STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
*Stat., Optim. Inf. Comput.*, Vol. **12**, March 2024, pp 405–417.
Published online in International Academic Press (www.IAPress.org)

# Quadratic programming method with an M-matrix

Katia Hassaini*, Mohand Ouamer Bibi

*Research Unit LaMOS, Department of Operations Research, Faculty of the Exact Sciences, University of Bejaia, 06000 Bejaia, Algeria*

**Abstract** In this study, we propose an new approach for solving a quadratic programming problem with an M-matrix and simple constraints. This approach is based on the algorithms of Chandrasekaran, Luk and Pagano. These methods use the fact that an M-matrix possesses a nonnegative inverse which allows to have a sequence of feasible points monotonically increasing. Introducing the concept of support for an objective function developed by Gabasov et al. in 1987, our approach leads to a more general condition which allows to have an initial feasible solution, related to a coordinator support and close to the optimal solution. The programming of our method under MATLAB and that of Luk and Pagano have allowed us to make a comparison between them, in the illustration of two practical examples. The numerical results indicate that our approach is more efficient than the approach proposed by Chandrasekaran, Luk and Pagano.

**Keywords** Convex Quadratic Programming, M-Matrices, Nonnegative Matrices, Projection Newton Method, Support Method.

**AMS 2010 subject classifications** 65K05, 90C2, 90C25, 90C30.

**DOI:** 10.19139/soic-2310-5070-1399

## 1. Introduction

In the literature, several approaches were suggested for solving a quadratic programming problem when the associated matrix $D$ is positive definite or semidefinite [35, 5, 36, 7, 13, 6, 10, 26, 3, 11, 4, 24]. However, it is possible to exploit the special properties of an M-matrix to obtain more efficient special algorithms. The M-matrices are known to have many applications in the modeling of the dynamic systems, in economic sciences and ecology [2, 37]. Such problems include various types of Dirichlet problems with obstacles [28], and models of the application of torsion to a bar [27]. Several of their properties are used in general to establish results of stability for the dynamic systems [31, 32, 29]. Quadratic minimization with an M-matrix arises directly in a variety of applications including portfolio optimization with transaction costs [23], and image segmentation [14]. Convex quadratic programming with an M-matrix is also studied on its own right [1, 12, 13, 19, 20, 33, 34, 22]. The M-matrices are also present in the obstacle problems [15], and active set methods are used to solve them [19], a direct algorithm for the solution to the affine two-sided obstacle problem with an M-matrix is presented in [12], another method for strictly convex quadratic problems is suggested in [13], this method presents an extension of the external points method [18].

The main contribution of this paper lies in the proposal of a new and efficient algorithm for solving quadratic programming problems with an M-matrix and simple constraints. This method takes advantage of the fact that an M-matrix has a nonnegative inverse (all the elements of the matrix $D^{-1}$ are nonnegative), which gives a monotonically increasing sequence of feasible solutions [33, 34]. By introducing the concept of support for an

*Correspondence to: Katia Hassaini (Email: katia.hassaini@univ-bejaia.dz). Research Unit LaMOS (Modelling and Optimization of Systems), Department of Operations Research, Faculty of the Exact Sciences, University of Bejaia, 06000 Bejaia, Algeria.

ISSN 2310-5070 (online) ISSN 2311-004X (print)

objective function [16], our approach differs from the method presented by Chandrasekaran [8], Luk and Pagano [20] by a more general condition that allows us to have an initial feasible solution, close to the optimal solution. This characteristic facilitates faster convergence to the optimal solution, reducing thus the number of iterations required compared with the Chandrasekaran, Luk and Pagano approaches.

The organization of the paper is as follows: in the second section, we present the problem and give some definitions related to our approach. In section 3, the algorithm for solving the quadratic programming problem with an M-matrix and simple constraints is presented. In Section 4, the programming of our method and that of Luk and Pagano under MATLAB have allowed us to make a comparison between them, in the illustration of two practical examples randomly generated, and this, by varying the number of variables. We finish the article by a conclusion.

## 2. Position of the problem and definitions

Let us consider the following problem of quadratic programming with simple constraints:

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} \ F(x) = \frac{1}{2}x^T Dx + c^T x, \\ subject\ to \qquad\qquad x \geq 0, \end{cases} \tag{1}$$

where $c = c(J) = (c_j,\ j \in J)$ and $x = x(J) = (x_j,\ j \in J)$ are real $n$-vectors, with $J = \{1, 2, \cdots, n\}$. The matrix $D = D(J, J)$ is a nonsingular symmetric square M-matrix of order $n$.

***Definition 1. [37]***

A matrix $D = (d_{ij},\ 1 \leq\ i,\ j \leq n)$ is said to be an M-matrix if it satisfies the following properties:

$$d_{ii} > 0 \qquad d_{ij} \leq 0,\ \ i \neq j\ , \qquad D^{-1} \geq 0,$$

where the symbol $D^{-1} \geq 0$ denotes that all the elements of the matrix $D^{-1}$ are nonnegative.

***Remark 1.*** A symmetric M-matrix is always positive definite $(x^T Dx > 0,\ \forall\ x \neq 0)$. Moreover, any submatrix of an M-matrix is itself an M-matrix.

