Bulletin of Mathematical Sciences and Applications
ISSN: 2278-9634, Vol. 15, pp 17-35
doi:10.18052/www.scipress.com/BMSA.15.17

Submitted: 2016-02-23
Revised: 2016-03-22
Accepted: 2016-03-31
Online: 2016-05-18

The Linear Complementarity Problem and a Modified Newton's Method to Find its Solution

Youssef EL FOUTAYENI[(1,3),a*] , and Mohamed KHALADI[(2,3),b]

[1] Analysis, Modeling and Simulation Laboratory, Hassan II University, Morocco

[2] Mathematical Populations Dynamics Laboratory, Cadi Ayyad University, Morocco

[3] Unit for Mathematical and Computer Modeling of Complex Systems, IRD, France

[a] foutayeni@gmail.com, [b] khaladi@uca.ma

Keywords: Linear complementarity problem, Sixth order method, Interior point methods, System of non-linear equations.

Abstract. In this paper, we present a new interior-point method of convergence order six to solve the linear complementarity problem. Computational efficiency in its general form is discussed and a comparison between the efficiency of the proposed method and existing ones is made. The performance is tested through numerical experiments on some test problems and a practical example of bio-economic equilibrium model.

Introduction

The linear complementarity problem, denoted by $LCP(q, M)$, is to find a column vector $z \in \mathrm{IR}^n$ such that $0 \leqslant z \perp (Mz + q) \geqslant 0$, or showing that no such vector exists, where $M \in \mathrm{IR}^{n\times n}$ and $q \in \mathrm{IR}^n$ are given data. More concretely, it is to find a column vector $z \in \mathrm{IR}^n$ satisfying

$$\begin{cases} z \geqslant 0 \\ Mz + q \geqslant 0 \\ z^T(Mz + q) = 0 \end{cases} \tag{1}$$

or to show that no such vector exists.

The linear complementarity problem, introduced by Cottle [1], is one of the most widely studied mathematical programming problems. Solving $LCP(q, M)$ for an arbitrary matrix M is NP-complete [2], while there are several classes of matrices M for which the associated $LCPs$ can be solved efficiently. For details of the theory of $LCPs$, see the books of Cottle et al. [3], Murty [4] and El foutayeni et al. [5-8].

Using the shorter notation, the linear complementarity problem defined above can be expressed as the $LCP(q, M)$. From the constraints $z \geqslant 0$, $w = Mz + q \geqslant 0$, and $z_i w_i = 0$; follows that z and w are required to be nonnegative, and that at least one of the component-pair (z_i, w_i) must be zero, that is, if $z_i > 0$, then $w_i = 0$ and if $w_i > 0$, then $z_i = 0$. The notation $x \perp y$ means that $x^T y = 0$. Inequalities involving vectors are understood to hold component-wise.

In this paper, we assume that M is a $P-matrix$; recall that a matrix $M \in \mathrm{IR}^{n\times n}$ is a $P-matrix$ if the determinants of all principal submatrices are positive. The feasible set and the strict feasible set of the $LCP(q, M)$, are denoted respectively by:

$S = \{z \in \mathrm{IR}^n : z \geqslant 0,\ Mz + q \geqslant 0\}$ and $\mathring{S} = \{z \in \mathrm{IR}^n : z > 0,\ Mz + q > 0\}$.

Let us assume that the strict feasible set of $LCP(q, M)$ is nonempty. We wish to find a column vector z^* in S such that $z^* \perp (Mz^* + q)$; such a vector is the unique solution of $LCP(q, M)$. Let $F(z) = zw = (z_1w_1, z_2w_2, ..., z_nw_n)^T$. Our aim is to construct an interior-point method for solving $LCP(q, M)$, more precisely, the goal is to build a sequence $z^{(k)} \in S$ such that $\left\|F(z^{(k)})\right\| \leqslant \epsilon$, where ϵ is given; this means that $z^{(k)}$ converges to z^*.

In the second section, we give the main result of this paper: we present a three-step iterative method of convergence order six for solving the linear complementarity problem. In the third section, we

discuss the efficiency and behavior of uniform convergence of the method and to verify the theoretical results through numerical experiments on some test problems and a practical example of bio-economic equilibrium model. The last section contains the concluding remarks.

The main result

The continuously rising success of interior point techniques applied to Linear Programming LP has stimulated research in various related fields. One possible line of generalization consists in looking at linear complementarity problem. This type of generalization is studied in the present paper.

As is done in this area, we use Z to denote the diagonal matrix with diagonal z and employ analogous notation for other quantities. Also we use I to denote the unit matrix whose dimension will vary with the context. A point $z \in \mathrm{IR}^n$ is said to be feasible for problem $LCP(q, M)$ if $z \geqslant 0$ and $Mz + q \geqslant 0$.

Lemma 1. *Let $z^{(k)} \in \mathring{S}$, then $F'\left(z^{(k)}\right)$ is nonsingular.*

Proof. Suppose that $z^{(k)} \in \mathring{S}$, then $z^{(k)} > 0$ and $w^{(k)} = Mz^{(k)} + q > 0$.

From the definition of function F, we have $F(z^{(k)}) = z^{(k)}w^{(k)}$.

Hence, it follows that $F'\left(z^{(k)}\right) = Z^{(k)}M + W^{(k)}$, where

$$\begin{cases} Z^{(k)} = Diag(z_1^{(k)}, z_2^{(k)}, ..., z_n^{(k)}), \\ W^{(k)} = Diag(w_1^{(k)}, w_2^{(k)}, ..., w_n^{(k)}). \end{cases}$$

Using the fact that $z^{(k)} \in \mathring{S}$, then

$$\begin{cases} z^{(k)} > 0, \\ w^{(k)} = Mz^{(k)} + q > 0. \end{cases}$$

Therefore, $Z^{(k)}$ and $W^{(k)}$ are positive semidefinite matrices; and since M is a $P - matrix$, it follows that $F'\left(z^{(k)}\right)$ is nonsingular. This completes the proof of the Lemma.

