MATHEMATICS OF COMPUTATION
VOLUME 63, NUMBER 208
OCTOBER 1994, PAGES 625–643

A COMPARISON OF BLOCK PIVOTING AND INTERIOR-POINT ALGORITHMS FOR LINEAR LEAST SQUARES PROBLEMS WITH NONNEGATIVE VARIABLES

LUÍS F. PORTUGAL, JOAQUIM J. JÚDICE, AND LUÍS N. VICENTE

ABSTRACT. In this paper we discuss the use of block principal pivoting and predictor-corrector methods for the solution of large-scale linear least squares problems with nonnegative variables (NVLSQ). We also describe two implementations of these algorithms that are based on the normal equations and corrected seminormal equations (CSNE) approaches. We show that the method of normal equations should be employed in the implementation of the predictor-corrector algorithm. This type of approach should also be used in the implementation of the block principal pivoting method, but a switch to the CSNE method may be useful in the last iterations of the algorithm. Computational experience is also included in this paper and shows that both the predictor-corrector and the block principal pivoting algorithms are quite efficient to deal with large-scale NVLSQ problems.

1. INTRODUCTION

The linear least squares problem with nonnegative variables (NVLSQ) can be stated as

$$\min_x \tfrac{1}{2}\|Ax - b\|_2^2 \quad \text{subject to } x \geq 0, \tag{1}$$

where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$ are given, $m > n$, $x \in \mathbb{R}^n$, and $\|\cdot\|_2$ represents the l_2 norm. This problem has been studied over the years and has found many applications in different areas of science and engineering [13]. By the definition of the l_2 norm, it is possible to restate the NVLSQ problem as the following quadratic program:

$$\min_x -(A^T b)^T x + \tfrac{1}{2}x^T A^T A x \quad \text{subject to } x \geq 0. \tag{2}$$

Since $A^T A$ is a positive semidefinite matrix, this program is convex, whence the Karush-Kuhn-Tucker optimal conditions constitute the following monotone linear complementary problem (LCP):

$$y = A^T A x - A^T b, \quad y \geq 0, x \geq 0, x^T y = 0. \tag{3}$$

Received by the editor March 30, 1993 and, in revised form, September 14, 1993.
1991 *Mathematics Subject Classification.* Primary 90C20; Secondary 65F50, 90C06.
Key words and phrases. Least squares problems, linear complementarity problem, quadratic programming, sparse matrices, large-scale optimization..
Support of this work has been provided by the Instituto Nacional de Investigação Científica de Portugal (INIC) under contract 89/EXA/5.

0025-5718/94 $1.00 + $.25 per page

If the matrix A has full column rank ($\text{rank}(A) = n$), the matrix A^TA is positive definite and the strictly convex program (2) and the strictly monotone LCP (3) have unique solutions for each vector b. In this paper we only deal with this case.

Owing to its equivalence to the quadratic problem (2), the unique optimal solution of the NVLSQ problem can be found by a large number of algorithms that have been designed for this type of programs. In particular, active-set methods and projected gradient algorithms can be used for this purpose. Recently, Bierlair, Toint, and Tuyttens [1] have studied the performance of projected gradient algorithms for the solution of large-scale NVLSQ problems. In another paper [2], Björck proposes an efficient implementation of the active-set method discussed in [13], which is capable of dealing with large-scale NVLSQ problems.

In [11] the performance of block principal pivoting methods for the solution of large-scale strictly convex quadratic programs with nonnegative variables has been investigated. This study has shown that this kind of algorithm is in general quite efficient for this type of programs. On the other hand, it has been shown in [17] that interior-point methods also seem to be a good alternative technique for the solution of these quadratic programs. In this paper we discuss the use of these last two procedures for the solution of large-scale NVLSQ problems. We describe two types of implementations for these algorithms that are based on the method of normal equations and on the corrected seminormal equations (CSNE) approach. We also undertake a computational study that shows that the predictor-corrector and the block pivoting algorithms are quite suitable for the solution of this type of problems and are much more efficient than the active-set method. The predictor-corrector algorithm should be implemented by using the normal equations method. On the other hand, the use of the normal equations is also advantageous in the implementation of the block principal pivoting algorithms. However, a possible switch to a CSNE approach may be profitable, particularly for ill-conditioned problems.

The organization of this paper is as follows. Sections 2 and 3 contain brief descriptions of the active-set, block pivoting, and interior-point methods. The implementations of these algorithms are discussed in §4. A technique for generating NVLSQ problems with a known optimal solution is introduced in §5. Computational experience with the algorithms is described in §6, and some concluding remarks are made in the last section of the paper.

2. Principal pivoting algorithms

Consider again the LCP (3). A point $(x, y) \in \mathbb{R}^{2n}$ is said to be a complementary solution if it satisfies

$$y = A^TAx - A^Tb, \tag{4}$$

$$x_iy_i = 0, \qquad i = 1, \ldots, n. \tag{5}$$

Next, we describe how a complementary solution can be obtained. Let F and G be two subsets of $\{1, \ldots, n\}$ such that $F \cup G = \{1, \ldots, n\}$ and $F \cap G = \varnothing$. Furthermore, consider the following column partition of the matrix A:

$$A = [A_F, A_G],$$

where $A_F \in \mathbb{R}^{m \times |F|}$, $A_G \in \mathbb{R}^{m \times |G|}$, and $|F|$, $|G|$ are the number of columns

of A_F and A_G, respectively. Based on this partition, we can rewrite (4) as

$$\begin{bmatrix} y_F \\ y_G \end{bmatrix} = \begin{bmatrix} A_F^T A_F & A_F^T A_G \\ A_G^T A_F & A_G^T A_G \end{bmatrix} \begin{bmatrix} x_F \\ x_G \end{bmatrix} - \begin{bmatrix} A_F^T b \\ A_G^T b \end{bmatrix}, \tag{6}$$

where x_F, $y_F \in \mathbb{R}^{|F|}$, x_G, $y_G \in \mathbb{R}^{|G|}$, $x = (x_F, x_G)$, and $y = (y_F, y_G)$.

A complementary basic solution is obtained by setting $x_G = 0$ and $y_F = 0$ in the conditions (6). The null variables x_i, $i \in G$, and y_i, $i \in F$, are called nonbasic, while x_i, $i \in F$, and y_i, $i \in G$, are said to be basic variables. We may compute the values of the basic variables x_F and y_G by the following procedure:

(1) Solve the unconstrained linear least squares (ULSQ) problem

$$\min_{x_F \in \mathbb{R}^{|F|}} \tfrac{1}{2}\|A_F x_F - b\|_2^2. \tag{7}$$

(2) Set

$$y_G = A_G^T(A_F x_F - b). \tag{8}$$

Hence the complementary basic solution is given by $x = (x_F, 0)$, $y = (0, y_G)$. This solution is called nondegenerate if the values of all the basic variables are nonzero. Otherwise it is said to be degenerate.