***Definition 2.***

A vector $x \geq 0$ is called a *feasible solution* of the problem (1). A feasible solution $x^0$ is said *optimal* if it gives to the objective function of the problem (1) his minimum value.

Thus, we have

***Theorem 1.*** A feasible solution $x^0$ of the problem (1) is optimal if and only if for all $j \in J$, the following conditions of optimality are satisfied [16]:

$$\begin{cases} x^0_j = 0 \qquad \Rightarrow \qquad g_j(x^0) \geq 0, \\ x^0_j > 0 \qquad \Rightarrow \qquad g_j(x^0) = 0, \quad j \in J, \end{cases} \tag{2}$$

where $g(x) = g(J) = Dx + c$ is the gradient of the objective function $F$ at the point $x$.

Let us consider the quadratic program without constraints

$$\min_{x\in\mathbb{R}^n} \ F(x) = \frac{1}{2}x^T Dx + c^T x, \tag{3}$$

whose the optimal solution $\widehat{x}$ satisfies the equation

$$D\widehat{x} + c = 0 \iff \widehat{x} = -D^{-1}c.$$

Let $J_S$ and $J_N$ be a partition of $J$: $J_S \cup J_N = J,\ \ J_S \cap J_N = \emptyset$. Then the gradient of the function $F$ at point $x$ can be written in the following form:

$$g = \left(\frac{g_S}{g_N}\right), \quad g_S = g(J_S) = D_S x_S + D_{SN} x_N + c_S, \quad g_N = g(J_N) = D_{NS} x_S + D_N x_N + c_N,$$

where

$$x = \left(\frac{x_S}{x_N}\right), \ \ c = \left(\frac{c_S}{c_N}\right), \ \ D_S = D(J_S, J_S),\ \ D_N = D(J_N, J_N),\ \ D_{SN} = D(J_S, J_N).$$

For all subset $J_S$ in $J$, the following condition holds:

$$\det D_S \quad = \quad \det D(J_S, J_S) \neq 0.$$

***Definition 3.***

- The subset $J_S$ is called a *support* of the objective function and the pair $J_p = \{J_S, J_N\}$ is a support of the problem (1).
- The couple $\{x, J_S\}$, comprising the feasible solution $x$ and the support $J_S$ is called a *support feasible solution*.
- A vector $\kappa = \kappa(J) = \big(\kappa(J_S), \kappa(J_N)\big)$ satisfying

$$\begin{cases} \kappa_N = 0, \\ \kappa_S = -D_S^{-1} c_S, \end{cases}$$

  is called a *pseudosolution* of the problem (1). A pseudosolution verifies $g_S(\kappa) = 0$.

- The support $J_P = \{J_S, J_N\}$ is called a *coordinator support* if there is a pseudosolution $\kappa$ such that:

$$g_j(\kappa) \quad \geq \quad 0, \qquad j \in J_N. \tag{4}$$

  In this case, we say that the pseudosolution $\kappa$ is associated to the coordinator support $J_P$.

***Theorem 2.*** A pseudosolution $\kappa$ associated to a coordinator support $J_p$ is optimal in the problem (1) if and only if

$$\kappa_j \geq 0, \qquad j \in J_S. \tag{5}$$

***Remark 2.*** Any pseudosolution $\kappa$, associated to a coordinator support $J_p$, is a feasible solution for the dual of the primal problem (1):

$$\begin{cases} F(\kappa) = -\frac{1}{2}\kappa^T D\kappa \longrightarrow \max, \\ \qquad D\kappa + c \geq 0. \end{cases} \tag{6}$$

***Remark 3.*** As $g(\widehat{x}) = 0$, then the optimal solution $\widehat{x}$ of the problem without constraints (3) is a pseudosolution of the problem (1), associated to the coordinator support $J_p = \{J_S, J_N\}$, where $J_S = J$ and $J_N = \emptyset$. According to the theorem 2, if $\widehat{x} \geq 0$, then $x^0 = \widehat{x}$ is the optimal solution of the problem (1).

Let us recall the following lemma:

***Lemma 1.*** [20].

**(a)** If $c \geq 0$, then $x^0 = 0$ solves the problem (1),

**(b)** If $c \leq 0$, then $x^{\,0} = -D^{-1}c$ solves the problem (1).

To eliminate these two trivial cases, let us consider the general one where the vector $c$ contains both positive and negative components, and construct the two following sets of indices:

$$J_S = \{j \in J : \widehat{x}_j \geq 0\}, \quad J_N = \{j \in J : \widehat{x}_j < 0\}, \quad J_S \cup J_N = J.$$

- If $J_S = J$, then $x^0 = \widehat{x} = -D^{-1}c$ is the optimal solution of the problem (1).
- Else, let $y$ be the projection of $\widehat{x}$ on the admissible set of the problem (1), where $y = (y_j,\ j \in J)$, $y_j = \max\{0, \widehat{x}_j\}$. Therefore we will have

$$\begin{cases} y_N = 0 > \widehat{x}_N, \\ y_S = \widehat{x}_S. \end{cases}$$

For the construction of our algorithm, we have established the following lemmas:

***Lemma 2.*** The following inequality holds:

$$g_S(y) \leq g_S(\widehat{x}).$$

***Proof.*** We have

$$\begin{aligned} g_S(\widehat{x}) &= D_S\,\widehat{x}_S + D(J_S, J_N)\,\widehat{x}_N + c_S \\ &= D_S\,y_S + c_S + D(J_S, J_N)\,\widehat{x}_N \\ &= g_S(y) + D(J_S, J_N)\,\widehat{x}_N \\ &\geq g_S(y), \end{aligned}$$

because $D(J_S, J_N) \leq 0$ and $\widehat{x}_N < 0$. $\square$

Since $g_S(\widehat{x}) = 0$, then by lemma 2 we deduce that $g_S(y) \leq 0$. Then we construct a vector $x$ such that

$$\begin{cases} x_N = y_N = 0, \\ x_S = -D_S^{-1}\,c_S. \end{cases} \tag{7}$$

Thus we have

$$g_S(y) \leq 0 \quad and \quad g_S(x) = 0.$$

***Lemma 3.*** The vectors $y$ and $x$ satisfy the following inequality:

$$x_S \geq y_S \geq 0.$$

***Proof.*** We have

$$D_S\,(x_S - y_S) \;=\; D_S\,x_S + c_S - (D_S\,y_S + c_S).$$

As $x_N = y_N = 0, g_S(y) \leq 0$ and $g_S(x) = 0$, then we deduce

$$D_S\,(x_S - y_S) \;=\; g_S(x) - g_S(y) \geq 0.$$

The submatrix $D_S$ is an M-matrix [20], that yields $D_S^{-1} \geq 0$. Consequently, from the above inequality, we obtain

$$x_S \;\geq\; y_S \;\geq\; 0. \;\square$$

By lemma 3, the constructed vector $x$ (7) is a feasible solution of the problem (1). We have then the following lemma:

***Lemma 4.***

$$F(x) \;\leq\; F(y).$$

***Proof.*** Let

$$2F(x) \;=\; x_S^T\,D_S x_S + 2\,c_S^T\,x_S.$$

As $x_S = -D_S^{-1}c_S$, we will have

$$2F(x) \;=\; c_S^T\,D_S^{-1}c_S - 2\,c_S^T\,D_S^{-1}c_S = -c_S^T\,D_S^{-1}c_S.$$

Because the submatrix $D_S$ is positive definite, then we can write

$$\begin{aligned} 2F(x) \;&\leq\; (y_S - x_S)^T\,D_S(y_S - x_S) - c_S^T\,D_S^{-1}c_S \\ &\leq\; y_S^T\,D_S\,y_S + x_S^T\,D_S\,x_S - 2y_S^T\,D_S\,x_S - c_S^T\,D_S^{-1}\,c_S \\ &\leq\; y_S^T\,D_S\,y_S + c_S^T\,D_S^{-1}\,D_S\,D_S^{-1}\,c_S + 2\,y_S^T\,D_S\,D_S^{-1}\,c_S - c_S^T\,D_S^{-1}\,c_S \\ &\leq\; y_S^T\,D_S\,y_S + 2\,c_S^T\,y_S \\ &\leq\; 2\,F(y). \end{aligned}$$

Hence

$$F(x) \;\leq\; F(y). \quad \square$$

***Remark 4.*** The constructed vector $x$ (7) satisfies thus the following inequality:

$$F(\widehat{x}) < F(x^0) \leq F(x) \leq F(y), \tag{8}$$

where $y$ is the projection of $\widehat{x}$ on the admissible set of the problem (1).

We recall the following theorem

***Theorem 3.*** [20] Assume $D_S^{-1} c_S \leq 0$ for some nonempty subset $J_S \subset J$. Define a vector $x$ with $x_S = -D_S^{-1} c_S$ and $x_N = 0$. Let $J_N^-$ and $J_N^+$ be two sets partitioning $J_N$ such that

$$J_N^- = \{j \in J_N : \ g_j(x) < 0\}, \ J_N^+ = \{j \in J_N : \ g_j(x) \geq 0\}.$$

If the set $J_N^-$ is empty, then

**1.** the vector $x$ solves the problem (1), else

**2.** let $\overline{J_S} := J_S \cup J_N^-$. Construct $\overline{x}$ with $\overline{x}(\overline{J_S}) = -D^{-1}(\overline{J_S}, \overline{J_S})\, c(\overline{J_S})$ and $\overline{x}(J_N^+) = 0$. We get

**(a)** $\overline{x}(J_S) \geq x(J_S) \geq 0, \ \ \overline{x}(J_N^-) \geq 0, \ \ \overline{x}(J_N^+) = 0,$

**(b)** $g_j(\overline{x}) \leq g_j(x), \quad j \in J_N^+,$

**(c)** $F(\overline{x}) < F(x).$

## 3. Algorithm of the method

Based on the previous theorem, we propose the following algorithm:

**Begin**

1. Compute the optimal solution $\widehat{x}$ of the problem (3):

$$g(\widehat{x}) = D\widehat{x} + c = 0 \Longrightarrow \widehat{x} = -D^{-1}c.$$

2. If $\widehat{x} \geq 0$, then stop and the vector $x^0 = \widehat{x}$ is the optimal solution of the problem (1).

3. Else, define the sets:

$$J_S = \{j \in J : \ \widehat{x}_j \geq 0\}, \quad J_N = \{j \in J : \ \widehat{x}_j < 0\}.$$