The error relationship $r^{(k)} = z^{(k)} - z^*$ is the basis for us to establish convergence theorems, it holds that

Lemma 2. *Let $r^{(k)} = z^{(k)} - z^*$.*

Then

$$\left[F'\left(z^{(k)}\right)\right]^{-1} F\left(z^{(k)}\right) = r^{(k)} - M_2\left(r^{(k)}\right)^2 + 2(M_2^2 - M_3)\left(r^{(k)}\right)^3 + o\left(\left(r^{(k)}\right)^4\right).$$

Proof. Let us use the result of Taylor's expansion on vector functions for F about z:

$$F(z+h) = F(z) + F'(z)\,h + \frac{1}{2!}F''(z)\,h^2 + ... + \frac{1}{p!}F^{(p)}(z)\,h^{(p)} + R_p, \tag{2}$$

where $F^{(i)}$ denotes the i^{th} Fréchet derivative of function F, and

$$\|R_p\| \leqslant \frac{1}{p!}\sup_{0\leqslant t\leqslant 1}\left\|F^{(p)}(x+th)\right\|\|h\|^p.$$

Let

$$r^{(k)} = z^{(k)} - z^*. \tag{3}$$

Then setting $z = z^*$ and using the fact that $F(z^*) = 0$ in (2), we obtain

$$F\left(z^{(k)}\right) = F'\left(z^*\right)\left[r^{(k)} + M_2\left(r^{(k)}\right)^2 + M_3\left(r^{(k)}\right)^3 + o\left(\left(r^{(k)}\right)^4\right)\right], \tag{4}$$

where $M_i = \frac{1}{i!}\left[F'\left(z^*\right)\right]^{-1} F^{(i)}\left(z^*\right) \in L_i(\mathrm{IR}^n, \mathrm{IR}^n)$ and $\left[F'\left(z^*\right)\right]^{-1}$ denotes the inverse of $F'\left(z^*\right)$.

This implies that

$$F'\left(z^{(k)}\right) = F'\left(z^*\right)\left[I + 2M_2 r^{(k)} + 3M_3\left(r^{(k)}\right)^2 + o\left(\left(r^{(k)}\right)^3\right)\right], \tag{5}$$

where I denotes the unit matrix.

Then

$$\left[F'\left(z^{(k)}\right)\right]^{-1} = B_0^{-1}\left[F'\left(z^*\right)\right]^{-1}, \tag{6}$$

where

$$B_0 = I + 2M_2 r^{(k)} + 3M_3\left(r^{(k)}\right)^2 + o\left(\left(r^{(k)}\right)^3\right).$$

The inverse of B_0 is given by

$$B_0^{-1} = I - 2M_2 r^{(k)} + \left(4M_2^2 - 3M_3\right)\left(r^{(k)}\right)^2 + o\left(\left(r^{(k)}\right)^3\right).$$

From (6), we have

$$\left[F'\left(z^{(k)}\right)\right]^{-1} = \left[B_1 + B_2 + o\left(\left(r^{(k)}\right)^3\right)\right]\left[F'\left(z^*\right)\right]^{-1}, \tag{7}$$

where

$$\begin{cases} B_1 = I - 2M_2 r^{(k)}, \\ B_2 = \left(4M_2^2 - 3M_3\right)\left(r^{(k)}\right)^2. \end{cases}$$

Now, from (4) and (7), it follows that

$$\left[F'\left(z^{(k)}\right)\right]^{-1} F\left(z^{(k)}\right) = r^{(k)} - M_2\left(r^{(k)}\right)^2 + 2\left(M_2^2 - M_3\right)\left(r^{(k)}\right)^3 + o\left(\left(r^{(k)}\right)^4\right).$$

This completes the proof of the Lemma.

Lemma 3. *Let*

$$\begin{cases} x^{(k)} = z^{(k)} - \frac{1}{2}\left[F'\left(z^{(k)}\right)\right]^{-1} F\left(z^{(k)}\right), \\ r_x^{(k)} = x^{(k)} - z^*. \end{cases}$$

Then

$$\left[F'\left(x^{(k)}\right)\right]^{-1} F'\left(z^*\right) = I - 2M_2 r_x^{(k)} + \left(4M_2^2 - 3M_3\right)\left(r_x^{(k)}\right)^2 + o\left(\left(r_x^{(k)}\right)^3\right).$$

Proof. Let $x^{(k)} = z^{(k)} - \frac{1}{2}\left[F'\left(z^{(k)}\right)\right]^{-1} F\left(z^{(k)}\right)$ and let $r_x^{(k)} = x^{(k)} - z^*$. Using the previous Lemma, we have

$$r_x^{(k)} = \frac{1}{2} r^{(k)} + \frac{1}{2} M_2\left(r^{(k)}\right)^2 - \left(M_2^2 - M_3\right)\left(r^{(k)}\right)^3 + o\left(\left(r^{(k)}\right)^4\right). \tag{8}$$

Expanding $F'\left(x^{(k)}\right)$ about z^*, we obtain

$$F'\left(x^{(k)}\right) = F'\left(z^*\right)\left[I + 2M_2 r_x^{(k)} + 3M_3\left(r_x^{(k)}\right)^2 + o\left(\left(r_x^{(k)}\right)^3\right)\right]. \tag{9}$$

In the same way, we have

$$\left[F'\left(x^{(k)}\right)\right]^{-1} F'\left(z^*\right) = B_3 + B_4 + o\left(\left(r_x^{(k)}\right)^3\right),$$

where

$$\begin{cases} B_3 = I - 2M_2 r_x^{(k)}, \\ B_4 = \left(4M_2^2 - 3M_3\right)\left(r_x^{(k)}\right)^2. \end{cases}$$

This completes the proof of the Lemma.

Lemma 4. *Let*

$$\begin{cases} y^{(k)} = z^{(k)} - \left[F'\left(x^{(k)}\right)\right]^{-1} F\left(z^{(k)}\right), \\ r_y^{(k)} = y^{(k)} - z^*. \end{cases}$$

Then

$$F\left(y^{(k)}\right) = F'\left(z^*\right)\left[r_y^{(k)} + o\left(\left(r_y^{(k)}\right)^2\right)\right].$$

Proof. Let $y^{(k)} = z^{(k)} - \left[F'\left(x^{(k)}\right)\right]^{-1} F\left(z^{(k)}\right)$ and let $r_y^{(k)} = y^{(k)} - z^*$ and using the previous Lemma, it follows that

$$r_y^{(k)} = B_5 + B_6 + B_7 + o\left(\left(r^{(k)}\right)^3\right), \tag{10}$$

where

$$\begin{cases} B_5 = 2M_2 r_x^{(k)} r^{(k)}, \\ B_6 = 2M_2^2 r_x^{(k)}\left(r^{(k)}\right)^2 - M_2\left(r^{(k)}\right)^2 - \left(4M_2^2 - 3M_3\right)\left(r_x^{(k)}\right)^2 r^{(k)}, \\ B_7 = -M_3\left(r^{(k)}\right)^3, \end{cases}$$

using (8), it follows that

$$r_y^{(k)} = \left(M_2^2 - \frac{1}{4}M_3\right)\left(r^{(k)}\right)^3 + o\left(\left(r^{(k)}\right)^4\right). \tag{11}$$

Expanding $F\left(y^{(k)}\right)$ about z^* we have

$$F\left(y^{(k)}\right) = F'\left(z^*\right)\left[r_y^{(k)} + o\left(\left(r_y^{(k)}\right)^2\right)\right].$$

This completes the proof of the Lemma.