A complementary basic solution is said to be feasible if $x_F \geq 0$ and $y_G \geq 0$. In this case it is the optimal solution of the NVLSQ. Hence a complementary basic solution is infeasible if there exists at least one $i \in F$ such that $x_i < 0$ or one $i \in G$ with $y_i < 0$. If a solution is infeasible, the set of infeasibilities $H = H_1 \cup H_2$ is nonempty, where

$$H_1 = \{i \in F : x_i < 0\} \quad \text{and} \quad H_2 = \{i \in G : y_i < 0\}. \tag{9}$$

Principal pivoting algorithms are procedures that use in each iteration infeasible complementary solutions until finding a feasible complementary solution. In each iteration the sets F and G are modified according to the following rules:

$$F = F - \overline{H}_1 \cup \overline{H}_2, \qquad G = G - \overline{H}_2 \cup \overline{H}_1, \tag{10}$$

where $\overline{H}_1 \subseteq H_1$ and $\overline{H}_2 \subseteq H_2$. A principal pivoting algorithm is said to be single if the cardinality of $\overline{H}_1 \cup \overline{H}_2$ is one. Otherwise the algorithm is called a block principal pivoting algorithm.

The active-set method for the solution of the NVLSQ problem has been introduced in [2]. The algorithm can be seen as a single principal pivoting algorithm, and may be stated in the following form:

Active-set algorithm.

Step 0. Set $F = \varnothing$, $G = \{1, \ldots, n\}$, $x = 0$, and $y = -A^T b$.

Step 1. Compute

$$r = \operatorname{argmin}\{y_i : i \in G\}.$$

If $y_r < 0$, set $\overline{H}_1 = \varnothing$, $\overline{H}_2 = \{r\}$, and update F and G by (10). Otherwise stop: $x^* = x$ is the optimal solution of the NVLSQ problem.

Step 2. Compute $\overline{x}_F$ by solving (7). If $\overline{x}_F \geq 0$, set $x = (\overline{x}_F, 0)$ and go to step 3. Otherwise let r be such that

$$\theta = \frac{-x_r}{\overline{x}_r - x_r} = \min\left\{\frac{-x_i}{\overline{x}_i - x_i} : i \in F \text{ and } \overline{x}_i < 0\right\}$$

and set $x = ((1-\theta)x_F + \theta\overline{x}_F, 0)$, $\overline{H}_1 = \{r\}$, and $\overline{H}_2 = \varnothing$. Update F and G by (10) and repeat step 2.

Step 3. Compute y_G by (8) and return to step 1.

The active-set method converges to the optimal solution of the NVLSQ problem provided its matrix has full column rank [10, 13]. Murty's method [16] is another single principal pivoting algorithm that can be applied for the solution of the NVLSQ problem in this case. Given a complementary basic solution, the last infeasibility

$$r = \max\{i \in H_1 \cup H_2\}$$

is considered and the sets F and G are updated by (10), where

$$\overline{H}_1 = \begin{cases} \varnothing & \text{if } r \in G, \\ \{r\} & \text{if } r \in F, \end{cases} \qquad \overline{H}_2 = \begin{cases} \varnothing & \text{if } r \in F, \\ \{r\} & \text{if } r \in G. \end{cases}$$

The algorithm is also finite but is usually less efficient than the active-set method [10]. Murty's method can start with any sets F and G, and this can be exploited in the design of an efficient block algorithm for the solution of the NVLSQ problem. This property is not shared by the active-set method, since it requires a complementary basic solution satisfying $x_F \geq 0$ to start with.

As stated before, in single principal pivoting algorithms the sets of basic and nonbasic variables only change in one element in each iteration. Hence these algorithms are not efficient for large-scale NVLSQ problems. Block principal pivoting algorithms allow the sets of basic and nonbasic variables to change in more than one element and seem much more suited for this type of problem. The first block principal pivoting algorithm for the strictly monotone LCP is due to Kostreva [12]. However, the algorithm may cycle and has been abandoned. Júdice and Pires [9] have studied this algorithm in practice and have concluded that it is quite suitable for large-scale strictly monotone LCPs. Cycling may occur, but it is extremely rare. In a more recent paper [11] they have proposed a simple way of transforming Kostreva's algorithm into a finite procedure by incorporating Murty's method. This latter algorithm should be used as seldom as possible if the hybrid algorithm is to be efficient for large-scale problems. The idea is to control the number of infeasibilities $|H_1 \cup H_2|$, where H_1 and H_2 are given by (9), as described below.

Let x be a complementary basic solution and let $\text{ninf} = |H_1 \cup H_2|$ be the number of infeasibilities associated with this solution. Suppose we apply a step of Kostreva's algorithm ($\overline{H}_i = H_i$, $i = 1, 2$). If $|H_1 \cup H_2|$ has been reduced, then we update ninf and repeat the same procedure. Otherwise we allow $p-1$ steps of Kostreva's method for $|H_1 \cup H_2|$ to be smaller than ninf. Here, p is a small positive integer ($p \leq 10$ usually works well in practice). If after $p-1$ steps, $|H_1 \cup H_2|$ is still larger than ninf, then we use Murty's method until we find a complementary solution for which $|H_1 \cup H_2| < \text{ninf}$. This is always possible, since Murty's method is a finite procedure for finding a complementary solution with a number of infeasibilities smaller than a required value. After such a solution is found, the whole procedure is repeated. The steps of this hybrid scheme are presented below.

Block principal pivoting algorithm.

Step 0. Let $F = \varnothing$, $G = \{1, \ldots, n\}$, $x = 0$, $y = -A^T b$, $p = \overline{p} \leq 10$, $\text{ninf} = n+1$, and α be a permutation of the set $\{1, \ldots, n\}$.

Step 1. If $x_F \geq 0$ and $y_G \geq 0$, stop: $x^* = (x_F, 0)$ is the solution of the NVLSQ problem. Otherwise define H_1 and H_2 as in (9).

(1) If $|H_1 \cup H_2| < \text{ninf}$, set $\text{ninf} = |H_1 \cup H_2|$, $p = \overline{p}$, $\overline{H}_1 = H_1$, and $\overline{H}_2 = H_2$.

(2) If $|H_1 \cup H_2| \geq \text{ninf}$ and $p \geq 1$, set $p = p - 1$, $\overline{H}_1 = H_1$, and $\overline{H}_2 = H_2$.

(3) If $|H_1 \cup H_2| \geq \text{ninf}$ and $p = 0$, set

$$\overline{H}_1 = \{r\} \text{ and } \overline{H}_2 = \varnothing, \text{ if } r \in H_1,$$
$$\overline{H}_1 = \varnothing \text{ and } \overline{H}_2 = \{r\}, \text{ if } r \in H_2,$$

where r is the last element of the set $H_1 \cup H_2$ according to the order defined by α.

