$ is a linear combination of rows of $\hat{A}$. Using Lemma 2, we can construct the contradictory statement which completes the proof. ||

In what follows it is supposed that this uniqueness condition is satisfied.

3. THE OUTLINE OF THE ALGORITHM

In [3] it was shown that this trajectory is piece-wise linear and is linear in each orthant of the euclidean space E^n. This property is used in the following algorithm, which consists of finding the "corner" points of the trajectory.

Suppose that an initial point $x^0 \equiv x(Q_0)$, $Q_0 > 0$ of the trajectory is known.[1] Let $C_0 = A^TA + \mathrm{diag}\,[\theta(-x^0)]$ and $d = A^Tb$. Evidently x^0 satisfies the equation

$$C_0x = d - qp \qquad (9)$$

[1] The solution process is described in the next section.

with

$$q = Q_o \quad .$$

Note that x^0 is unique, hence C_0 is nonsingular.

Solve the equation

$$C_0 y = d \tag{10}$$

for an auxiliary vectory y, and compute

$$q_1^{(i)} = \frac{y_i}{y_i - x_i^0} Q_0 \quad , \qquad i = 1,\ldots,n \quad . \tag{11}$$

for such is that $y_i \neq x_i^0$. In fact, $q_1^{(i)}$ is the value of parameter q with which the i^{th} component of the solution of (9) became zero. So Q_1

$$Q_1 = \max \{0; q_1^{(i)}\} = q_1^{(k)} \tag{12}$$
$$q_1^{(i)} < Q_0$$

defines the value of q corresponding to the first (with respect to x^0) angular point of the trajectory x(q). This point is defined by the relation

$$x^1 \equiv x(Q_1) = y - \frac{Q_1}{Q_0} (y - x^0) \quad . \tag{13}$$

This completes the first iteration of the algorithm. The next segment of the trajectory corresponds to the matrix C_1 which differs from C_0 in only one element, namely C_{kk}, for x_k has changed its sign at $q = Q_1$.[2] So,

---

[2] If k is not unique, we have to take an arbitrary Q_2, $0 < Q_2 \leq Q_1$ and find $x(Q_2)$ using x^1 as an initial approximation.

$$(C_1)_{ij} = \begin{cases} (C_0)_{kk} + \text{sign } (x_k^0), & \text{if } i = j = k \\ (C_0)_{ij}, & \text{otherwise} \end{cases} \quad . \tag{14}$$

Now we have to solve

$$C_1 y = d$$

for y. Then compute

$$q_2^{(i)} = \frac{y_i}{y_i - x_i^1} Q_1 \quad , \qquad i = 1,\ldots,n$$

$$Q_2 = \max \{0, q_2^{(i)}\} = q_2^{(l)}$$

$$q_2^{(i)} < Q_1$$

and the next angular point x^2:

$$x^2 = y - \frac{Q_2}{Q_1} (y - x^1)$$

and so on.

The algorithm terminates at $Q = 0$ in a finite number of steps and when it is so, y is the exact solution to (1)-(3). The proof is given in [2].

For the implementation of the algorithm we need a method for evaluation of the initial point x^0 and an efficient procedure for solving systems such as (9) which take into account the slight modification of the matrices $C_0, C_1, \ldots$ at each step of the algorithm.

The next section describes the iterative procedure for minimizing $F(q,x)$ for a fixed value of $q = Q_0 > 0$, i.e. for calculation of the initial point x^0.

4. DETERMINATION OF THE INITIAL POINT

The minimizing x of $F(Q_0,x)$ satisfies the equation

$$A^TAx + \theta(-x)x = A^Tb - Q_0p \tag{15}$$

which, being nonlinear in the large, is linear in every orthant of E^n. This property allows us to construct an effective computational procedure using a matrix factorization technique.

Let an arbitrary Q_0 be chosen. In what follows we will make use of the notations: $d_0 = A^Tb - Q_0p$, $\bar{C} = I + A^TA$ where I denotes the $n \times n$ unity matrix. Now (15) may be rewritten as

$$\bar{C}x = \sigma(x)x + d_0 \quad . \tag{16}$$

where $\sigma(x)$ is a diagonal matrix such that

$$\sigma_{ii}(x) = \begin{cases} 0, & \text{if } x_i < 0 \\ 1, & \text{if } x_i \geq 0 \end{cases} ,$$

Suppose that we have a vector $x^k \in E_k^n$, where E_k^n is an orthant in E^n. Let the vector x^{k+1} be defined by

$$\bar{C}x^{k+1} = \sigma_k x^k + d_0 \tag{17}$$

where

$$\sigma_k = \sigma(x^k) \quad . \tag{18}$$

The formulas (17)-(18) define the linear autononous iterative process for which the following theorem holds:

Theorem 2. The process (17)-(18) converges to the solution of (15) for any initial approximation.

Proof: For the proof of the theorem it is sufficient [3, p.] to show that (i) the process (17)-(18) is monotonic, (ii) the sequence $\{x^k\}$ is compact, and (iii) the algorithmic map from x^k to x^{k+1} is continuous.