Theorem 5. *The sequence* $\left(z^{(k)}\right)_{k\geqslant 0}$ *defined by*

$$z^{(k+1)} = y^{(k)} + \left(\left[F'\left(z^{(k)}\right)\right]^{-1} - 2\left[F'\left(x^{(k)}\right)\right]^{-1}\right) F\left(y^{(k)}\right),$$

converges to the zero of the function F *with convergence order six.*

Proof. Let $z^{(k+1)} = y^{(k)} + \left(\left[F'\left(z^{(k)}\right)\right]^{-1} - 2\left[F'\left(x^{(k)}\right)\right]^{-1}\right) F\left(y^{(k)}\right)$. Using (7), and applying Lemmas 3 and 4, it follows that

$$r^{(k+1)} = B_8 - B_9 + o\left(\left(r^{(k)}\right)^6\right), \tag{12}$$

where

$$\begin{cases} B_8 = 2M_2\left(2r_x^{(k)} - r^{(k)}\right) r_y^{(k)}, \\ B_9 = (4M_2^2 - 3M_3)\left[2\left(r_x^{(k)}\right)^2 - \left(r^{(k)}\right)^2\right] r_y^{(k)}. \end{cases}$$

Then, the error equation (12) on using (8) and (11) becomes

$$r^{(k+1)} = \left(4M_2^2 - \frac{3}{2}M_3\right)\left(M_2^2 - \frac{1}{4}M_3\right)\left(r^{(k)}\right)^5 + o\left(\left(r^{(k)}\right)^6\right), \tag{13}$$

which shows the sixth order of convergence. This completes the proof of the Theorem.

Remark that to prove $lim_{k\to+\infty} z^{(k)} = z^*$ solution of $LCP(q, M)$, it is equivalent to demonstrate the convergence of the iteration sequence $\{z^{(k)}\}_{k=0}^{+\infty}$ generated by previous Theorem and to show that for all $k \in$ IN, we have

$$z^{(k)} \in \mathring{S} = \{z \in \mathrm{IR}^n : z > 0,\ \ Mz + q > 0\}. \tag{14}$$

The relationship (14) is guaranteed by the following theorem.

Theorem 6. *If $z^{(0)} \in \mathring{S}$, then for all $k \in$ IN, we have $z^{(k)} \in \mathring{S}$.*

Proof. We aim to show that $\forall k \geqslant 0$, $x^{(k)} > 0$, $y^{(k)} > 0$, $z^{(k+1)} > 0$ and $w^{(k+1)} > 0$.

(A) Applying Lemma 2 and using the fact that

$$x^{(k)} = z^{(k)} - \frac{1}{2}\left[F'\left(z^{(k)}\right)\right]^{-1} F\left(z^{(k)}\right),$$

then we have

$$\begin{aligned} x^{(k)} &= z^{(k)} - \frac{1}{2}r^{(k)} + o\left(\left(r^{(k)}\right)\right) \\ &= \frac{1}{2}\left[z^{(k)} + z^*\right] + o\left(r^{(k)}\right) > 0. \end{aligned} \tag{15}$$

(B) Applying Lemmas 3 and 4, and using the fact that

$$y^{(k)} = z^{(k)} - \left[F'\left(x^{(k)}\right)\right]^{-1} F\left(z^{(k)}\right),$$

then we have

$$\begin{aligned} y^{(k)} &= z^{(k)} - \left[F'\left(z^*\right)\right]^{-1} F\left(z^{(k)}\right) + o\left(r^{(k)}\right) \\ &= z^{(k)} - r^{(k)} + o\left(r^{(k)}\right) \\ &= z^* + o\left(r^{(k)}\right) > 0. \end{aligned} \tag{16}$$

(C) By setting that

$$B_{10} = \left[F'\left(z^{(k)}\right)\right]^{-1} - 2\left[F'\left(x^{(k)}\right)\right]^{-1};$$

and applying Lemma 4 and Theorem 5, then we have

$$\begin{aligned} z^{(k+1)} &= y^{(k)} + \left(\left[F'\left(z^{(k)}\right)\right]^{-1} - 2\left[F'\left(x^{(k)}\right)\right]^{-1}\right) F\left(y^{(k)}\right) \\ &= y^{(k)} + B_{10}F'\left(z^*\right)\left[r_y^{(k)} + o\left(\left(r_y^{(k)}\right)^2\right)\right] \\ &= y^{(k)} - r_y^{(k)} + o\left(r_y^{(k)}\right) \\ &= z^* + o\left(r_y^{(k)}\right) > 0. \end{aligned} \tag{17}$$

(D) Using (17) and using the fact that $w^{(k+1)} = Mz^{(k+1)} + q$, then we have

$$w^{(k+1)} = w^* + o\left(r_y^{(k)}\right) > 0.$$

This completes the proof of the Theorem.

Corollary 7. *The sequence* $\left(z^{(k)}\right)_{k\geqslant 0}$ *converges to the solution of* $LCP(q, M)$.

Proof. Let $z^{(0)} \in \mathring{S}$. Applying the previous theorems, we have $z^{(k)} \in \mathring{S}$ for all $k \in IN$. Hence, it follows that $z^{(k)} > 0$, $w^{(k)} > 0$ and $z^{(k)}w^{(k)} \to 0$ when $k \to +\infty$. Then, the sequence $\left(z^{(k)}\right)_{k\geqslant 0}$ converges to z^* the solution of the Linear Complementarity Problem $LCP(q, M)$.

Now, we give the following algorithm for solving the linear complementarity problem.

Algorithm for solving LCP

Input:
M: The given matrix
q: The given column vector
ϵ: The error tolerance
Output:
An approximation to z^* and w^*;
begin
Step 0 : Select an initial point $z^{(0)} \in \mathring{S}$;
Step 1 : For $k = 0, 1, 2, ...$ until termination, do the following:
1. $w^{(k)} = Mz^{(k)} + q$;
2. $D(z^{(k)}) = diag\left(z_1^{(k)}, ..., z_n^{(k)}\right)$;
3. $D(w^{(k)}) = diag\left(w_1^{(k)}, ..., w_n^{(k)}\right)$;
4. $F(z^{(k)}) = \left(z_1^{(k)}w_1^{(k)}, ..., z_n^{(k)}w_n^{(k)}\right)^T$;
5. $F'\left(z^{(k)}\right) = D(z^{(k)})M + D(w^{(k)})$;
6. $x^{(k)} = z^{(k)} - \frac{1}{2}\left[F'\left(z^{(k)}\right)\right]^{-1} F\left(z^{(k)}\right)$;
7. $y^{(k)} = z^{(k)} - \left[F'\left(x^{(k)}\right)\right]^{-1} F\left(z^{(k)}\right)$;
8. $z^{(k+1)} = y^{(k)} + \left(\left[F'\left(z^{(k)}\right)\right]^{-1} - 2\left[F'\left(x^{(k)}\right)\right]^{-1}\right) F\left(y^{(k)}\right)$;
9. If $\left\|F(z^{(k+1)})\right\| \leqslant \epsilon$, then go to step 2, else go to step 1.10;
10. Set $z^{(k)} = z^{(k+1)}$, $k = k + 1$, and go to step 1.1;

Step 2 : Compute:
1. $z^* = z^{(k+1)}$;
2. $w^* = Mz^* + q$.

end.