Update F and G by (10).

Step 2. Compute x_F and y_G by (7) and (8) and return to step 1.

Although this algorithm works well in practice for strictly monotone LCPs [11], no complexity result has been established for its performance. Other combinations of the algorithms of Murty and Kostreva have been proposed by Júdice and Pires [11]. However, this scheme has proved, in general, to be the most efficient of this type of block principal pivoting methods. We also note that the block principal pivoting algorithm cannot deal with rank-deficient linear least squares problems with nonnegative variables. In contrast, these problems can be solved by the active-set method described in this section.

3. Interior-point methods

In this section we discuss Newton approaches to solve the monotone LCP. The first algorithm is applied to the following system of nonlinear equations:

$$F_1(x, y) = XYe = 0,$$

where $F_1: S_1 \to \mathbb{R}^n$, $S_1 = \{(x, y) \in \mathbb{R}^{2n}: (x, y) \geq 0 \text{ and } A^TAx - y - A^Tb = 0\}$, $e \in \mathbb{R}^n$ is a vector of ones, and X, $Y \in \mathbb{R}^{n \times n}$ are the diagonal matrices whose diagonal elements are the components of the vectors x and y, respectively.

The procedure computes a sequence of points $\{x^k\}$ by

$$(x^{k+1}, y^{k+1}) = (x^k, y^k) + \theta_k(u^k, v^k), \tag{11}$$

where θ_k is a positive stepsize and $(u^k, v^k) \in \mathbb{R}^{2n}$. The sequence of points generated by the algorithm must belong to the set S_1. To do this, we impose the following conditions:

$$\text{(i) } (x^0, y^0) \in S_1, \quad \text{(ii) } v^k = A^TAu^k, \quad \text{(iii) } (x^{k+1}, y^{k+1}) \in S_2,$$

where

$$S_2 = \{(x, y) \in \mathbb{R}^{2n}: (x, y) \geq 0\}. \tag{12}$$

The direction (u^k, v^k) is computed by solving the linear system

$$\begin{bmatrix} Y^k & X^k \\ A^TA & -I_n \end{bmatrix} \begin{bmatrix} u^k \\ v^k \end{bmatrix} = \begin{bmatrix} -X^kY^ke + \mu_k e \\ 0 \end{bmatrix},$$

where I_n represents the identity matrix of order n and μ_k is a centralization parameter [14].

The value of μ_k should be nonnegative to assure the variables to be positive. In addition, the value of μ_k has an upper bound related to the decrease of $g(x, y) = x^T y$ in each iteration k. In fact, we have

$$\begin{aligned}\Delta^k g &= g(x^{k+1}, y^{k+1}) - g(x^k, y^k)\\ &= (x^k + \theta_k u^k)^T (y^k + \theta_k v^k) - (x^k)^T y^k\\ &= ((u^k)^T v^k)\theta_k^2 + ((x^k)^T v^k + (y^k)^T u^k)\theta_k\\ &= ((u^k)^T A^T A u^k)\theta_k^2 + (-(x^k)^T y^k + n\mu_k)\theta_k .\end{aligned}$$

Since $A^T A$ is a symmetric positive definite matrix, the condition $-(x^k)^T y^k + n\mu_k < 0$ has to hold to assure that $\Delta^k g < 0$. Thus,

$$0 \le \mu_k < \frac{(x^k)^T y^k}{n} .$$

Let $\theta_k^{\max}$ be the largest value of θ_k such that (iii) holds. This parameter must also verify

$$0 < \theta_k < \min(\theta_k^{\max}, \hat{\theta}_k),$$

where

$$\hat{\theta}_k = \frac{(x^k)^T y^k - n\mu_k}{(u^k)^T v^k} .$$

By choosing θ_k in this way, a monotone decrease for $g(x, y)$ is always guaranteed in each iteration.

The second approach includes the application of Newton's method to the system of nonlinear equations

$$F_2(x, y) = \begin{bmatrix} XYe \\ A^T AXe - Ye - A^T b \end{bmatrix} = 0,$$

where (x, y) belongs to the set S_2 defined by (12). Hence in this approach we do not force the iterates x^k and y^k to satisfy the linear constraints $A^T Ax - y - A^T b = 0$. In this case, the Newton direction (u^k, v^k) is obtained by the solution of the system of linear equations

$$\begin{bmatrix} Y^k & X^k \\ A^T A & -I_n \end{bmatrix} \begin{bmatrix} u^k \\ v^k \end{bmatrix} = \begin{bmatrix} -X^k Y^k e + \mu_k e \\ -A^T A X^k e + Y^k e + A^T b \end{bmatrix} . \tag{13}$$

By following arguments similar to those presented before, we reach the same lower and upper bounds for the value of μ_k. However, in this case the value of the parameter θ_k has to verify

$$\begin{aligned} &0 < \theta_k \le \theta_k^{\max} && \text{if } (u^k)^T v^k \le 0,\\ &0 < \theta_k < \min(\theta_k^{\max}, \hat{\theta}_k) && \text{if } (u^k)^T v^k > 0, \end{aligned} \tag{14}$$

where $\hat{\theta}_k$ is defined as before.

Our computational experience has shown that the condition $\theta_k^{\max} < \hat{\theta}_k/2$ is usually fulfilled. This seems a good reason to choose $\theta_k < \theta_k^{\max}$. Hence, we follow the recommendation stated in [4] and set $\theta_k = 0.99995 \times \theta_k^{\max}$ in each iteration k. As in [4], the parameter μ_k is given by

$$\mu_k = \frac{(x^k)^T y^k}{n^2} . \tag{15}$$

The solution of the linear system (13) is obtained in two major steps. First, we compute u^k by solving the linear system

$$(A^TA + (X^k)^{-1}Y^k)u^k = (X^k)^{-1}\mu_k e - A^TAX^k e + A^Tb,$$

where $(A^TA + (X^k)^{-1}Y^k)$ is a symmetric positive definite matrix. This is in turn equivalent to solving the ULSQ problem

$$\min_{u^k \in \mathbb{R}^n} \frac{1}{2} \left\| \begin{bmatrix} A \\ (X^k)^{-1/2}(Y^k)^{1/2} \end{bmatrix} u^k - \begin{bmatrix} b - AX^k e \\ (X^k)^{-1/2}(Y^k)^{-1/2}\mu_k e \end{bmatrix} \right\|_2^2, \tag{16}$$

where $(X^k)^{1/2}$, $(Y^k)^{1/2} \in \mathbb{R}^{n\times n}$ are diagonal matrices whose diagonal elements are the square roots of the components of the vectors x^k and y^k. In a second step we compute v^k by

$$v^k = -Y^k e + (X^k)^{-1}\mu_k e - (X^k)^{-1}Y^k u^k. \tag{17}$$

The steps of the resulting Newton's algorithm are stated below.