Let us prove at first that the process (17)-(18) generates a strictly decreasing sequence $\{F(Q_0,x^k)\}$. Denote $F(Q_0,x)$ by $F^0(x)$, and introduce the function

$$F_k^0(x) = \tfrac{1}{2}(Ax-b)^T(Ax-b) + Q_0p^Tx + \tfrac{1}{2}x^T\Theta(-x^k)x$$
$$= \tfrac{1}{2}(Ax-b)^T(Ax-b) + Q_0p^Tx + \tfrac{1}{2}x^T(I-\sigma_k)x \quad .$$

So $F_k^0(x) = F^0(x)$ when $x \in E_k^n$.

Suppose that x^k and x^{k+1} are as in (17). $F_k^0(x)$ is convex with respect to x, hence

$$F_k^0(x^k) - F_k^0(x^{k+1}) \geq (x^k-x^{k+1})^T\nabla F_k^0(x^{k+1}) \qquad (19)$$

where $\nabla F_k^0(x^{k+1})$ is the gradient of $F_k^0(x)$ evaluated at the point x^{k+1}. It follows from (17) that

$$(x^k-x^{k+1})^T = (x^k)^T - (x^k)^T\sigma_k\bar{C}^{-1} - d_0^T\bar{C}^{-1} \quad . \qquad (20)$$

On the other hand

$$F_k^0(x^{k+1}) = \bar{C}x^{k+1} - \sigma_kx^{k+1} - d_0 = \sigma_k[x^k-\bar{C}^{-1}\sigma_kx^k-\bar{C}^{-1}d_0] \quad . \qquad (21)$$

Substituting (20) and (21) into (19) and using the symmetry of $\bar{C}^{-1}$ and σ_k we obtain

$$F_k^0(x^k) - F_k^0(x^{k+1}) \geq [(I-\bar{C}^{-1}\sigma_k)x^k-\bar{C}^{-1}d]^T$$
$$\sigma_k[(I-\bar{C}^{-1}\sigma_k)x^k - \bar{C}^{-1}d_0]$$
$$= (x^k-x^{k+1})^T\sigma_k(x^k-x^{k+1}) \geq 0 \quad . \qquad (22)$$

Let us introduce four sets of indices

$$J_1 = \{i/x_i^k \geq 0, x_i^{k+1} \geq 0\}$$
$$J_2 = \{i/x_i^k \geq 0, x_i^{k+1} < 0\}$$
$$J_3 = \{i/x_i^k < 0, x_i^{k+1} \geq 0\}$$
$$J_4 = \{i/x_i^k < 0, x_i^{k+1} < 0\} \quad .$$

It is easy to see that

$$F^0(x^{k+1}) = F_k^0(x^{k+1}) + \tfrac{1}{2}\sum_{i\in J_2}(x_i^{k+1})^2 - \tfrac{1}{2}\sum_{i\in J_3}(x_i^{k+1})^2 \quad (23)$$

Using (22) and (23) we obtain

$$F^0(x^k) \geq F_k^0(x^{k+1}) + \sum_{i\in J_1\cup J_2}(x_i^k - x_i^{k+1})^2 = F^0(x^{k+1}) - \tfrac{1}{2}\sum_{i\in J_2}(x_i^{k+1})^2$$

$$+ \tfrac{1}{2}\sum_{i\in J_3}(x_i^{k+1})^2 + \sum_{i\in J_1\cup J_2}(x_i^k - x_i^{k+1})^2 = F^0(x^{k+1}) - \tfrac{1}{2}\sum_{i\in J_2}(x_i^{k+1})^2$$

$$+ \sum_{i\in J_2}(x_i^k - x_i^{k+1})^2 + \tfrac{1}{2}\sum_{i\in J_3}(x_i^{k+1})^2 + \sum_{i\in J_1}(x_i^k - x_i^{k+1})^2 \quad . \quad (24)$$

If $J_1 \cup J_2 \cup J_3$ is empty then by (17)-(18)

$$x^{k+1} = \bar{C}^{-1}d_0 < 0$$

and

$$x^\nu = \bar{C}^{-1}d_0 = x^{k+1}, \text{ for all } \nu \geq k+1$$

that is x^{k+1} is a stationary point of the process. If J_2 is non-empty, we obtain that

$$\sum_{i\in J_2}(x_i^k - x_i^{k+1})^2 - \tfrac{1}{2}\sum_{i\in J_2}(x_i^{k+1})^2 \geq \sum_{i\in J_2}(x_i^{k+1})^2 - \tfrac{1}{2}\sum_{i\in J_2}(x_i^{k+1})^2 > 0$$

by definition of J_2. Hence

$$F^0(x^k) > F^0(x^{k+1}) \quad .$$

So suppose J_2 is empty. If J_3 is non-empty then it is evident from (24) that this inequality also holds if there exists $x_i^{k+1} > 0$, $i \in J_3$. If $x_i^{k+1} = 0$ for all $i \in J_3$ then consider the following possible cases.

If $J_1 \cup J_4$ is empty, then $x_i^k < 0$, $x_i^{k+1} = 0$ for all i. If follows from (16) that $\bar{C}x^{k+1} = d_0$ and hence $\bar{C}^{-1}d_0 = 0$. But then $x^{k+2} = \bar{C}^{-1}\sigma_{k+1}x^{k+1} + \bar{C}^{-1}d_0 = \bar{C}^{-1}d_0 = 0$. So, x^{k+1} is a stationary point of the process.