4. Construct $x$ as follows:

$$x_N = 0, \quad x_S = -D_S^{-1} c_S.$$

5. Let $J_N^-$ and $J_N^+$ be two sets partitioning $J_N$ such that

$$J_N^- = \{j \in J_N : \ g_j(x) < 0\}, \quad J_N^+ = \{j \in J_N : \ g_j(x) \geq 0\}.$$

6. **Repeat**

   **1.** Compute $g_N(x) = \ D(J_N, J_S)\, x_S \ + \ c_N,$

   **2.** Let $J_N^- = \{j \in J_N : \ g_j(x) < 0\},$

   **3.** If $J_N^-$ is nonempty, then

      **(a)** Let $J_S := J_S \cup J_N^-$ and $J_N := J_N \setminus J_N^-,$

      **(b)** Reconstruct $x$ such that $x_N \ = \ 0 \ \mathit{and} \ x_S = -D_S^{-1} c_S,$

   **until** the set $J_N^- = \emptyset$.

**End.**

## 4. Experimental results

In this section, we have chosen two representative problems. The goal is to show the effectiveness of our proposed algorithm in making a numerical comparison with the algorithm of Luk and Pagano [20]. All experiments were conducted on a computer with Intel(R) Core(TM) i3-2350 CPU @ 2.30 GHz with 4.00Go of RAM, working under Windows 7 operating system with MATLAB R2015a programming language. The criterion of the comparison between the two methods is the average CPU time (Avr-CPU) in seconds and the average number of iterations (Avr-Iter) necessary provided to obtain the optimal solution of the problem. All these tests were conducted on the same computer. The values presented in the tables represent the averages of $5$ test problems for each value $n$. We define by

- **Algorithm1**: Chandrasekaran, Luk and Pagano method[20],
- **Algorithm2**: proposed algorithm,
- $NJ_S$: the number of elements in the support $J_S$ of the objective function, just after having calculated $\widehat{x} = -D^{-1}c$,
- $\overline{NJ_S}$: the number of elements in the support $J_S$ of the objective function , at the optimum,
- $NP$: the number of elements in the set $P$, at the initialization of $x$, with $P = \{j \in J \in : c_j \leq 0\}$ and $\overline{P} = J \setminus P$,
- $\overline{NP}$: the number of elements in the set $P$, at the optimum,
- **Avr-Iter**: the average number of iterations performed by each algorithm,
- **Avr-CPU**: the average CPU time in seconds necessary provided to obtain the optimal solution of the problem.

### *4.1. Example 1.*

Let us consider the quadratic program (1):

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} \; F(x) = \frac{1}{2}x^T D x + c^T x, \\ subject\ to \qquad\qquad x \geq 0, \end{cases} \tag{9}$$

and, the matrix $D$ is the matrix corresponding to the finite difference discretization of the one-dimensional Dirichlet problem [38].

$$D = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix}_{n\times n} . \tag{10}$$

We choose to generate the vector $c$ in order to have three cases of the set $J_S$. Let $r_i$ be a random number from an uniform distribution $r_i \in U[0,1]$. The comparison criterion between the two methods is based on the average CPU time (Avr-CPU) in seconds and the average number of iterations (Avr-Iter) necessary provided to obtain the optimal solution of the problem. The results are presented in the tables below:

*4.1.1.* ***Case 1.*** The vector $c$ is generated so that to have $NJ_S = n$

$$c_i = 11 - 20\, r_i \qquad for \quad i = 1, 2, \ldots, n. \tag{11}$$

| Dimension | Algorithm1 | | | | Algorithm2 | | | |
|---|---|---|---|---|---|---|---|---|
| n | Avr-Iter | NP | $\overline{NP}$ | Avr-CPU | Avr-Iter | $NJ_S$ | $\overline{NJ_S}$ | Avr-CPU |
| 500 | 07 | 280 | 500 | 0.0798 | 0 | 500 | 500 | 0.0048 |
| 1000 | 06 | 559 | 1000 | 0.1172 | 0 | 1000 | 1000 | 0.0067 |
| 1500 | 07 | 846 | 1500 | 0.1642 | 0 | 1500 | 1500 | 0.0091 |
| 2000 | 08 | 1077 | 2000 | 0.1867 | 0 | 2000 | 2000 | 0.0098 |
| 2500 | 08 | 1356 | 2500 | 0.1910 | 0 | 2500 | 2500 | 0.0111 |
| 3000 | 08 | 1625 | 3000 | 0.1999 | 0 | 3000 | 3000 | 0.0130 |
| 4000 | 09 | 2187 | 4000 | 0.2022 | 0 | 4000 | 4000 | 0.0193 |
| 5000 | 11 | 2742 | 5000 | 0.2153 | 0 | 5000 | 5000 | 0.0269 |

Table 1. The average CPU time in seconds and the average number of iterations performed by each algorithm with $NJ_S = n$.