Newton's algorithm.

Step 0. Let $TOL1$ and $TOL2$ be two tolerances for zero, $k = 0$, and $(x_0, y_0) > 0$.

Step 1. Compute u^k and v^k by (16) and (17), respectively.

Step 2. Set $\theta_k^{\max} = \min\{\theta_k^1, \theta_k^2\}$, where

$$\theta_k^1 = \min\left\{-\frac{x_i^k}{u_i^k} : i = 1, \ldots, n \text{ and } u_i^k < 0\right\},$$

$$\theta_k^2 = \min\left\{-\frac{y_i^k}{v_i^k} : i = 1, \ldots, n \text{ and } v_i^k < 0\right\}.$$

Set $\theta_k = 0.99995 \times \theta_k^{\max}$ and update (x^{k+1}, y^{k+1}) by (11).

Step 3. If $(x^{k+1})^T y^{k+1} < TOL1$ and $\|A^TAx^{k+1} - A^Tb - y^{k+1}\|_2 < TOL2$ stop: $x^* = x^{k+1}$ is an approximate solution of the problem NVLSQ. Otherwise set $k := k + 1$ and return to step 1.

Some authors ([4] and [15]) have suggested the use of a predictor-corrector direction to improve Newton's algorithm. In a first step the direction (u^k, v^k) is predicted by

$$\begin{bmatrix} Y^k & X^k \\ A^TA & -I_n \end{bmatrix} \begin{bmatrix} u^k \\ v^k \end{bmatrix} = \begin{bmatrix} -X^kY^ke \\ -A^TAX^ke + Y^ke + A^Tb \end{bmatrix}. \tag{18}$$

To correct the direction, we compute z^k and w^k by

$$(z^k, w^k) = (u^k, v^k) + (\overline{u}^k, \overline{v}^k),$$

where $(\overline{u}^k, \overline{v}^k) \in \mathbb{R}^{2n}$ is found by

$$\begin{bmatrix} Y^k & X^k \\ A^TA & -I_n \end{bmatrix} \begin{bmatrix} \overline{u}^k \\ \overline{v}^k \end{bmatrix} = \begin{bmatrix} -U^kV^ke + \mu_k e \\ 0 \end{bmatrix}$$

with $U^k = \operatorname{diag}(u^k)$ and $V^k = \operatorname{diag}(v^k)$. The direction (z^k, w^k) may also be computed by solving the linear system

$$\begin{bmatrix} Y^k & X^k \\ A^TA & -I_n \end{bmatrix} \begin{bmatrix} z^k \\ w^k \end{bmatrix} = \begin{bmatrix} -X^kY^ke + \mu_k e - U^kV^ke \\ -A^TAX^ke + Y^ke + A^Tb \end{bmatrix}. \tag{19}$$

Using similar arguments as above, one can see that a decrease in $g(x, y)$ is guaranteed if the value of the parameter μ_k is restricted by

$$0 \leq \mu_k < \frac{(x^k)^T y^k + (u^k)^T v^k}{n}$$

and the value of θ_k satisfies (14), where

$$\hat{\theta}_k = \frac{(x^k)^T y^k + (u^k)^T v^k - n\mu_k}{(z^k)^T w^k}.$$

A good practical criterion for the choice of μ_k has been introduced by Lustig, Marsten, and Shanno [14] and simply consists of setting

$$\mu_k = \frac{(x^k + \theta_k u^k)^T (y_k + \theta_k v^k)}{n^2}, \tag{20}$$

where θ_k is computed as in Newton's method. This value of μ_k is used in (19) to obtain the predictor-corrector direction (z^k, w^k). It is important to notice that this choice of μ_k does not guarantee a decrease in $g(x, y)$, but works well in practice. After such a direction is found, the new point (x^{k+1}, y^{k+1}) is obtained so that all variables remain positive. The steps of the resulting procedure are stated below.

Predictor-corrector algorithm.

Step 0. Let $TOL1$ and $TOL2$ be two tolerances for zero, $k = 0$, and $(x_0, y_0) > 0$.

Step 1. Compute u^k and v^k by solving the linear system (18).

Step 2. Set $\theta_k^{\max} = \min\{\theta_k^1, \theta_k^2\}$, where

$$\theta_k^1 = \min\left\{-\frac{x_i^k}{u_i^k} : i = 1, \ldots, n \text{ and } u_i^k < 0\right\},$$

$$\theta_k^2 = \min\left\{-\frac{y_i^k}{v_i^k} : i = 1, \ldots, n \text{ and } v_i^k < 0\right\}.$$

Set $\theta_k = 0.99995 \times \theta_k^{\max}$.

Step 3. Let μ_k be given by (20). Compute the direction (z^k, w^k) by solving (19).

Step 4. Set $\theta_k^{\max} = \min\{\theta_k^1, \theta_k^2\}$, where

$$\theta_k^1 = \min\left\{-\frac{x_i^k}{z_i^k} : i = 1, \ldots, n \text{ and } z_i^k < 0\right\},$$

$$\theta_k^2 = \min\left\{-\frac{y_i^k}{w_i^k} : i = 1, \ldots, n \text{ and } w_i^k < 0\right\}.$$

Set $\theta_k = 0.99995 \times \theta_k^{\max}$ and $(x^{k+1}, y^{k+1}) = (x^k, y^k) + \theta_k(z^k, w^k)$.

Step 5. If $(x^{k+1})^T y^{k+1} < TOL1$ and $\|A^T A x^{k+1} - A^T b - y^{k+1}\|_2 < TOL2$ stop: $x^* = x^{k+1}$ is an approximate solution of the problem NVLSQ. Otherwise set $k := k + 1$ and return to step 1.

As before, it is important to note that the solution of the linear systems (18) and (19) can be stated in terms of unconstrained linear least squares problems. In fact, (18) is equivalent to

$$\min_{u^k \in \mathbb{R}^n} \frac{1}{2} \left\| \begin{bmatrix} A \\ (X^k)^{-1/2}(Y^k)^{1/2} \end{bmatrix} u^k - \begin{bmatrix} b - AX^k e \\ 0 \end{bmatrix} \right\|_2^2 \tag{21}$$

and

$$v^k = -Y^k e - (X^k)^{-1} Y^k u^k . \tag{22}$$

In a similar way, the linear system (19) reduces to the following ULSQ problem:

$$\min_{z^k \in \mathbb{R}^n} \frac{1}{2} \left\| \begin{bmatrix} A \\ (X^k)^{-1/2}(Y^k)^{1/2} \end{bmatrix} z^k - \begin{bmatrix} b - AX^k e \\ (X^k)^{-1/2}(Y^k)^{-1/2}(\mu_k - U^k V^k)e \end{bmatrix} \right\|_2^2 \tag{23}$$

and

$$w^k = -Y^k e + (X^k)^{-1}(-Y^k z^k + \mu_k e - U^k V^k e) . \tag{24}$$

It follows from the description of the predictor-corrector algorithm that each iteration of the procedure requires the solution of two ULSQ problems with the same matrix. Hence the computational effort of this algorithm per iteration is larger than that required by the simple version of Newton's method. However, as stated in [14], the number of iterations is usually smaller for the predictor-corrector algorithm, and this compensates for the larger computational effort of each predictor-corrector iteration. Furthermore, the solution obtained by the predictor-corrector strategy is usually more accurate. This explains why we have chosen the predictor-corrector algorithm in our experimentations. Finally, we note that the predictor-corrector algorithm is also capable of dealing with rank-deficient linear least squares problems with nonnegative variables.