If J_1 is empty but J_4 is not, then again by definition of σ_k we obtain $x^{k+2} = \bar{C}^{-1}\sigma_{k+1}x^{k+1} + \bar{C}^{-1}d_0 = \bar{C}^{-1}d_0 = x^{k+1}$, so x^{k+1} is a stationary point.

If J_1 is non-empty then consider the term $\sum_{i\in J_1}(x_i^k - x_i^{k+1})^2$ from (24). If this term equals zero, then it means that $x_i^k = x_i^{k+1} \geq 0$ for all $i \in J_1$. Then we have $x^{k+2} = \bar{C}^{-1}\sigma_{k+1}x^{k+1} + \bar{C}^{-1}d_0 = \bar{C}^{-1}\sigma_k x^k + \bar{C}^{-1}d_0 = x^{k+1}$ by definition of σ_k and by (17). So, in this case as well, x^{k+1} is a stationary point of the process.

Consider now the case when $J_2 \cup J_3$ is empty. In this case, $J_1 \cup J_4$ is non-empty, and $\sigma_{k+1} = \sigma_k$, hence $F_k^0(x^{k+1}) = F_{k+1}^0(x^{k+1}) = F^0(x^{k+1})$. Now we may rewrite (22) in the form

$$F^0(x^k) - F^0(x^{k+1}) \geq (x^k - x^{k+1})^T \sigma_k (x^k - x^{k+1}) \geq 0$$

If the last inequality satisfies as an equality, then it follows that $x_i^{k+1} = x_i^k$ for $i \in J_1$. Since $\sigma_{k+1} = \sigma_k$ we obtain

$$x^{k+2} = \bar{C}^{-1}\sigma_k x^{k+1} + \bar{C}^{-1}d_0 = \bar{C}^{-1}\sigma_k x^k + \bar{C}^{-1}d_0 = x^{k+1}$$

that is x^{k+1} is a stationary point of (17)-(18).

Thus we have proved that if x^k is not a stationary point of (17)-(18) then

$$F^0(x^k) > F^0(x^{k+1}) \quad .$$

The compactness of $\{x^k\}$ is evident. Since $F^0(x)$ is convex and by the assumption has a unique minimizer, the set $\Omega(a) = \{x/F^0(x) \leq a\}$ is compact. Let $a = F^0(x^0)$ where x^0 is an initial approximation. Then $x^k \in \Omega(a)$ because the process is monotonic.

We have to prove now that the algorithmic map $M: x^k \to x^{k+1}$ is continuous. Let us suppose that we have two vectors $(x^k)^1$ and $(x^k)^2$. Without loss of generality we may consider them as

belonging to the same orthant of E^n, E_k^n. Suppose that $(x^{k+1})^1$ and $(x^{k+1})^2$ have been evaluated in accordance with (17), i.e.,

$$(x^{k+1})^1 = \bar{C}^{-1}\sigma_k(x^k)^1 + \bar{C}^{-1}d_0$$
$$(x^{k+1})^2 = \bar{C}^{-1}\sigma_k(x^k)^2 + \bar{C}^{-1}d_0 \quad .$$

Then

$$\begin{aligned} \| (x^{k+1})^1 - (x^{k+1})^2 \| &= \| \bar{C}^{-1}\sigma_k((x^k)^1 - (x^k)^2) \| \\ &\leq \| \bar{C}^{-1} \| \cdot \| \sigma_k \| \cdot \| (x^k)^1 - (x^k)^2 \| \\ &\leq \| \bar{C}^{-1} \| \cdot \| (x^k)^1 - (x^k)^2 \| \quad . \end{aligned}$$

For any $\varepsilon > 0$, having let $\| (x^k)^1 - (x^k)^2 \| < \delta = \dfrac{\varepsilon}{\| \bar{C}^{-1} \|}$ we have $\| (x^{k+1})^1 - (x^{k+1})^2 \| < \varepsilon$. $\|$

5. FACTORIZATION AND THE UPDATING PROCEDURE

The implementation of the iterative procedure involves the $L_0D_0L_0^T$--factorization of $\bar{C}$ where L_0 is a lower triangular matrix with a unit main diagonal, and D_0--a diagonal matrix. With this factorization at hand, the computation of x^{k+1} satisfying (17) consists of forward and backward substitution which is very easy to implement.

The structure of $\bar{C}$ is used explicitly in computing the $L_0D_0L_0^T$ factorization. It is easy to see that

$$C = I + A^TA = I + \sum_{j=1}^{m} A_j^TA_j \quad , \qquad (25)$$

where A_j is the -j^{th} row of A. So we may use a procedure for updating the factors of a modified matrix with a rank-one modification. In other words, starting with the unit matrix I, we compute factors for the matrix

$$\bar{C}^1 = I + A_1^TA_1 = L_0^1D_0^1L_0^{1T} \quad .$$

Then we repeat the procedure and obtain

$$\bar{C}^2 = \bar{C}^1 + A_2^T A_2 = L_0^2 D_0^2 L_0^{2T} \quad .$$

In m steps we will have the desired factorization:

$$\bar{C}^m \equiv \bar{C} = L_0^m D_0^m L_0^{mT} \equiv L_0 D_0 L_0^T$$

For the factor updating it is convenient to use Bennett's method [5], which in the case of rank-one modification can be described as follows:

Let $\bar{B} = B + \gamma uu^T$, where B is a symmetric $n \times n$ matrix, γ - is a scalar, and u - n-vector. If LDL^T - factorization of B is known

$$B = LDL^T$$

then $\bar{L}\bar{D}\bar{L}^T$ factorization of $\bar{B}$ is generated by the following recurrence relations:

1) Set $\gamma_1 = \gamma$, $u^{(1)} = u$, $i = 1$.