*4.1.2.* ***Case 2.*** For $NJ_S < n$, the vector $c$ is generated by

$$c_i = 11 - 22\, r_i \qquad for \quad i = 1, 2, \ldots, n. \tag{12}$$

| Dimension | Algorithm1 | | | | Algorithm2 | | | |
|---|---|---|---|---|---|---|---|---|
| n | Avr-Iter | NP | $\overline{NP}$ | Avr-CPU | Avr-Iter | $NJ_S$ | $\overline{NJ_S}$ | Avr-CPU |
| 500 | 14 | 249 | 498 | 0.1099 | 06 | 420 | 498 | 0.0146 |
| 1000 | 21 | 499 | 999 | 0.1182 | 07 | 982 | 999 | 0.0165 |
| 1500 | 23 | 727 | 1487 | 0.1226 | 21 | 142 | 1487 | 0.0378 |
| 2000 | 26 | 1010 | 1960 | 0.1396 | 30 | 851 | 1960 | 0.0569 |
| 2500 | 22 | 1242 | 2454 | 0.1406 | 32 | 330 | 2454 | 0.0598 |
| 3000 | 26 | 1482 | 2923 | 0.1473 | 35 | 1074 | 2923 | 0.0621 |
| 4000 | 26 | 2001 | 3958 | 0.1914 | 40 | 1172 | 3958 | 0.1101 |
| 5000 | 26 | 2500 | 4969 | 0.1974 | 40 | 2237 | 4969 | 0.1230 |

Table 2. The average CPU time in seconds and the average number of iterations performed by each algorithm with $NJ_S < n$.

*4.1.3.* ***Case 3.*** For $NJ_S = 0$, one generates the vector $c$ by

$$c_i = 11 - 25\, r_i \qquad for \quad i = 1, 2, \ldots, n. \tag{13}$$

| Dimension | Algorithm1 | | | | Algorithm2 | | | |
|---|---|---|---|---|---|---|---|---|
| n | Avr-Iter | NP | $\overline{NP}$ | Avr-CPU | Avr-Iter | $NJ_S$ | $\overline{NJ_S}$ | Avr-CPU |
| 500 | 07 | 199 | 335 | 0.1053 | 11 | 0 | 335 | 0.0167 |
| 1000 | 09 | 444 | 752 | 0.1099 | 12 | 0 | 752 | 0.0232 |
| 1500 | 09 | 648 | 1051 | 0.1106 | 13 | 0 | 1051 | 0.0287 |
| 2000 | 10 | 853 | 1444 | 0.1218 | 16 | 0 | 1444 | 0.0431 |
| 2500 | 11 | 1149 | 1894 | 0.1240 | 16 | 0 | 1894 | 0.0487 |
| 3000 | 08 | 1360 | 2272 | 0.1249 | 11 | 0 | 2272 | 0.0529 |
| 4000 | 09 | 1758 | 2862 | 0.1341 | 14 | 0 | 2862 | 0.0681 |
| 5000 | 08 | 2206 | 3660 | 0.1456 | 17 | 0 | 3660 | 0.0936 |

Table 3. The average CPU time in seconds and the average number of iterations performed by each algorithm with $NJ_S = 0$.

A comparison of the average CPU time for the proposed algorithm and the Luk and Pagano algorithm [20], is showed in Figure 1.

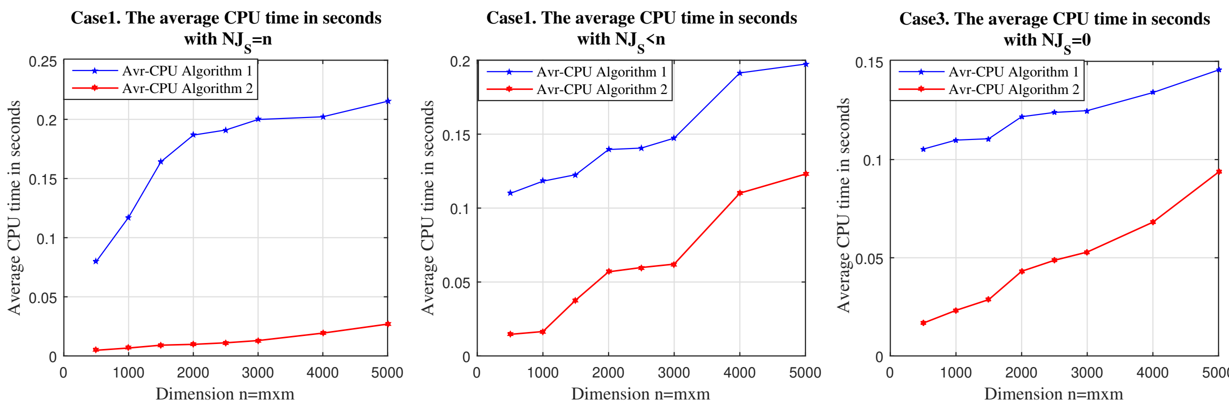


Figure 1. The average CPU time (Avr-CPU) in seconds performed by each algorithm for example 1

In this example, we see that our approach is more efficient in machine time than the approach of Chandrasekaran, Luk and Pagano. And that, whatever the number of elements in the support $J_S$ of the objective function at the initial step of the algorithm.