4. Implementation issues

In this section we discuss the implementations of the three algorithms introduced in the previous sections for the solution of large-scale linear least squares problems with nonnegative variables (NVLSQ). The implementation of the active-set method has been discussed in [2]. In the first step of this procedure, a so-called *analyze* phase is performed in which a permutation of the columns of the matrix A is found by applying the minimum-degree strategy to the matrix $A^T A$ [8]. Hence, the algorithm seeks a solution of the NVLSQ problem whose matrix A contains the columns of the original matrix in the order achieved by the minimum-degree procedure. This phase is important, since we are able to control the amount of fill-in that occurs during the so-called *factor* phase. After the *analyze* phase, the permuted matrix A is stored by rows in a collection of vectors scheme. Each iteration of the active-set method requires the computation of the vectors x_F and y_G that are given by the formulas (7) and (8). The *factor* and *solve* phases are to find these vectors. In the *factor* phase, the QR factorization of A_F is computed. Since in the active-set algorithm the set F is modified in exactly one element in each iteration, it is advisable to update the QR factorization instead of computing it from scratch. As stated in [2], there are efficient procedures to perform this task when an index is taken or added to the set F. In the *solve* phase the corrected seminormal equations

(CSNE) method [3] is applied to find the vector x_F, that is, x_F is computed by

$$R^T R x_F = A_F^T b, \tag{25}$$

where $A_F = Q\begin{bmatrix} R \\ 0 \end{bmatrix}$. As stated in [3], a step of an iterative refinement procedure is included in this method. We also note that the orthogonal matrix Q does not have to be stored, and this leads to important storage savings. The *solve* phase terminates by computing the vector y_G according to the formula (8). It is important to notice that y_G only requires the data structure of the matrix A, the vector b, and the vector x_F that has been previously computed in this phase.

The formulas (7) and (8) are also required to get the information that is necessary in each iteration of the block principal pivoting algorithm. However, in this case modifications of more than one element are allowed in the set F. So it may be better in many cases to compute the QR factorization from scratch according to the scheme described in [7], instead of updating it from the previous iteration. Extensive computational experience reported in [18] has led to a heuristic rule for deciding whether an updating or a computation of the QR factorization should be made. In this rule we compute the quantity

$$\lambda = \frac{|\overline{H}_1 \cup \overline{H}_2|}{|F - \overline{H}_1 \cup \overline{H}_2|},$$

where $\overline{H}_1$ and $\overline{H}_2$ are the sets mentioned in §2. If $\lambda > 0.2$, then it is cheaper to compute the QR factorization from scratch. Otherwise an update of the factorization is advisable. This update consists of performing $|\overline{H}_1 \cup \overline{H}_2|$ modifications of one element. Apart from this difference, the implementation of the block pivoting algorithm is similar to the active-set method. Hence, the implementation contains three phases, namely *analyze, factor*, and *solve* with purposes similar to those presented before.

It is also possible to design another implementation of the block pivoting algorithm that is based on the method of normal equations. As before, an *analyze* phase is first applied, where a permutation of the columns of A is found by using the minimum-degree procedure for the matrix A. Then the permuted matrix A is stored by rows in a collection of vectors scheme. In the *factor* phase the Cholesky decomposition of $A_F^T A_F$ is computed, that is, an upper triangular matrix R is found such that $A_F^T A_F = R^T R$. Then this matrix R is used to find the vector x_F according to the formula (25). The formula (8) is used to compute the vector y_G in the same way as described before. These two calculations constitute the *solve* phase.

It is nowadays widely accepted that the method of normal equations is cheaper than the CSNE method, but the latter procedure is able to find solutions with better accuracy. If we look cautiously at the block pivoting method, we realize that precision is not too important in all the iterations but the last. In fact, it is only necessary to known whether x_i, $i \in F$, and y_j, $j \in G$, are negative or not in each iteration of the algorithm. So, it seems that the implementation based on the normal equations is more appropriate for the block algorithm. A switch to an implementation based on the CSNE method is easy to perform and may be interesting in the last iterations, particularly for ill-conditioned least squares problems. In the last section of this paper we present computational

experience with both the implementations of the block algorithm which will support these claims.

A last implementation issue for the block method is concerned with the choice of the permutation set α that has to be defined a priori. The choice of this set is not important for the modifications of more than one element, but has an important effect on the number of iterations when many modifications of one element take place in the set F. Extensive computational experience has shown that this type of modifications occurs rarely in practice. Hence the permutation given by the minimum-degree algorithm is sufficient for our purposes. We suggest in [18] other heuristic rules that may be worthwhile in case too many one-element modifications of the set F take place.

To conclude this section, we briefly describe the implementation of the predictor-corrector algorithm. The information that is required in each iteration of this algorithm is found in the formulas (21)–(24). The main computational effort of this algorithm consists in the solution of the damped linear least squares problem of the form

$$\min_{z\in\mathbb{R}^n} \tfrac{1}{2}\|B_k z - c\|_2^2,$$

where

$$B_k = \begin{bmatrix} A \\ (X^k)^{-1/2}(Y^k)^{1/2} \end{bmatrix} \tag{26}$$

and X^k, Y^k are diagonal matrices with positive diagonal elements. Since

$$B_k^T B_k = A^T A + (X^k)^{-1} Y^k,$$

the matrices $A^T A$ and $B_k^T B_k$ only differ in the diagonal elements. Hence, the fill-in that occurs in the QR factorization of B_k is the same that would occur if the QR factorization of A were computed. This suggests to apply an *analyze* phase to the matrix A and find a permutation of the columns of this matrix. As before, this phase terminates by storing the permuted matrix A in a collection of vectors scheme.

Each iteration of the predictor-corrector algorithm contains a *factor* phase in which the QR factorization of the matrix B_k (26) or the Cholesky factorization of the matrix $B_k^T B_k$ is computed. These factorizations are used in the *solve* phase to compute the vectors u^k and z^k given by (18) and (19) by using the CSNE method or the method of normal equations, respectively. The vectors v^k and w^k given by (22) and (24) are quite simple to compute after the vectors u^k and z^k have been calculated.