2) $\bar{D}_{ii} = D_{ii} + \gamma_i (u_1^{(i)})^2$.

3) If $i = n$, then go to (8) else $j = 1$.

4) $u_j^{(i+1)} = u_{j+1}^{(i)} - L_{i+j,i} u_1^{(1)}$.

5) $\bar{L}_{i+j,i} = L_{i+j,i} + \gamma_i u_1^{(i)} u_j^{(i+1)} / \bar{D}_{ii}$, $j = j + 1$.

6) If $j < n - i + 1$, then go to (4).

7) $\gamma_{i+1} = \gamma_i - (\gamma_i u_1^{(1)})^2 / D_{ii}$, $i = i + 1$; go to (2).

8) Stop.

Here $u_1^{(i)}$ denotes the first component of the vector $u^{(i)}$.

This procedure requires $\sim n^2 + 0(n)$ multiplications while the direct factorization of $\bar{B}$ requires some $n^3/3$ multiplications. So, factorization of $\bar{C}$ requires $\sim mn^2$ multiplications.

The same procedure is used for calculation of the LDL^T factorization of C_0. Namely, when x^0 is defined, we know which coordinates of x^0 are positive, so we can compute C_0 as

$$C_0 = \bar{C} - \Sigma e_i e_i^T \quad , \tag{26}$$

where e_i is the i^{th} unit n-vector, and summation in (26) is taken over all i's such that $x_i^0 > 0$. The number of positive coordinates of x^0 defines how many times we have to use the updating procedure to compute the factorization of C_0.

If, at - say the k^{th} step of the algorithm - a certain coordinate of x, e.g. x_i, changes its sign, then we will use the same updating procedure for calculation of the $L_{k+1}D_{k+1}L_{k+1}^T$ factorization of the matrix C_{k+1}. Clearly, vector u now takes the form $u = e_i$, where e_i - i^{th} unit vector.

Again, equation (10) is solved by use of forward and backward substitutions.

6. ANOTHER APPROACH TO FACTORIZATION

The factorization scheme described above suffers one heavy drawback, namely, when it is applied, it causes tremendous fill-in in the matrix L. To remedy this problem, one can exploit the special structure of C. Consider the following problem

$$\min p^T\bar{x}$$

subject to

$$\bar{A}\bar{x} \leq b$$

$$\bar{x} \geq 0$$

where $\bar{x} \in E^{n-m}$ and $\bar{A}$ is $m \times (n-m)$-matrix or adding slack variables we get the constraints:

$$y + \bar{A}\bar{x} = b$$

$$y, \bar{x} \geq 0 \quad .$$

Using the notation $x^T = (y,\bar{x})^T$ and $A = [E,\bar{A}]$ where E is unit $m \times m$-matrix, we reduce this problem to the form (1)-(3). Now A^TA takes the form

$$A^TA = \begin{bmatrix} E & \bar{A} \\ \bar{A}^T & \bar{A}^T\bar{A} \end{bmatrix} .$$

It is evident that the matrix

$$\tilde{C} = \begin{bmatrix} E & \bar{A} \\ \bar{A}^T & I + \bar{A}^T\bar{A} \end{bmatrix}$$

where I-unit $(n-m) \times (n-m)$-matrix is nonsingular, for

$$\tilde{C} = \begin{bmatrix} E & 0 \\ \bar{A}^T & I \end{bmatrix} \begin{bmatrix} E & \bar{A} \\ 0 & I \end{bmatrix} = LL^T . \qquad (27)$$

Let us rewrite (15) in the form

$$\tilde{C}x = \gamma(x)x + d_0 \qquad (28)$$

where $\gamma(x)$ is a diagonal matrix such that

$$\gamma_i(x_i) = \begin{cases} -1 & \text{if } x_i < 0 \text{ and } i \leq m \\ 0 & \text{if } x_i < 0 \text{ and } i > m \\ 1 & \text{if } x_i \geq 0 \text{ and } i > m \\ 0 & \text{if } x_i \geq 0 \text{ and } i \leq m \end{cases} .$$

Evidently the systems (15) and (28) are equivalent, hence we may solve (28) using the process (17)-(21) with obvious modifications.

So, in this case, the triangular factorization of C is known in advance, and we save memory and CPU time.

Triangular factorization of matrix C_0 (see (9)) may easily be obtained from (27) using Forrest-Tomlin (FT) updating procedure [6].

Consider y and $\bar{x}$ parts of vector x separately. Suppose that first r coordinates of vector $[y^0, \bar{x}^0] = x^0$ are negative, and its last coordinates are nonnegative. It is clear that a general case can be reduced to this situation using permutations.