### 4.2. Example 2.

In this second example, the matrix $D$ of the problem is chosen as a 5−point finite difference Laplacian operator[38]:

$$D = \begin{pmatrix} B & -I & 0 & \cdots & 0 \\ -I & B & -I & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -I & B & -I \\ 0 & \cdots & 0 & -I & B \end{pmatrix}_{m^2 \times m^2}, \quad B = \begin{pmatrix} 4 & -1 & 0 & \cdots & 0 \\ -1 & 4 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 4 & -1 \\ 0 & \cdots & 0 & -1 & 4 \end{pmatrix}_{m \times m}, \tag{14}$$

with $n = m^2$. The matrix $I$ is the identity matrix.

We choose to generate the vector $c$ in order to have three cases of the set $J_S$ as in the previous example. The comparison criterion between the two methods is based on the average CPU time (Avr-CPU) in seconds and the average number of iterations (Avr-Iter) necessary provided to obtain the optimal solution of the problem. The results obtained are presented in the tables below:

#### *4.2.1.* ***Case 1.*** The vector $c$ is generated by

$$c_i = 8 - 10\, r_i \qquad for \quad i = 1, 2, \ldots, n. \tag{15}$$

| Dimension | Algorithm1 | | | | Algorithm2 | | | |
|---|---|---|---|---|---|---|---|---|
| $n = m \times m$ | Avr-Iter | NP | $\overline{NP}$ | Avr-CPU | Avr-Iter | $NJ_S$ | $\overline{NJ_S}$ | Avr-CPU |
| $20 \times 20$ | 01 | 393 | 400 | 0.0768 | 0 | 400 | 400 | 0.0053 |
| $30 \times 30$ | 01 | 813 | 900 | 0.0791 | 0 | 900 | 900 | 0.0070 |
| $40 \times 40$ | 01 | 1222 | 1600 | 0.0801 | 0 | 1600 | 1600 | 0.0081 |
| $50 \times 50$ | 01 | 1603 | 2500 | 0.0891 | 0 | 2500 | 2500 | 0.0100 |
| $60 \times 60$ | 01 | 2387 | 3600 | 0.0960 | 0 | 3600 | 3600 | 0.0146 |
| $70 \times 70$ | 01 | 3207 | 4900 | 0.1320 | 0 | 4900 | 4900 | 0.0225 |

Table 4. The average CPU time in seconds and the average number of iterations performed by each algorithm with $NJ_S = n$.

#### *4.2.2.* ***Case 2.*** The vector $c$ is generated by

$$c_i = 8 - 16\, r_i \qquad for \quad i = 1, 2, \ldots, n. \tag{16}$$

| Dimension | Algorithm1 | | | | Algorithm2 | | | |
|---|---|---|---|---|---|---|---|---|
| $n \times n$ | Avr-Iter | NP | $\overline{NP}$ | Avr-CPU | Avr-Iter | $NJ_S$ | $\overline{NJ_S}$ | Avr-CPU |
| $20 \times 20$ | 12 | 198 | 380 | 0.1093 | 15 | 46 | 380 | 0.0179 |
| $30 \times 30$ | 12 | 441 | 882 | 0.1143 | 15 | 453 | 882 | 0.0233 |
| $40 \times 40$ | 15 | 849 | 1590 | 0.1199 | 17 | 895 | 1590 | 0.0283 |
| $50 \times 50$ | 16 | 1253 | 2498 | 0.1323 | 18 | 2464 | 2498 | 0.0353 |
| $60 \times 60$ | 19 | 1792 | 3522 | 0.1492 | 24 | 590 | 3522 | 0.0653 |
| $70 \times 70$ | 24 | 2360 | 4860 | 0.1734 | 22 | 2948 | 4890 | 0.0679 |

Table 5. The average CPU time in seconds and the average number of iterations performed by each algorithm with $NJ_S < n$.

#### *4.2.3.* ***Case 3.*** The vector $c$ is generated by

$$c_i = 8 - 20\, r_i \qquad for \quad i = 1, 2, \ldots, n. \tag{17}$$

| Dimension | Algorithm1 | | | | Algorithm2 | | | |
|---|---|---|---|---|---|---|---|---|
| $n \times n$ | Avr-Iter | NP | $\overline{NP}$ | Avr-CPU | Avr-Iter | $NJ_S$ | $\overline{NJ_S}$ | Avr-CPU |
| $20 \times 20$ | 04 | 154 | 219 | 0.1037 | 06 | 0 | 219 | 0.0141 |
| $30 \times 30$ | 05 | 339 | 506 | 0.1079 | 07 | 0 | 506 | 0.0204 |
| $40 \times 40$ | 05 | 64 7 | 937 | 0.1108 | 07 | 0 | 937 | 0.0259 |
| $50 \times 50$ | 04 | 899 | 1221 | 0.1139 | 05 | 0 | 1221 | 0.0306 |
| $60 \times 60$ | 04 | 1277 | 1740 | 0.1199 | 05 | 0 | 1740 | 0.0458 |
| $70 \times 70$ | 06 | 1891 | 2592 | 0.1401 | 07 | 0 | 2592 | 0.0628 |

Table 6. The average CPU time in seconds and the average number of iterations performed by each algorithm $NJ_S = 0$.

A comparison of the average CPU time for the proposed algorithm, the Luk and Pagano algorithm [20], is showed in Figure 2. And that, whatever the number of elements in the support $J_S$ of the objective function at the initial step of the algorithm.

