sciendo

Műszaki Tudományos Közlemények vol. 11. (2019) 43–46.
https://doi.org/10.33894/mtk-2019.11.07
Hungarian: https://doi.org/10.33895/mtk-2019.11.07
https://eda.eme.ro/handle/10598/31244

EME MTSZ MTK

NUMERICAL RESULTS FOR THE GENERAL LINEAR COMPLEMENTARITY PROBLEM

Zsolt DARVAY,[1] Ágnes FÜSTÖS [2]

Babeș-Bolyai University, Faculty of Mathematics and Computer Science, Cluj-Napoca, Romania

[1] *darvay@cs.ubbcluj.ro*

[2] *fustosagi@yahoo.com*

Abstract

In this article we discuss the interior-point algorithm for the general complementarity problems (LCP) introduced by Tibor Illés, Marianna Nagy and Tamás Terlaky. Moreover, we present a various set of numerical results with the help of a code implemented in the C++ programming language. These results support the efficiency of the algorithm for both monotone and sufficient LCPs.

Keywords: *interior-point algorithm, general linear complementarity problem, numerical results, sufficient matrix, object-oriented programming.*

1. Introduction

Linear complementarity problems (LCPs) can be used to solve various practical problems. The problem is defined by a matrix M, which describes a linear relationship, but also a complementarity condition must hold.

The LCP is $\mathbb{NP}$-complete, so without some conditions on M we cannot efficiently find the solution. There are algorithms that solve the LCP in polynomial time if the matrix M is positive semi-definite, or if the matrix is $P_*(\kappa)$ (although in this case the complexity also depends on κ), but no polynomial algorithm for checking the $P_*(\kappa)$ property was given.

Tibor Illés, Marianna Nagy and Tamás Terlaky [1, 2, 3] provided a method for modifying the interior-point algorithms so that the general linear complementarity problem could be solved in polynomial time. This means that we calculate an approximate solution or conclude that the $P_*(\kappa)$ property doesn't hold.

We implemented the algorithm in the C++ programming language.

2. Description of the LCP

In case of the LCP we want to find vectors $x, s \in \mathbb{R}^n$ that satisfy:

$$\begin{cases} Mx + q = s, \\ xs = 0, \\ x \geq 0, s \geq 0, \end{cases} \tag{1}$$

where $q \in \mathbb{R}^n$, $M \in \mathbb{R}^{n\times n}$ and xs denotes the component-wise product of vectors x and s.

3. The $P_*(\kappa)$ property

The class of $P_*(\kappa)$ matrices was first introduced by Kojima et al. [4] as a generalization of positive semi-definite matrices.

A matrix $M \in \mathbb{R}^{n\times n}$ is $P_*(\kappa)$ if the following condition holds for any value of $x \in \mathbb{R}^n$:

$$(1 + 4\kappa) \sum_{i\in\mathcal{I}_+(x)} x_i(Mx)_i + \sum_{i\in\mathcal{I}_-(x)} x_i(Mx)_i \geq 0. \tag{2}$$

A matrix $M \in \mathbb{R}^{n\times n}$ is P_* if it is $P_*(\kappa)$ for some positive κ, i.e.

$$P_* = \bigcup_{\kappa>0} P_*(\kappa).$$

4. The P_0 property

A matrix $M \in \mathbb{R}^{n\times n}$ is P_0 if it does not have a principal minor, which is negative.

A matrix $M \in \mathbb{R}^{n\times n}$ is P_0 if and only if

$$M' = \begin{bmatrix} -M & I \\ S & X \end{bmatrix}$$

is nonsingular for any positive diagonal matrices $X, S \in \mathbb{R}^{n\times n}$.

5. Other relations used in the algorithm

The algorithm follows the central trajectory and determines the next point in each iteration using a Newton step.

The system of equations, which defines the Newton step, is the following:

$$\begin{aligned} -M\Delta x + \Delta s &= 0, \\ s\Delta x + x\Delta s &= a. \end{aligned} \qquad (3)$$

If M is a P_0 matrix then the previous system has a unique solution and the Newton step with length α is given by the relations

$$x(\alpha)=x+\alpha\Delta x,\ s(\alpha)=s+\alpha\Delta s. \qquad (4)$$

The distance from the central path is determined by the following proximity measure:

$$\delta_c(xs,\mu) = \left\| \sqrt{\frac{xs}{\mu}} - \sqrt{\frac{\mu}{xs}} \right\| \qquad (5)$$

If the proximity measure exceeds a predetermined upper bound ($\delta_c(xs,\mu) < \tau$), then the iterates are close to the central path. In this case, we reduce the barrier parameter μ.