We have briefly described two possible implementations of the predictor-corrector algorithm. By looking carefully at the *factor* phases of these implementations, we immediately come to the conclusion that the gap between the computational efforts of the two implementations is much bigger in this case than in the block algorithm. In fact, the CSNE approach requires the computation of the QR factorization of the damped matrix B_k that has $m + n$ rows and n columns. Recall that in the block method the matrix A_F has $|F|$ columns and m rows, whence the number of rows and columns of the matrix may be much smaller in this latter case. So, it seems that the use of normal equations in the predictor-corrector is even more advisable than in the block

pivoting algorithm. Computational experience presented in the last section will confirm exactly this statement and will reject the use of the CSNE approach for the predictor-corrector algorithm.

To finish the description of the implementation of the predictor-corrector algorithm, we discuss the choice of the tolerances $TOL1$ and $TOL2$. Extensive computational experience has shown that the feasibility tolerance $TOL2$ should not be too small. If ε_M is the machine epsilon, then $TOL2 \simeq n\sqrt{\varepsilon_M}$ is usually a good choice. On the other hand, the gap tolerance $TOL1$ may be chosen quite small. We have chosen in our experiments $TOL1 \simeq n\varepsilon_M$. This choice of $TOL1$ usually leads to accurate solutions even when the normal equations approach is used. This will also be confirmed by the computational results presented in the last section of this paper.

5. Generation of test problems

In this section we propose a technique to generate large-scale linear least squares test problems with nonnegative variables (NVLSQ) with a known optimal solution. Consider the NVLSQ (1) problem and its associated LCP (3). Furthermore, assume that the vectors $\bar{x}$ and $\bar{y}$ are given, and consider the following index sets:

$$\begin{aligned} F &= \{i: \bar{x}_i > 0 \text{ and } \bar{y}_i = 0\}, \\ G &= \{i: \bar{x}_i = 0 \text{ and } \bar{y}_i > 0\}, \\ D &= \{i: \bar{x}_i = 0 \text{ and } \bar{y}_i = 0\}. \end{aligned} \tag{27}$$

The procedure to be described in this section attempts to find a vector $b \in \mathbb{R}^m$ such that $(\bar{x}, \bar{y})$ is the unique solution of the LCP, where A is a given sparse matrix of order $m \times n$.

Let F, G, and D be the sets defined by (27), and consider the following partition of the matrix A:

$$A = [A_F, A_G, A_D],$$

where $A_S \in \mathbb{R}^{m \times |S|}$ is the submatrix of A containing all the columns of A whose indices belong to the set S, and $|S|$ represents the number of elements of S. We can also write the vectors $\bar{x}$ and $\bar{y}$ in the form

$$\bar{x} = \begin{bmatrix} \bar{x}_F \\ 0 \\ 0 \end{bmatrix}, \qquad \bar{y} = \begin{bmatrix} 0 \\ \bar{y}_G \\ 0 \end{bmatrix}.$$

Hence, the vectors $\bar{x}$ and $\bar{y}$ satisfy the following equalities:

$$\begin{cases} A^T z = \bar{y}, \\ z = A_F \bar{x}_F - b. \end{cases}$$

Therefore, the vector b can be obtained in two steps, by first computing the vector z satisfying the system $A^T z = \bar{y}$, and then setting

$$b = A_F \bar{x}_F - z. \tag{28}$$

Since $m > n$, the system $A^T z = \bar{y}$ has an infinite number of solutions. Hence, we are satisfied with the minimum l_2 norm solution of this system, which is the unique optimal solution of the following optimization problem:

$$\min_{z \in \mathbb{R}^m} \|z\|_2 \quad \text{subject to } A^T z = \bar{y}.$$

This problem is in turn equivalent to the following set of equations:

$$A^T A\lambda = \overline{y}, \tag{29}$$

$$z = A\lambda. \tag{30}$$

We conclude that the vector b can be found by using the equalities (28), (29), and (30). It is now obvious that we wish to get a quite accurate vector b. Therefore, precision is quite important in the solution of the system (29), and we recommend the CSNE method with iterative refinement [3] to be used in this context. By using this procedure and the formulas (28) and (30), we obtain the following algorithm for the computation of the vector b:

Algorithm GENB.

Step 1. Compute the sparse QR decomposition of A and let P be the associated permutation matrix.

Step 2. Compute λ by solving the linear systems

$$R^T\overline{\lambda} = P^T\overline{y} \quad \text{and} \quad R\lambda = \overline{\lambda}.$$

Step 3. Compute

$$r = P^T\overline{y} - P^T A^T AP\lambda.$$

Step 4. Compute δ_λ as follows:

$$R^T\overline{\delta}_\lambda = r \quad \text{and} \quad R\delta_\lambda = \overline{\delta}_\lambda$$

and δ_r by setting $\delta_r = -P^T A^T AP\delta_\lambda$.

Step 5. Correct λ and r by

$$\lambda = \lambda + \delta_\lambda, \qquad r = r + \delta_r.$$

Step 6. If $\|r\|_2$ is not sufficiently small, go to step 4. Otherwise compute

$$b = A_F\overline{x}_F - AP\lambda$$

and stop.

The importance of this generator lies in the possiblity of generating NVLSQ test problems with some features that are important in the study of the efficiency of algorithms. In fact, by simple choices of the values of the variables $\overline{x}_i$ and $\overline{y}_i$ and the sets F, G, and D associated with the optimal solution of the NVLSQ problem, we can generate test problems with the following characteristics:

(i) Badly- or well-scaled optimal solutions. This is related to the values of the quantities

$$\frac{\max_i |\overline{x}_i|}{\min_i |\overline{x}_i|} \quad \text{and} \quad \frac{\max_i |\overline{y}_i|}{\min_i |\overline{y}_i|}.$$

(ii) Small or large optimal active-sets; this depends on the number of elements of the set $G \cup D$.

(iii) Nondegenerate or degenerate optimal solution (that is, strict complementarity holds or not at the optimal solution); this depends on the set D to be empty or not, respectively.

6. Computational experience

In this section we report on computational experience with the active-set (AS), block principal pivoting (BLOCK) ($\bar{p} = 3$), and predictor-corrector (PRECOR) algorithms on the solution of some linear least squares problems with nonnegative variables (NVLSQ). As stated before, we have considered two implementations for the BLOCK and PRECOR algorithms, differing in the procedure that is used to solve the required ULSQ problems in each iteration. The versions 1 of these algorithms (BLOCK1, PRECOR1) employ a CSNE approach for the solution of the ULSQ problems, while the method of normal equations is used in their versions 2 (BLOCK2, PRECOR2). The implementations of the algorithms have been coded in FORTRAN 77 and have been tested on a SUN SPARC station SLC, whose machine epsilon is $\varepsilon_M \simeq 10^{-16}$.