Numerical tests and discussion

In this section, we provide numerical examples to demonstrate the efficiency of our method. To do so, we conducted the numerical experiments on some test problems. In the following, we will implement our algorithm in MATLAB 7.2 and run it on a personal computer with a 2.60 GHZ CPU processor and 3.5 Go memory. We stop the iterations if the condition $||F^{(k)}(z)|| \leqslant 10^{-6}$ is satisfied. In order to do this, we implement the MATLAB code of the following functions $F(z)$, $F'(z)$ and ModifiedNewton function for solving $F(z) = 0$ (see Appendix A). Now, let's take some known examples for solving LCP.

Example 1. Let us consider the following linear complementarity problem $LCP(q, M)$, find vector z satisfying $Mz + q \geqslant 0$, $z \geqslant 0$ and $z^T(Mz + q) = 0$, where $M = (m_{ij})_{1\leq i,j\leq n}$ such as $m_{ii} = 4$, $m_{i,i+1} = m_{i+1,i} = -1$ for all $i = 1, ..., n$ and zero in the rest and $q = (q_i)_{1\leq i\leq n}$ such as $q_i = -1$.

This example is used by C. Geiger [9]. In order to calculate the optimal solution of this problem by using the method suggested, we implement the MATLAB program for calculating number of iterations, time, and Error norm (see Appendix B). Table 1 lists number of iterations (noted "iter") required, time (noted "Time") taken and the error norm (noted "ErrorNorm") when solving the example 1.

Table 1
Displayed number of iterations required, time taken and the error norm when solving the example 1

n	Iter	Time	ErrorNorm
1	2	0	0
2	3	0	0
3	3	0	0
4	3	0	0
5	3	0	1.479818E-16
10	3	0	2.524921E-16
50	3	0	2.446870E-53
100	4	0	0
500	5	1.9656126	3.870577E-48
1000	5	7.8624504	1.415563E-30

Following examples would demonstrate the concept. Let us plot, the curve of the variation of the number of iterations (Fig. 1), the curve of the execution time (Fig. 2), and the curve of the relative residual (Fig. 3), according to the size of the problem. Computations are carried out with parameters: problem size n varies from 1 up to 300, with an increment of 2, and $z^{(0)} = (10^{-2}nI - M^{-1})\,q$. The tolerance ϵ is set to 10^{-6}. We create a script file and we type the code given in Appendix C. When we run the file, MATLAB displays the following plots (Figures 1-3).

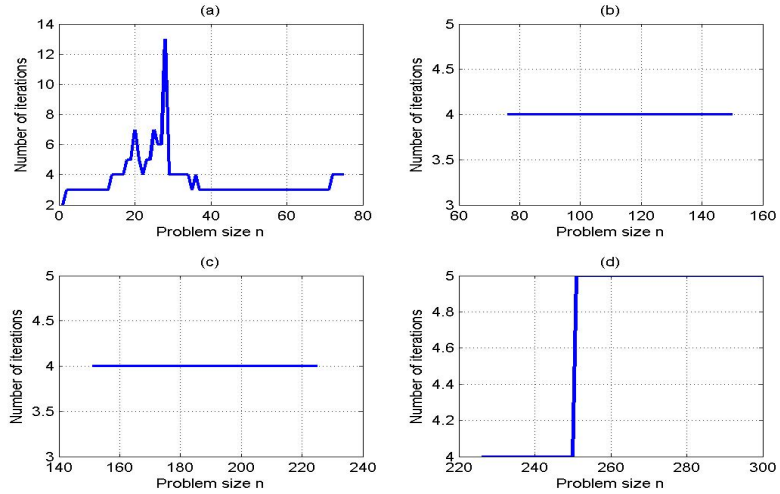

Fig. 1: Plot of the number of iterations according to size of problem.

Determining the number of iterations: in simulation, one major question is how many iterations are needed to reach the optimal solution in the results. Simulation as a tool provides an approximation of the actual relationship between the input and output variables. The precision of the approximation is

based on the number of iterations of the simulation done. More iterations lead to greater precision. But the relationship between iterations and precision depends on the relationship between the variables in the precision. In addition, the analyst must decide which output variable is the variable of interest, and what degree of precision is required. The next step is to determine the number of iterations required to achieve a solution to problem 1.

In the first figure, the y-axis shows number of iterations which varies between 1 and 5 (except (a) where the number of iterations varies between 1 and 13), and the x-axis is the size problem which varies between 1 and 300, with an increment of 1. Please note that these graphs show that the number of iterations is uniformly bounded for every value of $n \in \{1, 2, ..., 300\}$. Moreover, the number of iterations necessary to reach the optimal solution does not exceed 3 iterations when the size of the problem varies between 40 and 70, it does not exceed 4 iterations when n varies between 70 and 250, and it does not exceed 5 iterations when the size of the problem varies between 250 and 300, as shown in the Fig. 1, and this, regardless of the value of $n \in \{1, 2, ..., 300\}$. The figure shows also that the number of iterations required to achieve a solution ranges from 1 to about 13. Most of the time, about 4 or 5 iterations are required.

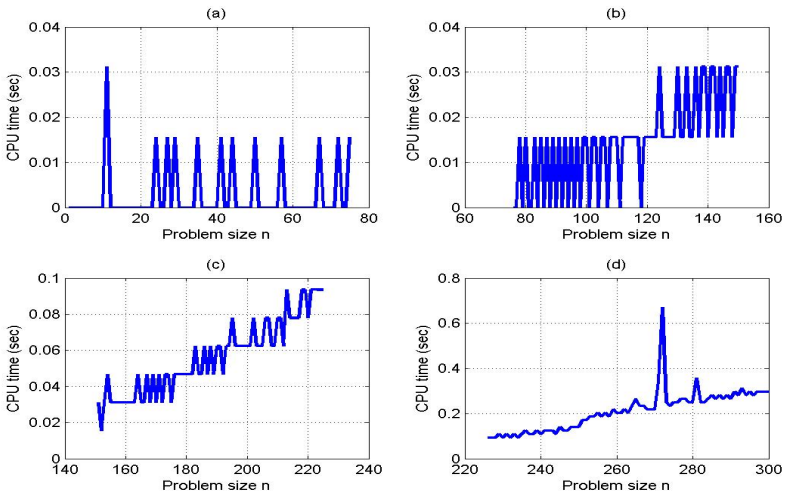

Fig. 2: Plot of Execution time according to problem size.

Analyzing the algorithm's performance: the stopwatch timer function cputime enable us to get information on how our algorithm is performing and help us to make interpretations. It is useful for measuring relative execution time and identifying specific performance bottlenecks in algorithm. The cputime function provides a robust measurement of the time required for algorithm execution. The Fig. 2 is to determine the execution time required to achieve a solution of problem 1. Note that the cputime function returns the total CPU time (in seconds) used by the algorithm from the time it was started. This number can overflow the internal representation and wrap around.