To approximate the local κ, we use the following function in each step:

$$\kappa(x) = -\frac{1}{4}\frac{x^T Mx}{\sum_{i\in J_+(x)} x_i (Mx)_i}. \qquad (6)$$

If, during an inner iteration, the decrease of the proximity measure is not sufficient, i.e.

$$\delta_c^2(xs,\mu) - \delta_c^2(x(\bar{\alpha})s(\bar{\alpha}),\mu) < \frac{5}{3(1+4\kappa)}, \qquad (7)$$

then the matrix, which defines the LCP is not a $P_*(\kappa)$ matrix for the local κ. In this case, we calculate κ for the new step direction vector Δx [1].

6. Stopping conditions

During the algorithm, we study several different conditions. If one of these holds, then different conclusions can be drawn. We study the following:

1. The complementarity gap reaches a predefined threshold: $x^T s < \epsilon$.
2. The new κ is not defined, i.e. there is no positive product $\Delta x_i \Delta s_i$ when calculating formula (2). In this case, M is not a P_* matrix.
3. The new κ exceeded the predefined $\tilde{\kappa}$ fupper bound: $\kappa(\Delta x) > \tilde{\kappa}$. In this case, M is not a $P_*(\tilde{\kappa})$ matrix.

7. The algorithm

We define the algorithm for the general LCP [1] from the implementation point of view as follows:

Input: *$\tilde{\kappa} > 0$ upper bound for κ; $\tau \geq 2$ proximity parameter; $\varepsilon > 0$ accuracy parameter; $0 < \sigma < 1$ barrier update parameter; (x_0, s_0) initial point, $\mu_0 > 0$ s.t. $\delta_c(x_0 s_0, \mu) < \tau$.*

Output: *(x, s) the solution of the LCP or a message confirming that the matrix is not $P_*(\kappa)$.*

begin
$x := x_0$; $s := s_0$; $\mu := \mu_0$; $\kappa := 0$;
while *$x^T s > \varepsilon$* **do begin**
$\mu = \sigma (x^T s)/n$;
while *$\delta_c(xs, \mu) \geq \tau$* **do begin**
calculate *$(\Delta x, \Delta s)$* with *$a = \mu e - xs$;*
if *M is singular* **then**
return *the matrix is not P_0;*
end if
$\bar{\alpha}$= maximumStep(x, s, μ);
if *$\delta_c^2(xs, \mu) - \delta_c^2(x(\bar{\alpha})\, s(\bar{\alpha}), \mu) < 5/(3(1+4\kappa))$* **then**
calculate *κ(Δx);*
if *κ(Δx)* is not defined **then**
return *the matrix is not P_*;*
end if
if *$\kappa(\Delta x) > \tilde{\kappa}$)* **then**
return *the matrix is not $P_*(\tilde{\kappa})$;*
end if
κ = κ(Δx);
end if
x = x(ᾱ); s = s(ᾱ);
end
end
end.

The function for calculating the maximum step size is given as follows:

function maximumStep(x, s, μ):

$$\alpha_1 = \min\left\{ -\frac{x_i}{\Delta x_i} \;\middle|\; \Delta x_i < 0 \right\};$$

$$\alpha_2 = \min\left\{ -\frac{s_i}{\Delta s_i} \;\middle|\; \Delta s_i < 0 \right\};$$

$$\alpha = \min\{\alpha_1, \alpha_2\};$$

nr_it = 0;
while *$(\delta_c(x(\alpha/2)\, s(\alpha/2), \mu) < \delta_c(x(\alpha)\, s(\alpha), \mu)$* **and** *nr_it < 10)* **do begin**
α=α/2;

```
    nr_it = nr_it + 1;
  end
  return α;
end
```

We implemented the algorithm in the C++ programming language, using the code introduced in the publication [5], within the Visual Studio integrated development environment.

8. Numerical results

8.1. The "speed" of convergence based on the changes of σ

In the implementation of the algorithm, μ is calculated in each iteration using the formula $\mu = \sigma (x^T s) / n$. Table 1. shows how the choice of σ influences the number of iterations. We studied the sufficient LCPs published on the webpage [6]. The authors of [7] were the first ones who presented numerical results to these sufficient LCPs. The following notations are used in the table: M = matrix, σ = sigma, O = number of outer iterations, ∑I = sum of all inner iterations, AI = average number of inner iterations.