To construct the matrix C_0, we now have to add unity to the first r diagonal elements of C and subtract unity from the last s ones. The factorization (27) now takes the form:

$$C_0 = \begin{pmatrix} E & 0 \\ \bar{A}^T & I \end{pmatrix} = \begin{pmatrix} 2E & 0 & \bar{A} \\ 0 & E & \\ & & \\ 0 & I & 0 \\ 0 & 0 & 0 \end{pmatrix} = LH \qquad (29)$$

(r, n-m, m, s)

where all E's and I's denote unity matrices of appropriate size, and the shaded area represents first r columns of $\bar{A}^T$ taken with the opposite sign. Note that $r \geq s$, for, if not, then H is singular.

Let us now describe the process of reduction of H to upper triangular form. First of all, note that the right-lower part of H corresponds to those coordinates of x^0 which during the iterative process were supposed to be negative. So at the first step we have to permute the columns of A so that all n - m of the last columns of A correspond to the negative coordinates of x^0. In other words, we have to exchange s last columns with s of the r first columns, so the permuted matrix ${}_pA$ takes the form

$$ {}_pA = \begin{pmatrix} E & 0 & \cdots & 0 \\ 0 & 0 & \cdots & E \\ 0 & E & \cdots & 0 \end{pmatrix} \begin{matrix} \\ \\ \}s \end{matrix} $$

(r-s; s; s)

So now the first r - s and the last n - m columns of ${}_pA$ correspond to negative coordinates of x^0 while the others--to the non-negative ones.

The equality constraints of the original problem become now

$$ {}_{p}Ax = b \quad . $$

Premultiplying this equality by

$$ B^{-1} = \begin{bmatrix} E & & 0 \\ 0 & & 0 \\ 0 & & E \end{bmatrix}^{-1} $$

we reduce it to the form

$$ \tilde{A}x = \tilde{b} $$

where

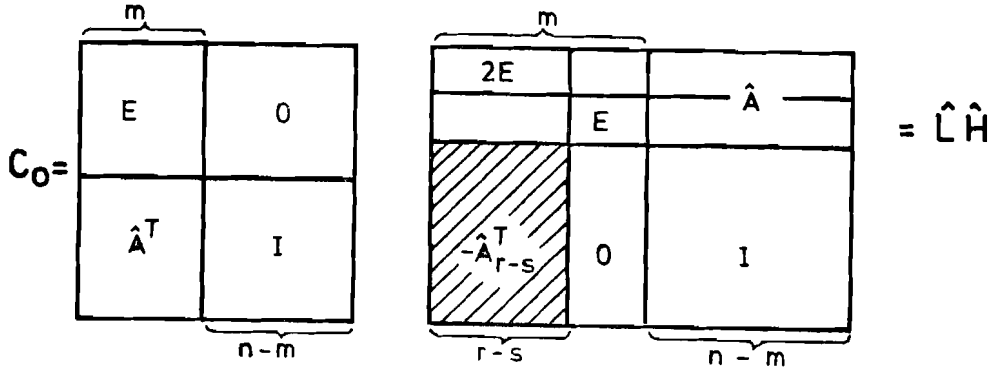
$$ \tilde{A} = \underbrace{\boxed{E}}_{m} \; \underbrace{\boxed{\hat{A}}}_{n-m} \quad , \tag{30} $$

and $\tilde{A} = B^{-1}\bar{A}$.

Naturally we don't need to perform explicit multiplication because we may keep B^{-1} in Product Form of the Inverse (PFI) or in Elimination Form of the Inverse (PFI). The matrix C_0 for (30) now takes the form

$$ C_0 = \begin{bmatrix} 2E & 0 & \multirow{2}{*}{$\hat{A}$} \\ 0 & E & \\ \hat{A}^T & & I + \hat{A}^T \hat{A} \end{bmatrix} $$

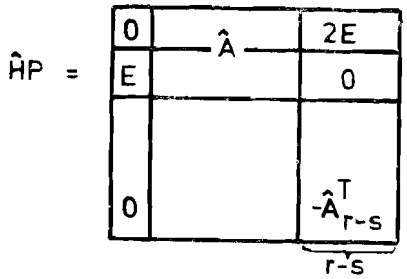
or



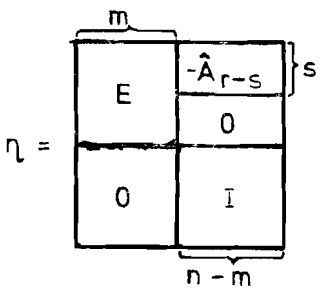
(31)

where the shaded area in $\hat{H}$ coincides with the first r - s columns of $\hat{A}^T$ taken with the opposite signs.

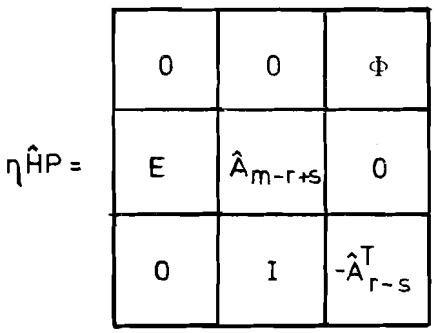
Now we have to operate on these columns to reduce $\hat{H}$ to an upper triangular form. Applying FT-procedure we first use column permutations to change $\hat{H}$ to the form



where P is the column permutation matrix. Premultiplying this matrix by



we get



where

$$\Phi = E + \hat{A}_{r-s}\hat{A}^{T}_{r-s} \quad . \tag{32}$$

Premultiplying (32) by P^T we get

$$U = P^T \eta H P = \begin{array}{|c|c|c|} \hline E & \hat{A}_{m-r-s} & 0 \\ \hline 0 & I & \hat{A}^T_{r-s} \\ \hline 0 & 0 & \Phi \\ \hline \end{array}$$

This matrix is almost upper triangular except the block Φ which being of a rather small size can be easily LDL^T-factorized with the help of (32) and Bennett procedure described above.