Fig. 2 shows that the execution time always varies between 0 and about 0.35 (sec) when the size of the problem varies between 1 and 300, with an increment of 1. Most of the time, between 0 and 0.015 (sec) are required. It can be noted that, if the size of problem n=300, the execution time CPU is lower than 0.30 (sec). Recall that although we can measure performance using the timeit or tic and toc functions, the cputime function is better for this purpose. Generally for CPU-intensive calculations run on Microsoft(R) Windows(R) machines, the elapsed time using cputime and using tic and toc are close in value, ignoring any first time costs. There are cases, however, that show a significant difference between these methods. For example, in the case of a Pentium 4 with hyperthreading running Windows, there can be a significant difference between the values returned by cputime versus tic and toc.

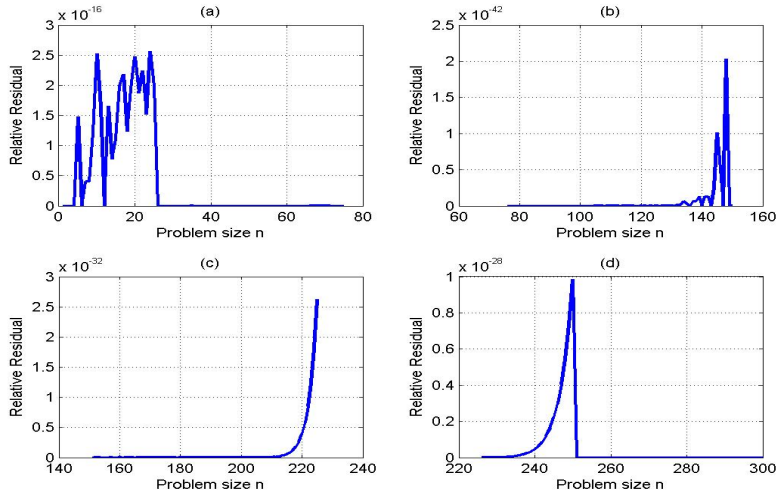

Fig. 3: Plot of Relative residual according to problem size.

From Fig. 3, it can be observed that a moderate increase of the error norm (while increasing n), in which the y-axis shows error norm which varies between 0 and $2.5\ 10^{-16}$ as shown in Fig. 2 (a), but most of the time, it varies between 0 and $1.0\ 10^{-28}$ as shown in Fig. 2 (b)-(d). The x-axis is the size problem which varies between 1 and 300, with an increment of 1.

Now we compare the results obtained by our method with that obtained by the Yu method [10] and the CHKS method [11]. In order to do this, we implement the MATLAB program for calculating the optimal solution z*, the final values F(z*), number of iterations and time (see Appendix D). The results are summarized in Tables 2-4, where Time denotes the total cost time (in second) for solving the problem; z* and F (z*) denote the final values of the iteration point and the function F(x); Iter denotes the iteration number when the algorithm terminates. From Tables 2-4, we can see that our method can comparable with the Yu method and the CHKS method from the iteration number and the CPU times, which shows that the performance of our method is effective.

Table 2
Numerical results of our method (first example).

	z^*	$F(z^*)$	Iter	Time(s)
n=4	(0.363636,0.454545, 0.454545,0.363636)	(3.230000E-16,4.037000E-16, 6.056000E-16,-1.615000E-16)	4	0
n=8	(0.366013,0.464052, 0.490196,0.496732, 0.496732,0.490196, 0.464052,0.366013)	(1.625000E-16,0,0,1.103000E-16 1.103000E-16,0,0,0)	4	0.031200

Table 3
Numerical results of the Yu method (first example).

	z^*	$F(z^*)$	Iter	Time(s)
n=4	(0.363636,0.454545, 0.454545,0.363636)	(0, 0, -1.11022E-16, 0)	5	0.031
n=8	(0.366013,0.464052, 0.490196,0.496732, 0.496732,0.490196, 0.464052,0.366013)	(-1.11022E-16, 0, 0, 0, 0,-1.11022E-16, 0, 0)	5	0.016

Table 4
Numerical results of the CHKS method (first example).

	z^*	$F(z^*)$	Iter	Time(s)
n=4	(0.363636,0.454545, 0.454545,0.363636)	(-6.72751E-12, -5.38214E-12, -5.38214E-12, -6.72751E-12)	5	0.016
n=8	(0.366013, 0.464052, 0.490196, 0.496732, 0.496732, 0.490196, 0.464052, 0.366013)	(-6.68399E-12, -5.27156E-12, -4.9909E-12, -4.92495E-12, -4.92495E-12, -4.9909E-12, -5.27178E-12, -6.68399E-12)	5	0.031

Example 2. Let us consider the following linear complementarity problem $LCP(q, M)$ where $M = (m_{ij})_{1\leq i,j\leq n}$ such as $m_{ij} = i\delta_{ij}/n$ where δ is the Kronecker's delta ($\delta_{ii} = 1$ and $\delta_{ij} = 0$ if $i \neq j$) and $q = (q_i)_{1\leq i\leq n}$ such as $q_i = -1$.

This example is used by Z. Yu [10]. To illustrate the convergence behavior and computational efficiency, we will compare four methods that work directly on the linear complementarity problem: Lemke method noted by (LM), Chen method [11] noted by (CHKSM), Yu method [10] noted by (YQM), and method proposed in this paper noted by (EKM). We stop the iterations if the condition $||F\left(z^{(k)}\right)|| \leqslant 10^{-6}$ is satisfied. For every method, we analyze the number of iterations needed to converge to the solution. In the comparison of performance of methods, we also include CPU time utilized in the execution, where Iter denotes the iteration number when the algorithm terminates; and Time denotes the total cost time (in second) for solving the $LCP(q, M)$. Note the exact solution of $LCP(q, M)$ is given by $z_i = \frac{n}{i}$.

Firstly, we study the performance of the proposed method in the case of size of problem n=4 and n=8. In particular, we compare the results obtained by our method with that obtained by the Yu method [10] and the CHKS method [11]. In order to do this, we implement the MATLAB program for calculating the optimal solution z*, the final values F(z*), number of iterations and time (see Appendix E). The results are summarized in Tables 5-7, where Time denotes the total cost time (in second) for solving the problem; z* and F (z*) denote the final values of the iteration point and the function F(x); Iter denotes the iteration number when the algorithm terminates. From Tables 5-7, we can see that our method can comparable with the Yu method and the CHKS method from the iteration number and the CPU times, which shows that the performance of our method is effective.