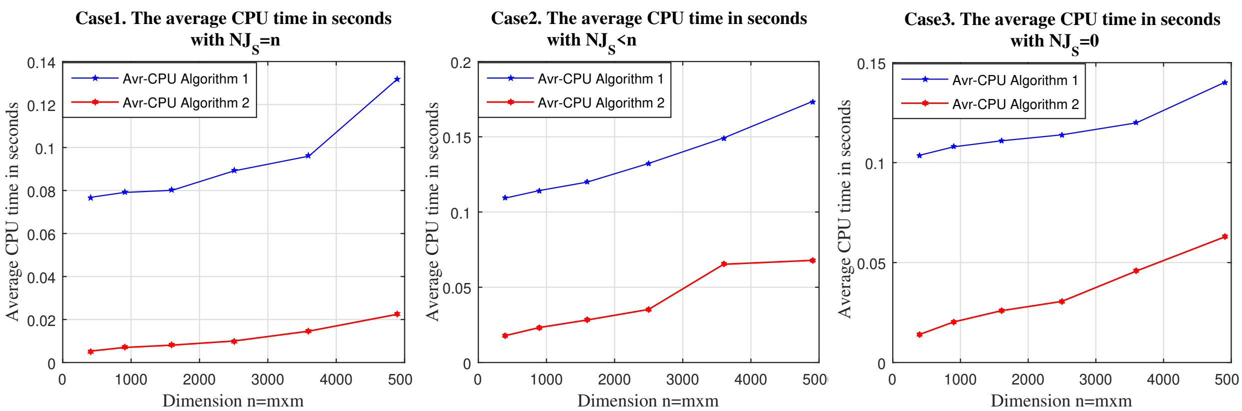


Figure 2. The average CPU time (Avr-CPU) in seconds performed by each algorithm for the example 2.

From the numerical examples above, our approach often requires less average CPU time than Chandrasekaran, Luk and Pagano's approach. This is true whatever the number of elements in the support $J_S$ of the objective function at the initial step of the algorithm.

## 5. Conclusion

The lemmas 2, 3 and 4 allowed us to start the algorithm with a feasible solution $x$ checking the conditions of the theorem 3 and the inequality (8). If the matrix of our objective function $D = I$, where $I$ is the identity matrix , then we have

$$J_S = \{j \in J : c_j \leq 0\}, \quad J_N = \{j \in J : c_j > 0\},$$

and we find the conditions of the initialization of the algorithms of Chandrasekaran, Luk and Pagano [8, 20]. Let us notice that their algorithms finish with $J_N$ or $J_N^-$ empty, while ours always finishes with $J_N^- = \emptyset$ and $J_N \neq \emptyset$, and this, because of our initialization. Indeed, the case $J_N = \emptyset$ corresponds to the optimal solution $x^0 = \widehat{x}$.

From the two numerical examples, we see that our approach is more efficient in machine time than the approach of Chandrasekaran, Luk and Pagano. This is true for both examples, whatever the number of elements in the support $J_S$ of the objective function at the initial step of the algorithm.