As it follows from the above in this approach, we need no calculation at all to get the triangular factors of $\tilde{C}$, for they are explicitly defined by (27) in terms of and only of the elements of $\bar{A}$. So no extra memory or CPU time is needed.

For LU-factorization of C_0 we need extra memory to store Φ matrix from (32) in factorized form. The amount required is approximately $\frac{1}{2}(r-s)^2$. The number of multiplications needed for LDL^T factorization of Φ amounts to $\sim(n-m)(r-s)^2$ which is relatively small. On the other hand, the other non-zeros of η and U coincides with those of $\hat{A}$ to within the sign. So, use of the FT-procedure saves a great deal of storage and CPU time when computing the triangular factorization of C_0.

The same scheme is used for calculation of triangular factors of matrices $C_1C_2,\ldots$.

7. APPLICATION OF DYNAMIC LINEAR PROGRAMMING

Consider now so-called dynamic linear programs, which sometimes are referred to as "staircase" programs. Evidently, in this case, fill-in in L matrix grows linearly with the dimension of the problem. This class of problems now attracts many researchers, but what should be done in this field outweighs heavily what has been done. For a short review see, e.g. [7].

The problem is

$$\min c_0^{(1)} u^0 + \sum_{k=1}^{n-1} (c_k^{(1)} u^k + c_k^{(2)} x^k) + c_n^{(2)} x_n \qquad (33)$$

subject to

$$G_k x^k + D_k u^k = b_k \qquad (34)$$

$$A_k x^k + B_k u^k - x^{k+1} = s_k \qquad (35)$$

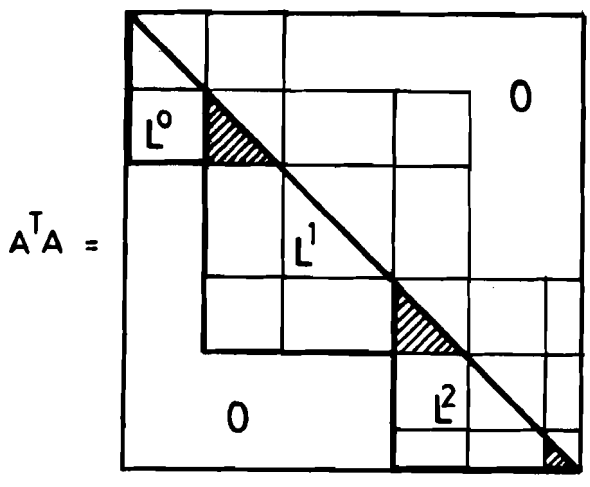
$$x^k \geq 0 \quad , \quad u^k \geq 0 \quad , \quad k = 0,\ldots,N-1 \qquad (36)$$

where x^0 is known initial state vector, x^k, u^k-unknown n- and r^k -dimensional vectors respectively, $b_k \in E^{m_k}, c_k^{(2)}, s_k \in E^n, c_k^{(1)} \in E^{r_k}$, and G_k, D_k, A_k, B_k--constant matrices of appropriate sizes.

In static formulation, the constraints matrix A of (35)-(36) in the case when N = 2 takes the form

$$A = \begin{bmatrix} D_0 & & & & & \\ B_0 & -E & & & & \\ & G_1 & D_1 & & & \\ & A_1 & B_1 & -E & & \\ & & & G_2 & D_2 & \\ & & & A_2 & B_2 & -E \end{bmatrix}$$

since x^0 is known and $G_0 x^0$ and $A_0 x^0$ can be be subtracted. Matrice $A^T A$ takes the form



On this figure, the heavy line engulfs the matrix L where $I + A^TA = LDL^T$ and D is a diagonal matrix. The matrix L in turn consists of N - 1 trapezoidal blocks $L^0, L^1, \ldots, L^{N-1}$. It is easy to show that block-triangular shape with a variable band-width doesn't change with change of diagonal elements provided L remains positive definite.

The triangular factorization of $I + A^TA$ is carried out in the following way. First, we transform blocks of A corresponding to k = 0 to matrices L^0, D^0 and upper triangle of L^1 shaded in Fig. 1. Then blocks of A corresponding to k = 1 are transformed to matrices L^1, D^1 and upper shaded triangle of L^2 and so on. The number of operations needed for determination of L and D is of order $\sum_{k=0}^{N-1} [m_k(n+r_k)^2 + n(2n+r_k^2)]$ and grows linearly with N.

When L and D are constructed, the algorithm continues as in the general case. The important idea here is the storage scheme for L and D. The elements of D are represented as diagonal elements of L which in turn are stored in two arrays [8]: VE(Values of Elements) and PD(Positions of the Diagonal elements in VE).

Array VE(k) contains all the non-zero elements of L^k written column by column.