Table 5
Numerical results of our method (second example).

	z^*	$F(z^*)$	Iter	Time(s)
n=4	(4,2,1.333333,1)	(0,0,0,0)	2	0
n=8	(8,4,2.666667,2, 1.6,1.333333,1.142857,1)	(0,0,-0.592100E-15,0, 0,0.592100E-15,0,0)	3	0

Table 6
Numerical results of the Yu method (second example).

	z^*	$F(z^*)$	Iter	Time(s)
n=4	(4,2,1.3333,1)	(-1.1102E-16,-1.1102E-16,0,0)	5	0.016
n=8	(8,4,2.6667,2, 1.6,1.3333,1.1428,1)	(2.2204E-16,0,0,2.2204E-16, -1.1102E-16,2.2204E-16,0,0)	8	0.047

Table 7
Numerical results of the CHKS method (second example).

	z^*	$F(z^*)$	Iter	Time(s)
n=4	(4,2,1.33333,1)	(-6.11511E-13,-1.22324E-12, -1.83498E-12,-2.44638E-12)	5	0.016
n=8	(8,4,2.66667,2, 1.6,1.33333,1.14286,1)	(-3.16676E-10,-6.33363E-10, -9.50053E-10,-1.26674E-9, -1.58342E-9,-1.90009E-9, -2.21673E-9, -2.53336E-9)	7	0.031

Secondly, we study the performance of the proposed method in the case of size of problem n=100, n=500 and n=1000. In particular, we compare the results obtained by our method with that obtained by the Lemke method, the Yu method [10] and the CHKS method [11]. The test results of this example are summarized in Table 8 (we implement the same MATLAB program in Appendix E by replacing "nn=[4,8]" with "nn=[100,500,1000]"). From Table 8, we can see that our method can comparable with the Lemke method, the Yu method, and the CHKS method from the iteration number and the CPU times. However, our method can solve the problem while the Lemke method and CHKS method failed to solve the problem, which shows that the performance of our method is effective.

Table 8
Numerical results of the second example with n=100, 500 and 1000.

Method	n	Iter	Time(s)	ErrorNorm
LM	100	25	66,2334626	*****
	500	Fail	*****	*****
	1000	Fail	*****	*****
CHKSM	100	22	4,4141276	7,100257E-11
	500	43	445,2737276	5,890708E-10
	1000	Fail	*****	*****
YQM	100	19	0,2928386	7,100130E-08
	500	37	29,6834786	5,890602E-07
	1000	51	108,5587586	8,548258E-07
EKM	100	6	0.018086	5.751105E-16
	500	7	1.977462	4.771388E-15
	1000	8	7.235814	6.924089E-15

Example 3. We will now provide a practical example of this paper to illustrate the effectiveness of the proposed method. To do so, we will consider a bio-economic equilibrium model which describes the dynamics of a fish population fished by several fishermen seeking to maximize their profits. Specifically, we consider a bio-economic equilibrium model of a fish population exploited by several fishermen represented by their fishing efforts $(E_i)_{i=1,\ldots,n}$, where n is the number of fishermen. For more details, we refer to Y. EL Foutayeni et al. [12]. The objective is to find the fishing effort E_i^* maximizing each fisherman's profit $\pi_i^*(E^*)$, at biological equilibrium, without any consultation between the fishermen. All of them have to respect two constraints, the first one is the sustainable management of the resources and the second one is the preservation of the biodiversity.

Each fisherman strives to maximize its profit choosing a fishing effort strategy. With these considerations, our problem leads to the following Generalized Nash Equilibrium Problem (GNEP):

$$\text{Each fisherman } i \in \{1, ..., n\} \text{ must solve problem } (P_i)$$

$$(P_i) \begin{cases} \max \pi_i(E) = -\frac{pK}{r} q_i^2 E_i^2 + pKq_i(\frac{pKq_i - c_i}{pKq_i} - \frac{1}{r}\sum_{j=1, j\neq i}^{n} q_j E_j)E_i \\ \text{subject to} \\ \qquad \frac{1}{r} q_i E_i < -\frac{1}{r}\sum_{j=1, j\neq i}^{n} q_j E_j + 1 \\ \qquad E_i \geqslant 0 \\ \qquad (E_j)_{j=1,...,n; j\neq i} \text{ are given.} \end{cases}$$

where p is the price of the fish population; r is the intrinsic growth rate; K is the area's environmental carrying capacity or saturation level for a given fish population; q_i is the catchability coefficient of fisherman i; and c_i is the harvesting costs per fishing effort employed by fisherman i. For more details and description of variables, refer to Y. EL Foutayeni et al. [12].

We recall that the Generalized Nash Equilibrium Problem (*GNEP*) is an extension of the Nash Equilibrium Problem (*NEP*), in which each fisherman's strategy (fishing effort) set is dependent on the rival fishermen strategies. Mathematically, $(E_1^*, ..., E_n^*)$ is called generalized Nash equilibrium point, if and only if, E_i^* is a solution of the problem (P_i) for $(E_j^*)_{j=1,...,n; j\neq i}$ are given.

The fishing effort $E^* = (E_1^*, ..., E_n^*)$ depends on: (a) the catchability coefficients q_i; (b) the costs of fishing c_i; and (c) the price of fish population p. It was shown that (see Y. EL Foutayeni et al. [12]) solving Generalized Nash Equilibrium Problem (GNEP) is equivalent to solving LCP(q,M) defined by:

Find a column vector $z = (qE^*, 0)^T \in \text{IR}^{n+1}$ satisfying

$$\begin{cases} z \geqslant 0 \\ w = Mz + q \geqslant 0 \\ z^T w = 0 \end{cases} \tag{18}$$

where $M = \begin{bmatrix} 2 & 1 & 1 & 1 & 1 & 0 \\ 1 & 2 & 1 & 1 & 1 & 0 \\ 1 & 1 & 2 & ... & 1 & 0 \\ ... & ... & ... & ... & 1 & 0 \\ 1 & 1 & 1 & ... & 2 & 0 \\ -1 & -1 & -1 & ... & -1 & 1 \end{bmatrix}$ and $q = \begin{bmatrix} r(\frac{c_1}{pKq_1} - 1) \\ r(\frac{c_2}{pKq_2} - 1) \\ ... \\ ... \\ r(\frac{c_n}{pKq_n} - 1) \\ r \end{bmatrix}$.

For solving the linear complementarity problem (18) we first show that the matrix M is a P-matrix. to do so, we denote by $(M_i)_{i=1,...,n+1}$ the submatrix of M, then we have $det(M_i) = i + 1 > 0$ for all $i = 1, ..., n$ and $det(M) = n + 1$. So the matrix M is P-matrix and therefore the $Lcp(q, M)$ admits one and only one solution.