## REFERENCES


1. A. Atamtürk, and A. Gómez, *Strong formulations for quadratic optimization with M-matrices and indicator variables*, Math. Program, vol. 170, no. 1, p. 141-176., 2018.
2. A. Berman, and R. J. Plemmons, *Nonnegative Matrices in Mathematical Sciences*, Academic Press, New York, 1979.
3. D. P. Bertsekas, *Projected Newton Method for Optimization Problems with Simple Constraints*, SIAM Journal on Control and Optimization, vol. 20, no 2, p. 221-246, 1982.
4. B. Brahmi and M. O. Bibi, *Dual Support method for solving convex quadratic programs*, Optimization, Vol. 59, n. 6, p. 851–872, 2010.
5. A. Bounceur, S. Djemai, B. Brahmi, M. O. Bibi, and R. Euler, *A Classification Approach for an Accurate Analog/RF BIST Evaluation Based on the Process Parameters*, Journal of Electronic Testing, Vol. 34, 321–335, 2018.
6. M. O. Bibi, N. Ikheneche, and M. Bentobache, *A hybrid direction algorithm for solving a convex quadratic problem*, International Journal Mathematics in Operations Research, Vol. 16, No. 2, pp. 159-178, 2020.
7. R. Cambini, and C. Sodini, *A sequential method for a class of box constrained quadratic programming problems*, Mathematical Methods of Operations Research, vol. 67, no 2 , p. 223-243, 2008.
8. R. Chandrasekaran, *A special case of the complementary pivot problem*, Opsearch, vol. 7, p. 263-268, 1970.
9. G. Cimatti, *On a problem of the theory of lubrication governed by a variational inequality*, Applied Mathematics and Optimization, vol. 3, no. 2-3, pp. 227-242, 1976.
10. A. N. Daryina, and A. F. Izmailov, *On Active-Set Methods for the Quadratic Programming Problem*, Computational Mathematics and Mathematical Physics, vol. 52, p. 512-523, 2012.
11. A. Friedlander, J. Martinez, and M. Raydan, *A new method for large-scale box constrained convex quadratic minimization problems*, Optimization Methods and Software, vol. 5, no. 1, p. 57–74, 1995.
12. Y. J. Jiang, and J. P. Zeng, *Direct algorithm for the solution of two-sided obstacle problems with M-matrix*, Numerical Linear Algebra with Applications, vol. 18, no. 1, p. 167-173. 2011.
13. P. Hungerländer, and F. Rendl, *A feasible active set method for strictly convex quadratic problems with simple bounds*, SIAM Journal on Optimization, **25**(3), 1633–1659, 2017.
14. D.S. Hochbaum, *Multi-label markov random fields as an efficient and effective tool for image segmentation, total variations and regularization*, Numer. Math. Theory Methods Appl, Vol. 6, no. 1, 169–198, 2013.
15. Y.C. Hsieh, D. L. Bricker, *Solving obstacle problems by using a new infeasible interior point algorithm*, Journal of the Chinese Institute of Industrial Engineers, vol. 16, no. 6, 771–780, 1999.
16. R. Gabasov, F. M. Kirillova, O. I. Kostyukova, and V.M. Raketsky, *Constructive methods of optimization*, volume 4: Convex Problems, University Press, Minsk (1987).
17. R. Glowinski, *Lectures on numerical methods for non-linear variational problems*, Springer Science & Business Media, Berlin, New York, 2008.
18. K. Kunish, and F. Rendl, *An infeasible active set method for convex problems with simple bounds*, SIAM Journal on Optimization, vol. 14, no 1, p. 35-52, 2003.
19. T. Kärkkäinen, K. Kunisch, and P. Tarvainen, *Augmented Lagrangian Active Set Methods for Obstacle Problems*, Journal of optimization theory and applications, vol. 119, no.3, p. 499–533, 2003.
20. F. T. Luk and M. Pagano, *Quadratic programming with M-Matrices*, Linear Algebra and Its Applications, vol. 33, p. 15-40, 1980.
21. Y. Lin, and C. W. Ciyer, *An alternating direction implicit algorithm for the solution of linear complementarity problems arising from free boundary problems*, Applied Mathematics and Optimization, vol. 13, no. 1, p. 1-17, 1985.
22. L. Li and Y. Kobayashi, *A block recursive algorithm for the linear complementarity problem with an M-matrix*, International Journal of Innovative, Computing, Information and Control, Vol. 2, N. 6, 1327–1335, 2006.
23. M.S, Lobo, F. M. azel, and S. Boyd, *Portfolio optimization with linear and fixed transaction costs*, Annals of Operations Research, vol. 152, 341–365, 2007.
24. M. Pakdaman, and S. Effati, *Bounds for convex quadratic programming problems and some important applications*, International Journal of Operational Research, vol. 30, no. 2, 277-287, 2017.
25. E. Polak, *Computational Methods in Optimization: A Unified Approach* , Mathematics in Science and Engineering, London, Academic Press, 1971.
26. S. Radjef, and M. O. Bibi, *An Effective Generalization of the Direct Support Method in Quadratic Convex Programming*, Applied Mathematical Sciences, vol. 6, no 31, p. 1525-1540, 2012.
27. J. F. Rodrigues, *Obstacle Problems in Mathematical Physics*, North-Holland, New York, 1987.
28. F. Scarpini, *Some algorithms solving the unilateral Dirichlet problems with two constraints*, Calcolo, vol. 12, no. 2, pp. 113-149, 1975.
29. Q. Song, F. Liu, J. Cao, and W. Yu. *M-matrix strategies for pinning-controlled leader-following consensus in multiagent systems with nonlinear dynamics*, IEEE Trans Cybern, vol. 43, no. 6, p. 1688-1697, 2013.
30. D. D. Šiljak, *Large-scale Dynamic Systems: Stability and Structure*, North-Holland, New York, 1978.
31. D. M. Stipanovic, and D. D. Šiljak, *Stability of polytopic systems via convex M-matrices and parameter-dependent Liapunov functions*, Nonlinear Analysis, Theory, Methods and Applications, vol. 40, no. 1, p. 589-609, 2000.

32. D. M. Stipanovic, S. Shankaran, and C.J. Tomlin, *Multi-agent avoidance control using an M-matrix property*, The Electronic Journal of Linear Algebra, vol. 12, p. 64-72, 2005.
33. A. Stachurski, *An Equivalence between two algorithms for a class of quadratic programming problems with M-matrices*, Optimization, vol. 21, no 6, p. 871-878, 1990.
34. A. Stachurski, *Monotone sequences of feasible solutions for quadratic programming problems with M-matrices and box constraints*, In: System Modelling and Optimization: Proceedings of 12th IFIP Conference, Budapest, Hungary, September 2–6, 1985. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 896-902, 2006.
35. A. Stachurski, *On a conjugate directions method for solving strictly convex QP problem*, Mathematical Methods of Operations Research, vol. 86, no. 3, p. 523–548, 2017.
36. B. Schofield, *On Active Set Algorithms for Solving Bound-Constrained Least Squares Control Allocation Problems*, In: Proceedings of the American Control Conference, 2597–2602. Seattle, Washington, USA, 2008, June.
37. R. S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, N.J, 1962.
38. G. Windisch, *M-matrices in Numerical Analysis*, Springer-Verlag, German, 2013.