Array PD(k) is defined by the recurrence relations:

$$PD(k,1) = 1$$

$$PD(k,j+1) = PD(k,j) + 2n + r_k - j + 1$$

$$j = 2, \ldots, (n+r_k) \quad .$$

An element l_{ij}^k of the matrix L can be recovered from the above storage scheme as follows:

$$l_{ij}^k = VE(k, PD(k,j)+i-j) \quad .$$

This storage scheme for L^k and D^k allows us to recover l_{ij}^k from a one-dimensional array without multiplications or divisions, and thus reduces the CPU time.

8. APPLICATION TO BLOCK-ANGULAR PROGRAMS

The other important class of specially structured LP programs where the proposed algorithm seems to be most effective is the class of so-called block-angular problems with coupling columns. These problems were, perhaps, most popular in literature on specially structured LP problems, because they have simple structure as well as meaningful economic interpretation [9].

Consider the problem of

$$\min \sum_{i=1}^{N} P_i^T x_i + P_0^T x_0$$

subject to

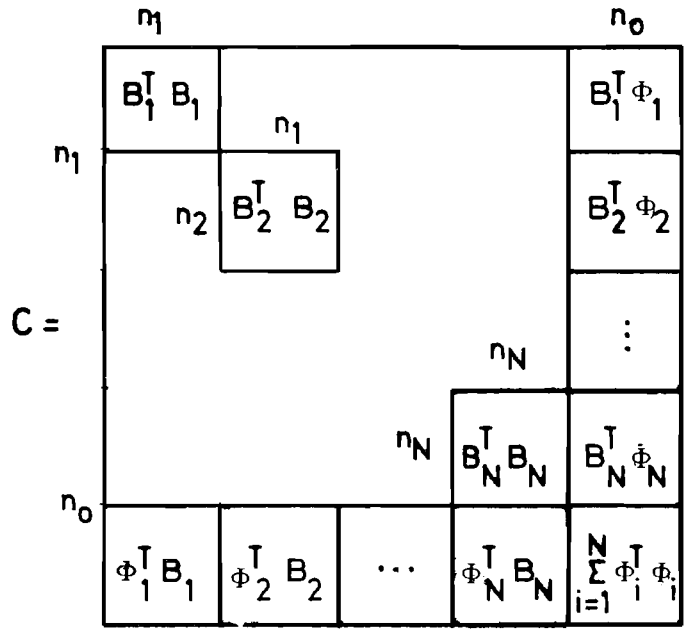
$$B_i x_i + \Phi_i x_0 = b_i \qquad (37)$$
$$i = 1,\ldots,N$$

where x_i-nonnegative n_i-vector, P_i-known vector of the same dimension, and b_i-known m_i-vector for all $i = 1,\ldots,N$. B_i and Φ_i are known matrices of appropriate size.

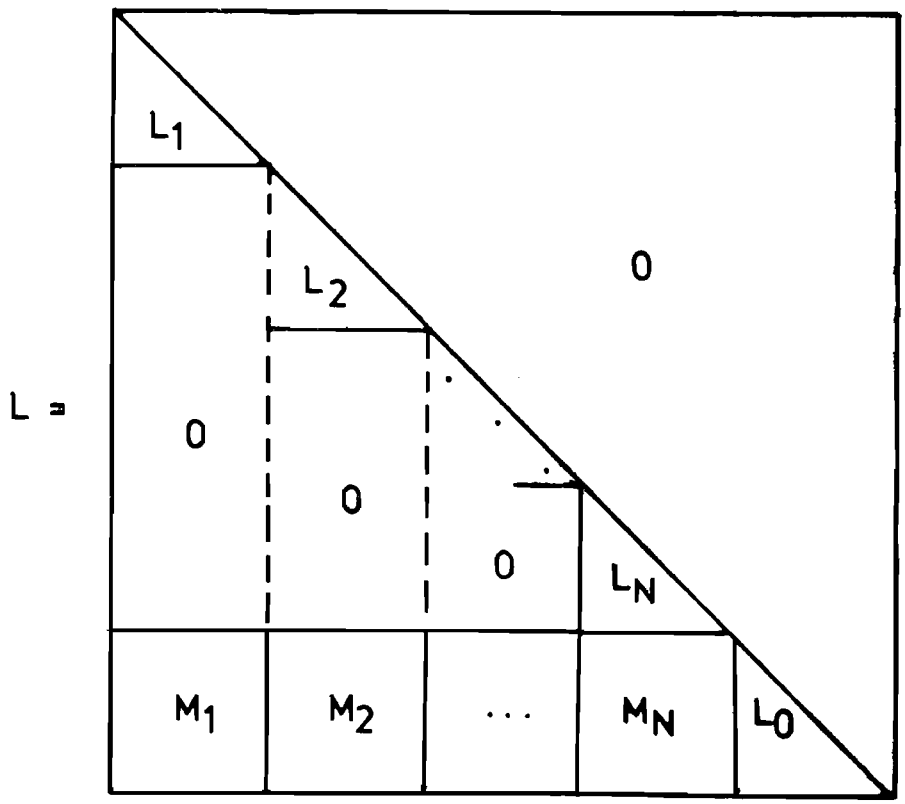
Constraints (37) may be written as one matrix constraint with the following matrix

$$A = \begin{bmatrix} B_1 & & & \Phi_1 \\ & B_2 & 0 & \Phi_2 \\ & & \ddots & \vdots \\ 0 & & & B_N \quad \Phi_N \end{bmatrix}$$