In the following, we will implement the proposed method in MATLAB 7.2. To illustrate the convergence behavior and computational efficiency, we will compare our method noted by (EKM) with the existing methods Lemke noted by (LM), Chen [11] noted by (CHKSM), and Yu [10] noted by (YQM). We stop the iterations if the condition $||F\left(z^{(k)}\right)|| \leqslant 10^{-6}$ is satisfied. For every method, we analyze the number of iterations needed to converge to the solution. In the comparison of performance of methods (see Table 9), we also include CPU time utilized in the execution. When looking for an approximation with six significant digits, we obtain that using the proposed method who does not require a lot of iterations and not a lot of CPU time by against, the other methods require a lot of arithmetic operations, and therefore, it is too difficult, time consuming, and expensive to find an approximate solution of the exact solution. However, in the particular case when $n = 4$ (four different fishermen), the fishing effort which maximize the profit is given by (see Y. EL Foutayeni et al. [12])

$$E^* = \begin{bmatrix} 30.252051 \text{ unit of effort} \\ 60.504659 \text{ unit of effort} \\ 60.504105 \text{ unit of effort} \\ 60.504103 \text{ unit of effort} \end{bmatrix}$$

and the corresponding profit π^* is the following

$$\pi^* = \begin{bmatrix} 17.227042\ Euros \\ 34.454718\ Euros \\ 51.681129\ Euros \\ 17.227042\ Euros \end{bmatrix}$$

Table 9 compares the performance of the proposed method with the existing methods Lemke, Chen, and Yu. Column 2 shows the approximation to the fishing effort which maximize the profit; column 3 reports the number of iterations; and column 4 reports the number of times. The numerical results reported in Table 9 show that the proposed method performs well for this example. In particular, we can see that it can produce an approximate solution with fewer iterations and shorter CPU-time for this example and can solve large-scale problems.

Table 9
Numerical results

Method	Approximation to the solution (Fishing effort E)	Iter	Time(s)
LM	(30.252000, 60.504123, 60.504846, 60.504482)	12	0.009567
CHKSM	(30.252051, 60.504659, 60.504105, 60.504103)	05	0.016000
YQM	(30.252051, 60.504659, 60.504105, 60.504103)	05	0.031000
EKM	(30.252051, 60.504659, 60.504105, 60.504103)	02	0.000987

where Iter denotes the iterations number when the algorithm terminates; and Time denotes the total cost time (in second) for solving the linear complementarity problem (18). From this table, one can see that the proposed method (EKM) for solving the linear complementarity problem converges very rapidly relative to (LM), (CHKSM) and (YQM) from the iteration number and the CPU times, which shows that the performance of our method is effective.

Conclusion

In this paper we have given an interior-point method for solving linear complementarity problems. This method converges very rapidly compared to all other methods. The results discussed in this paper are very favorable. The results of numerical experiments on some test problems and practical example showed that this method works well for solving LCP. With regard to the nice theoretical results of our algorithm, the computational results reported are very encouraging. We expect our algorithm can also solve large-scale problems well. As one of the comments that we wish to emphasize is that we want to show in the future that we can use this new method for solving nonlinear complementarity problems.

Acknowledgements. The authors are sincerely grateful to the anonymous referees and the Corresponding Editor for their valuable comments, which lead to a substantial improvement in the contents of this paper.

Appendix A

In this appendix we give the MATLAB program for functions F, F', and ModifiedNewton function for solving $F(z) = 0$.

```
%-------------------------------------------------------------------
% MATLAB program for F(z) = z(Mz + q)
%-------------------------------------------------------------------
function fun=F(z)
    global n q M
    w=M*z+q;
fun=z.*w;

%-------------------------------------------------------------------
% MATLAB program for F'(z) = ZM + W
%-------------------------------------------------------------------
function fun=dF(z)
    global n q M
    Z=diag(z);
    W=diag(M*z+q);
fun=Z*M+W;

%-------------------------------------------------------------------
% MATLAB program for the function ModifiedNewton for solving F(z) = 0
%-------------------------------------------------------------------
function [z,nb]=ModifiedNewton(F,dF,z0,eps)
    global n q M
    err=1;
    zt=z0;z=z0;
    nb=0;
    while (err>eps)
        nb=nb+1;
        x=z-0.5*(feval(dF,z)\feval(F,z));
        y=z-feval(dF,x)\feval(F,z);
        z=y+(feval(dF,z)-2*feval(dF,x))\feval(F,y);
        err=norm(z-zt);
        zt=z;
    end
```

Appendix B

In this appendix we give the MATLAB program for Table 1.

```
%-------------------------------------------------------------------
% MATLAB program for calculating iter, CPU and ErrorNorm
%-------------------------------------------------------------------
eps=1e-6;
global n q M
iter=[];ErrorNorm=[];CPU=[];nn=[1,2,3,4,5,10,50,100,500,1000];
for n=nn
    q = zeros(n,1); M = zeros(n, n); z0=zeros(n,1);
    for i = 1:n
        q(i) = -1;
        M(i,i) = 4;
    end
```

```
    for i=1:n-1 M(i,i+1) = -1; end
    for i=1:n-1 M(i+1,i) = -1; end
    z0=-M\q+0.01*n*q;
     t=cputime;
     [z,nbiter]=ModifiedNewton('F','dF',z0,eps);
     CPU=[CPU,cputime-t];
     iter=[iter,nbiter];
     ErrorNorm=[ErrorNorm,norm(F(z))];
end
```