We have considered two types of test problems in our experiments. In the first category (problems TP) the elements of the matrix A have been randomly generated. The vector b has been generated according to the technique described in the previous section, where the positive components of the solution vectors $\bar{x}$ and $\bar{y}$ have been set equal to one. The second set of test problems contains four least squares problems of the Harwell-Boeing collection [6]. In these test problems, the matrix A and the right-hand side vector b are given, whence we have only to request the values of the variables x to be nonnegative in order to get the desired NVLSQ problems. Table 1 contains information about all these test problems under the following headings:

m–number of rows of A,
n–number of columns of A,
nza–number of nonzero elements of A,
nzr–number of nonzero elements of the matrix R of the QR factorization of A.

Table 1. Test problems

	m	n	nza	nzr
TP1	3000	250	6065	3602
TP2	3000	500	18970	9154
TP3	3000	750	5852	19340
TP4	3000	1000	38737	39862
TP5	500	250	2355	2366
TP6	1000	250	1999	2182
TP7	2000	250	2705	2268
TP8	3000	250	10638	2492
TP9	4000	250	12547	2305
TP10	1000	250	4006	2165
TP11	1000	250	2119	945
WELL1033	1033	320	4732	2261
ILLC1033	1033	320	4732	2261
WELL1850	1850	712	8758	6749
ILLC1850	1850	712	8758	6749

TABLE 2. Number of iterations to solve problem TP3 for different values of $|F|$

$\|F\|$	AS	BLOCK	PRECOR
90	92	4	26
195	203	5	25
200	204	5	22
285	291	5	25
375	381	4	21
550	556	4	21

We have stressed before the importance of the number of elements $|F|$ of the set F associated with the unique optimal solution of the NVLSQ problem. This value $|F|$ represents the number of constraints that are inactive at the optimal solution. In our first experiment we have investigated the role that this value plays in the behavior of the algorithms. To do this, we have used the technique described in the previous section to generate six different instances of the test problem TP3 differing in the value of $|F|$ associated with the optimal solution of the NVLSQ problem. In Table 2 we display the number of iterations that the algorithms AS, BLOCK, and PRECOR take to solve these six test problems. As expected, the results indicate that the number of iterations of the AS algorithm is always greater than $|F|$. The dependence on this value is the main drawback of the active-set algorithm and discourages its use for the solution of large-scale NVLSQ problems. In fact, since it is impossible to know a priori the number of constraints that are inactive at the optimal solution, this number may be quite large for large-scale NVLSQ problems with thousands of variables. In this case the active-set algorithm takes too long to find the desired optimal solution. In contrast, the algorithms BLOCK and PRECOR do not seem to be influenced by this value $|F|$ and are much more appropriate for the solution of large-scale NVLSQ problems.

In our second experiment, we have investigated the importance of the number of variables on the performance of the algorithms. To do this, we have generated four test problems, where m is fixed ($m = 3000$) and n takes four different values. In all these test problems the vector b has been generated by the technique described in the previous section, where the sets F, G, and D associated with the unique optimal solution satisfy

$$|F| = \frac{n}{2}, \quad |G| = \frac{n}{4}, \quad |D| = \frac{n}{4}.$$

The results presented in Table 3 (next page) lead to the following conclusions:

(i) The number of iterations of the active-set method increases with n. This confirms the results of our first experiment, since $|F|$ increases with n.

(ii) The number of iterations of the BLOCK and PRECOR algorithms do not seem to be influenced by an increase in the value of n.

(iii) The gap between the CPU time of the versions 1 and 2 of the BLOCK algorithm increases with n. The situation is much more dramatic for the PRECOR algorithm, where its version 1 is not by any means competitive with the version 2.

TABLE 3. Solution of NVLSQ problems with fixed m and different values of n

TP	n	AS		BLOCK			PRECOR		
		IT	TIME	IT	TIME		IT	TIME	
					BLOCK1	BLOCK2		PRECOR1	PRECOR2
TP1	250	125	7.8	4	3.04	1.01	13	27.34	2.31
TP2	500	250	52.44	3	12.77	2.71	14	140.00	7.69
TP3	750	383	59.44	4	21.02	3.85	22	330.06	20.65
TP4	1000	500	235.15	4	51.48	22.10	16	> 500	50.09

TABLE 4. Solution of NVLSQ problems with fixed n and different values of m

TP	m	AS		BLOCK			PRECOR		
		IT	TIME	IT	TIME		IT	TIME	
					BLOCK1	BLOCK2		PRECOR1	PRECOR2
TP5	500	131	3.90	6	1.58	0.78	21	14.77	1.71
TP6	1000	127	4.14	3	1.65	0.34	23	23.47	1.71
TP7	2000	125	9.47	2	3.19	0.41	13	32.87	1.61
TP8	3000	125	14.50	2	6.03	0.50	13	51.17	2.17
TP9	4000	125	17.60	2	8.41	0.52	13	70.72	2.28

(iv) The version 2 of the PRECOR algorithm and both versions of the BLOCK algorithm are more efficient than the active-set algorithm, and the gap increases with n.

The effect of an increase in the number m of equations on the performance of the algorithms has been studied in our third experiment. We have considered for a fixed value of n ($n = 250$) five problems that differ in the value of m. The results displayed in Table 4 indicate that the number m does not play an important role on the number of iterations of the three algorithms. As expected, the CPU time for the versions 2 of the BLOCK and PRECOR algorithms does not seem to be influenced by an increase in the number m of equations. In contrast, the versions 1 of the BLOCK and PRECOR algorithms and the active-set method seem to be affected by an increase in the value of m. Again, the effect is much stronger in the PRECOR algorithm.

As a final conclusion of these three experiments, we can claim that the BLOCK and PRECOR algorithms are quite suited for large-scale NVLSQ problems if they are implemented by using the method of normal equations. The CSNE approach can still be useful in the BLOCK algorithm, but should not be employed in the PRECOR algorithm. The active-set method is not competitive with the BLOCK and PRECOR algorithms when n is sufficiently large. These conclusions are confirmed by the results displayed in Table 5 of the solution of the NVLSQ problems taken from the Harwell-Boeing collection.