The matrix $C = A^T A$ in this case takes the form



The matrix $\bar{C} = I + A^T A$ again may be factorized as $\bar{C} = LDL^T$ where D is a non-singular diagonal matrix and L-nonsingular lower triangular matrix with the following structure



The blocks of $\bar{C}$,L and D are related by

$$I_i + B_i^T B_i = L_i D_i L_i^T$$

$$\Phi_i^T B_i = M_i D_i L_i^T \quad , \qquad i = 1,\ldots,N$$

$$I_0 + \sum_{i=1}^{N} \Phi_i^T \Phi_i = \sum_{i=1}^{N} M_i D_i M_i^T + L_0 D_0 L_0^T \quad . \tag{38}$$

It follows that the matrices $L_i, D_i, i = 1,\ldots,N$ may be computed independently, and when they are available we proceed to computation of $M_i, i = 1,\ldots,N$ and D_0, L_o.

It should be emphasized that we may use the FT-procedure for computation of the triangular factors $L_i, i = 1,\ldots,N$, and Bennett's procedure for factorization of the right-lower block of $\bar{C}$. Namely, the factors $L_i, i = 1,\ldots,N$ are easily obtainable as in (27), and L_0, D_0 are computed in two steps using the formula

$$L_0 D_0 L_0^T = [(I_0 + \sum_{i=1}^{N} \Phi_i^T \Phi_i) - \sum_{i=1}^{N} M_i D_i M_i^T] \quad .$$

Here the transformation of the expression in parentheses corresponds to the first step while the transformation of the term in brackets--to the second.

Note that at each step of the algorithm described above, only one diagonal element of C changes. Suppose that at some step a diagonal element of i-th block ($i \leq N$) changes. Then it follows from (38) that only the elements of L_0, D_i, M_i, D_0, L_0 change while all the other blocks remain as before. When a diagonal element of 0-th block changes, only the elements of L_0 and D_0 have to be modified.

The equation (10) may now be solved using the following recurrence equations.

Forward transformation:

$$Z_i = L_i^{-1} b_i \quad , \qquad i = 1,\ldots,N$$

$$Z_0 = L_0^{-1} (b_0 - \sum_{i=1}^{N} M_i Z_i)$$

Backward transformation:

$$Y_0 = L_0^{-T} D_0^{-1} Z_0$$

$$Y_i = L_i^{-T}(D_i^{-1} Z_i - M_i^T Y_0) \quad , \qquad i = N,\ldots,1 \quad .$$

Similar relations are used in solving (15). It is easy to see that during the solution process we have to keep in core memory at one time only blocks $L_i D_i M_i$. All the other blocks may be stored in a drum or a disk. So this approach represents a type of decomposition, for to compute the next "corner point" of the trajectory (i.e. the next approximation to the solution) we, in fact, have to solve successively 2(N+1) triangular systems of linear equations.

9. CONCLUSION

The method described in this paper is based on two main ideas: the use of penalty functions and application of matrix factorization techniques. The main result is that the use of a smooth penalty function allows us to find the exact solution to the original problem in a finite number of steps. The method differs from the usual implementation of penalty function methods in that at each step we now have to solve only linear systems of equations differing from each other in only one diagonal element. Numerical experiments show that the gain in speed and accuracy is tremendous in comparison with the usual implementation.

The number of steps depends very much on choice of the initial value of the penalty coefficient. The smaller Q_0 the fewer the number of steps. Ususally (for problems of medium size) the number of steps is much smaller than that of the simplex method.

The number of operations required at each step in the dynamic case grows linearly with the number of time periods. In principle, it allows us to solve very large problems, for all but one of the trapezoidal matrices may be stored on a disk or a drum. The same conclusion is true for block-diagonal programs.

REFERENCES

[1] Gill, P. and W. Murray, *Numerical Methods for Constrained Optimization*, Academic Press, London, New York, San Francisco, 1974.

[2] Polyak, B.T. and N.V. Tretyakov, On One Iterative Method for Linear Programming and Its Economic Interpretation, *Economics and Mathematical Methods*, Vol. VIII (1972), 5, (in Russian).

[3] Chebotarev, S.P., *On the Variation of Penalty Coefficient in Linear Programs, Automation and Remote Control*, No. 7, 1973.

[4] Zangwill, W.I., *Non-Linear Programming: A Unified Approach*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1969.

[5] Bennet, J.M., Triangular Factors of Modified Matrices, *Numerische Mathematik*, Vol. 7 (1965), 217-221.

[6] Forrest, J.F.H. and J.A. Tomlin, Updating Triangular Factors of the Basis to Maintain Sparsity in the Product Form Simplex Method, *Mathematical Programming 2*, 263, 278.

[7] Propoi, A.I., *On Dynamic Linear Programming*, RM-76-78, International Institute for Applied Systems Analysis, Laxenburg, Austria.

[8] Tewarson, R.P., *Sparse Matrices*, Academic Press, New York and London, 1973.

[9] Lasdon, R., *Optimization Theory for Large Systems*, The MacMillan Co, Collier-MacMillan Ltd., London, 1970.