Appendix C

In this appendix we give the MATLAB program for plotting Figures 1-3.

```
%————————————————————————————————————————
% MATLAB program for plotting Figures 1-3
%————————————————————————————————————————
eps=1e-6;
global n q M
iter=[];ErrorNorm=[];CPU=[];nn=1:300;
for n=nn
    q = zeros(n,1); M = zeros(n, n); z0=zeros(n,1);
    for i = 1:n
        q(i) = -1;
        M(i,i) = 4;
        end
    for i=1:n-1 M(i,i+1) = -1; end
    for i=1:n-1 M(i+1,i) = -1; end
   z0=-M\q+0.01*n*q;
    t=cputime;
    [z,nbiter]=ModifiedNewton('F','dF',z0,eps);
    CPU=[CPU,cputime-t];
    iter=[iter,nbiter];
    ErrorNorm=[ErrorNorm,norm(F(z))];
end
N=300/4;
for i=nn(1):nn(N)
    NN1(i)=nn(i); ITER1(i)=iter(i); CPU1(i)=CPU(i); EN1(i)=ErrorNorm(i);
end
for i=nn(N+1):nn(2*N)
    NN2(i-N)=nn(i); ITER2(i-N)=iter(i); CPU2(i-N)=CPU(i); EN2(i-N)=ErrorNorm(i);
end
for i=nn(2*N+1):nn(3*N)
    NN3(i-2*N)=nn(i); ITER3(i-2*N)=iter(i); CPU3(i-2*N)=CPU(i); EN3(i-2*N)=ErrorNorm(i);
end
for i=nn(3*N+1):nn(4*N)
    NN4(i-3*N)=nn(i); ITER4(i-3*N)=iter(i); CPU4(i-3*N)=CPU(i); EN4(i-3*N)=ErrorNorm(i);
end
%————————————Plotting Figure 1————————————
A=figure;
set(A,'Units','Normalized','Outerposition',[0 0 0.45 0.45]);
subplot(2,2,1)
```

```
plot(NN1,ITER1,'linewidth',2);%AREA(iter)
grid on
xlabel('Problem size n');
ylabel('Number of iterations');
title('(a)');
subplot(2,2,2)
plot(NN2,ITER2,'linewidth',2);%AREA(iter)
grid on
xlabel('Problem size n');
ylabel('Number of iterations');
title('(b)');
subplot(2,2,3)
plot(NN3,ITER3,'linewidth',2);%AREA(iter)
grid on
xlabel('Problem size n');
ylabel('Number of iterations');
title('(c)');
subplot(2,2,4)
plot(NN4,ITER4,'linewidth',2);%AREA(iter)
grid on
xlabel('Problem size n');
ylabel('Number of iterations');
title('(d)');
saveas(A,'Figure1.fig')
saveas(A,'Figure1.jpg')
%————————————Plotting Figure 2————————————
B=figure;
set(B,'Units','Normalized','Outerposition',[0 0 0.45 0.45]);
subplot(2,2,1)
plot(NN1,CPU1,'linewidth',2);%AREA(CPU)
grid on
xlabel('Problem size n');
ylabel('CPU time (sec)');
title('(a)');
subplot(2,2,2)
plot(NN2,CPU2,'linewidth',2);%AREA(CPU)
grid on
xlabel('Problem size n');
ylabel('CPU time (sec)');
title('(b)');
subplot(2,2,3)
plot(NN3,CPU3,'linewidth',2);%AREA(CPU)
grid on
xlabel('Problem size n');
ylabel('CPU time (sec)');
title('(c)');
subplot(2,2,4)
plot(NN4,CPU4,'linewidth',2);%AREA(CPU)
grid on
xlabel('Problem size n');
```

```
ylabel('CPU time (sec)');
title('(d)');
saveas(B,'Figure2.fig')
saveas(B,'Figure2.jpg')
%————————————Plotting Figure 3————————————
C=figure;
set(C,'Units','Normalized','Outerposition',[0 0 0.45 0.45]);
subplot(2,2,1)
plot(NN1,EN1,'linewidth',2);%AREA(ErrorNorm)
grid on
xlabel('Problem size n');
ylabel('Relative Residual');
title('(a)');
subplot(2,2,2)
plot(NN2,EN2,'linewidth',2);%AREA(ErrorNorm)
grid on
xlabel('Problem size n');
ylabel('Relative Residual');
title('(b)');
subplot(2,2,3)
plot(NN3,EN3,'linewidth',2);%AREA(ErrorNorm)
grid on
xlabel('Problem size n');
ylabel('Relative Residual');
title('(c)');
subplot(2,2,4)
plot(NN4,EN4,'linewidth',2);%AREA(ErrorNorm)
grid on
xlabel('Problem size n');
ylabel('Relative Residual');
title('(d)');
saveas(C,'Figure3.fig')
saveas(C,'Figure3.jpg')
```

Appendix D

In this appendix we give the MATLAB program for Tables 2-4.

```
%————————————————————————————————————
% MATLAB program for Tables 2-4
%————————————————————————————————————
eps=1e-6;
global n q M
iter=[];ErrorNorm=[];CPU=[];nn=[4,8];
for n=nn
    q = zeros(n,1); M = zeros(n, n); z0=zeros(n,1);
    for i = 1:n
        q(i) = -1;
        M(i,i) = 4;
        z0(i) = 1;
    end
    z0(1)=2;
```

```
        for i=1:n-1 M(i,i+1) = -1; end
        for i=1:n-1 M(i+1,i) = -1; end
         t=cputime;
         [z,nbiter]=ModifiedNewton('F','dF',z0,eps);
         CPU=[CPU,cputime-t];
         iter=[iter,nbiter];
         ErrorNorm=[ErrorNorm,norm(F(z))];
end
```

Appendix E

In this appendix we give the MATLAB program for Table 5.

```
%-------------------------------------------------------------------
% MATLAB program for Table 5
%-------------------------------------------------------------------
eps=1e-6;
global n q M
iter=[];ErrorNorm=[];CPU=[];nn=[4,8];
for n=nn
        q = zeros(n,1); M = zeros(n, n); z0=zeros(n,1);
        for i = 1:n
                q(i) = -1;
                M(i,i) = i/n;
        end
        z0=-M\q+0.01*n*q;
         t=cputime;
         [z,nbiter]=ModifiedNewton('F','dF',z0,eps);
         CPU=[CPU,cputime-t];
         iter=[iter,nbiter];
         ErrorNorm=[ErrorNorm,norm(F(z))];
end
```

References

[1] R.W. Cottle, The Principal Pivoting Method of Quadratic Programming, in G.B. Dantzig and A.F Veinott (Eds.), Mathematics of Decision Sciences, Part 1, AMS, Providence, RI, (1968) 142-162.

[2] S.J. Chung, NP-completeness of the linear complementarity problem, J. Optim. Theory Appl., 60 (1989) 393-399.

[3] R.W. Cottle, J.S. Pang, R.E. Stone, The linear complementarity problem, Academic Press, 1992.

[4] K.G. Murty, Linear Complementarity, Linear and Nonlinear Programming, Helderman-Verlag, 1988.

[5] Y. EL Foutayeni, M. Khaladi, A Min-Max Algorithm for Solving the Linear Complementarity Problem, J. Math. Sci. Appl, 1 (2013) 6-11.

[6] Y. EL Foutayeni, M. Khaladi, General Characterization of a Linear Complementarity Problem, Amer. J. Model. Optim., 1 (2013) 1-5.

[7] Y. EL Foutayeni, M. Khaladi, Using vector divisions in solving the linear complementarity problem, J. Comput. Appl. Math., 236 (2012) 1919-1925.

[8] Y. EL Foutayeni, M. Khaladi, A New Interior Point Method for Linear Complementarity Problem, Appl. Math. Sci., 4 (2010) 3289-3306.

[9] C. Geiger, C. Kanzow, On the resolution of monotone complementarity problems, Comput. Optim. Appl. 5 (1996) 155-173.

[10] Z. Yu, Y. Qin, A cosh-based smoothing Newton method for P_0 nonlinear complementarity problem, Nonlinear Anal. Real World Appl., 12 (2011) 875-884.

[11] B. Chen, P.T. Harker, A noninterior-point continuation method for linear complementarity problems, SIAM J. Matrix Anal. Appl. 14 (1993) 1168-1190.

[12] Y. EL Foutayeni, M. Khaladi, Fishermen's Profits Maximization: Case of Generalized Nash Equilibrium of a Non-symmetrical Game, J. Acta Biotheoretica Springer, 62 (2014) 325-338, 10.1007/s10441-014-9223-y.