It is well known that the orthogonal factorizations usually lead to more accurate optimal solutions for the unrestricted linear least squares problem than the method of normal equations. The condition number of the matrix of the linear

TABLE 5. Solution of Harwell-Boeing NVLSQ problems

	AS		BLOCK			PRECOR		
	IT	TIME	IT	TIME		IT	TIME	
				BLOCK1	BLOCK2		PRECOR1	PRECOR2
WELL1033	283	18.67	11	10.38	1.76	20	33.96	1.98
ILLC1033	201	9.99	10	6.10	0.80	18	31.07	1.74
WELL1850	635	88.31	10	47.43	6.98	25	209.58	7.16
ILLC1850	438	52.65	9	31.86	3.69	22	185.61	6.45

TABLE 6. Importance of condition number on the solution of NVLSQ problems

TP	cond(A_F)	AS	BLOCK1	BLOCK2	PRECOR1	PRECOR2
TP1	1.00E + 00	1.00E-15	2.00E-16	8.00E-14	1.00E-16	3.00E-16
TP10	1.00E + 02	1.00E-14	9.00E-15	1.00E-12	2.00E-15	2.00E-15
TP11	1.00E + 05	8.00E-11	4.00E-12	4.00E-08	5.00E-13	4.00E-13
TP6	1.00E + 06	4.00E-07	2.00E-07	8.00E-05	6.00E-12	6.00E-12

least squares problem is the most important factor for the choice of orthogonal factorizations. In fact, the method of normal equations may run into some numerical difficulties for ill-conditioned problems. Based on the observations, we have decided to investigate the importance of the condition number on the accuracy of the optimal solution found by the algorithms discussed in this paper. To do this, we have generated problems with four different LINPACK estimates [5] for the condition number of the submatrix A_F associated with the inactive constraints of the optimal solution of the NVLSQ problems. Each problem has been solved by the active-set method and by the two versions of the BLOCK and PRECOR algorithms. For each problem we have calculated the relative error of the computed solution x^*, which is given by

$$\frac{\|x^* - \bar{x}\|_2}{\|\bar{x}\|_2},$$

where $\bar{x}$ is the exact optimal solution that is fixed in the generator described in the previous section. As before, $|F| = \frac{n}{2}$, $|G| = \frac{n}{4}$, $|D| = \frac{n}{4}$ are the sets of indices in the generation of the test problems. The results displayed in Table 6 show that the accuracy of the solutions is adversely affected by an increase in the condition number of A_F. As expected, the version 2 of the BLOCK method, that is based on the method of normal equations, is much more influenced by the condition number than its version 1 that relies on the CSNE method. This version provides a more accurate solution than the active-set method. This is also understandable, since updatings of the QR factorization are performed, while in general the QR factorization is computed from scratch in the BLOCK method. However, surprisingly or not, the two versions of the PRECOR algorithm have given the most accurate solutions, and we can see no substantial difference between them. So these results support our recommendation for the use of the normal equations in the implementation of the PRECOR algorithm.

However, a switch from the normal equations to the CSNE approach may be appropriate in the BLOCK method in the last iterations, particularly for the solution of ill-conditioned NVLSQ problems.

7. Conclusions

We have investigated the use of block principal pivoting and predictor-corrector algorithms for the solution of large-scale linear least squares problems with nonnegative variables (NVLSQ). This study has shown that both algorithms are quite efficient for this type of optimization problems. The predictor-corrector should be implemented by using the method of normal equations for the solution of the unrestricted linear least squares problem that are requested in each iteration of the algorithm. This type of approach should also be used in the implementation of the block principal pivoting algorithm, but a switch to the corrected seminormal equations (CSNE) method may be useful in the last iterations of the procedure.

It is possible to generalize the block principal pivoting and the predictor-corrector algorithms for the solution of linear least squares problems with bound constraints. The solution of this type of problems seems to be worthwhile in the design of sequential techniques for the solution of nonlinear least squares problems with bound constraints. These are two topics of current research.

Bibliography

1. M. Bierlair, P. Toint, and D. Tuyttens, *On iterative algorithms for linear least-squares problems with bound constraints*, Linear Algebra Appl. **143** (1991), 111–143.
2. Å. Björck, *A direct method for sparse least-squares problems with lower and upper bounds*, Numer. Math. **54** (1988), 19–32.
3. ———, *Algorithms for linear least-squares problems*, Computer Algorithms for Solving Linear Algebraic Equations (E. Spedicato, ed.), NATO Adv. Sci. Inst. Ser. F: Comput. and Systems Sci., vol. 77, Springer, Berlin and New York, 1992, pp. 57–92.
4. J. Carpenter, I. Lustig, J. Mulvey, and D. Shanno, *Higher order predictor-corrector interior point methods with applications to quadratic objectives*, Tech. Report Rutcor Research Report #67.90, 1990.
5. A. Cline, C. Moler, G. Stewart, and J. Wilkinson, *An estimate for the condition number of a matrix*, SIAM J. Numer. Anal. **16** (1979), 368–675.
6. I. Duff, R. Grimes, and J. Lewis, *Sparse matrix test problems*, ACM Trans. Math. Software **15** (1989), 1–14.
7. A. George and M. Heath, *Solution of sparse linear least-squares problems using Givens rotations*, Linear Algebra Appl. **34** (1980), 69–83.
8. A. George and J. Liu, *Computer solution of large sparse symmetric positive definite systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.
9. J. Júdice and F. Pires, *Bard-type methods for the linear complementarity problem with symmetric positive definite matrices*, IMA J. Appl. Math. **2** (1988/89), 23–56.
10. ———, *Direct methods for convex quadratic programs subject to box constraints*, Investigação Operacional **9** (1989), 51–68.
11. ———, *Solution of large-scale strictly convex linear complementarity problems*, Investigação Operacional **11** (1991), 31–51.
12. M. Kostreva, *Block pivot methods for solving the complementarity problem*, Linear Algebra Appl. **21** (1978), 207–215.

13. C. Lawson and R. Hanson, *Solving least-squares problems*, Prentice-Hall, Englewood Cliffs, NJ, 1974.
14. I. Lustig, R. Marsten, and D. Shanno, *Computational experience with a primal-dual interior point method for linear programming*, Linear Algebra Appl. **152** (1991), 191–222.
15. S. Mehrotra, *On the implementation of a (primal-dual) interior point method*, Tech. Report 03, Department of Civil Engineering and Management Sciences, Northwestern University, Evanston, IL, 1990.
16. K. Murty, *Note on a Bard-type scheme for solving the complementarity problem*, Oper. Res. **11** (1974), 123–130.
17. P. Pardalos, Y. Ye, and C. Han, *An interior point algorithm for large-scale quadratic problems subject to box constraints*, Lecture Notes in Control and Inform. Sci., vol. 144, Springer, Berlin and New York, 1990, pp. 413–422.
18. L. Portugal, *Solution of least-squares problems*, Master's thesis, Faculdade de Ciências e Tecnologia Universidade de Coimbra, 1992. (Portuguese)

(L. F. Portugal) Departamento de Ciências da Terra, Universidade de Coimbra, 3000 Coimbra, Portugal

(J. J. Júdice and L. N. Vicente) Departamento de Matemática, Universidade de Coimbra, 3000 Coimbra, Portugal

E-mail address, J. J. Júdice: judice@lince.mat.uc.pt