Table 1. *Results for sufficient LCPs*

M	σ	O	∑I	AI
HEF_10_01	0.15	7	24	3.429
	0.4	13	26	2
	0.6	23	46	2
	0.9	115	115	1
HEF_10_02	0.15	7	19	2.714
	0.4	13	26	2
	0.6	23	46	2
	0.9	115	115	1
HEF_20_01	0.15	7	23	3.286
	0.4	13	28	2.154
	0.6	23	46	2
	0.9	115	115	1
HEF_20_02	0.15	7	18	2.571
	0.4	13	27	2.077
	0.6	23	46	2
	0.9	115	115	1

Below we present the results for the following monotone LCP presented in article [8]:

$$M = \begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 0 & 1 \\ 1 & 0 & 1 & 2 \\ -1 & -1 & -2 & 0 \end{pmatrix}, \quad q = \begin{bmatrix} -8 \\ -6 \\ -4 \\ 3 \end{bmatrix}$$

The results are summarized in Table 2. The decrease of the parameter σ causes the decrease of the total number of outer and inner iterations, but the average number of inner iterations increases.

In the following section we analyze the lower bound of the parameter κ.

Table 2. *Results for monotone LCP*

σ	O	∑I	AI
0.15	7	28	4
0.4	13	31	2.384
0.6	24	31	1.291
0.9	114	118	1.035

8.2. The exponential increase of the lower bound of κ

The study of the following matrix was suggested by Zsolt Csizmadia. In [9] the authors proved that the matrix is sufficient, but the corresponding κ parameter may increase exponentially depending on the size of the matrix.

$$M = \begin{pmatrix} 1 & 0 & 0 & \cdots & \ldots & 0 \\ -1 & 1 & 0 & \ldots & \ldots & 0 \\ -1 & -1 & 1 & \ldots & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & 0 \\ -1 & -1 & -1 & \cdots & -1 & 1 \end{pmatrix}$$

Based on this matrix, we studied the LCP using $q = [0\ 1\ \ldots\ n\text{-}1]^T$.

In the case of sufficient matrices, in general, we cannot determine the value of the parameter κ, but we can provide a lower bound for it. The algorithm calculates the local value of κ for the given iteration, based on formula (6). The maximum of these gives a lower bound for κ.

The results are shown in Table 3.

Table 3. *Increase of κ depending on the size of the matrix*

n	σ	κ
5	0.8	2.70815
10	0.8	347.535
20	0.8	1261800

9. Conclusions

We presented numerical results for the general LCP introduced by Illés, Nagy and Terlaky. During the implementation, a specific method for determining the maximum step length was introduced.

The algorithm worked efficiently on sufficient and monotone LCPs as well. In the case of the matrix introduced by Zsolt Csizmadia we studied the change of the parameter κ depending on the size of the matrix.

Acknowledgement

The authors acknowledge the research support of the Transylvanian Museum Society (EME).

References

[1] Illés T., Nagy M., Terlaky T.: *A polynomial path-following interior point algorithm for general linear complementarity problems*. Journal of Global Optimization 47/3. (2010) 329–342. https://doi.org/10.1007/s10898-008-9348-0

[2] Illés T., Nagy M., Terlaky T.: *EP Theorem for Dual Linear Complementarity Problems*. Journal of Optimization Theory and Applications 140/2. (2009) 233–238. https://doi.org/10.1007/s10957-008-9440-0

[3] Illés T., Nagy M., Terlaky T.: *Polynomial Interior Point Algorithms for General Linear Complementarity Problems*. Algorithmic Operations Research, 5. (2010) 1–12.

[4] Kojima M., Megiddo N., Noma T., Yoshise A.: *A Unified Approach to Interior Point Algorithms for Linear Complementarity Problems*. Lecture Notes in Computer Science, 538, Springer Verlag, Berlin, Germany. (1991)

[5] Darvay Zs., Takó I.: *Computational comparison of primal-dual algorithms based on a new software*. unpublished manuscript. (2012)

[6] Morapitiye S.: *Sufficient Matrices*. (accessed on 12 February 2019). http://math.bme.hu/~sunil/su-matrices/

[7] Darvay Zs., Illés T., Povh J., Rigó P. R.: *Predictor-corrector interior-point algorithm for sufficient linear complementarity problems based on a new search direction*. manuscript. (2019)

[8] Hock W., Shittkowski K.: *Test Examples for Nonlinear Programming Codes*. Lecture Notes in Economics and Mathematical Systems 187. Springer, Berlin (1981) https://doi.org/10.1007/978-3-642-48320-2

[9] de Klerk E., E.-Nagy M.: *On the complexity of computing the handicap of a sufficient matrix*. Mathematical Programmming Serie A 129. (2011) 383–402. https://doi.org/10.1007/s10107-011-0465-z