Bulletin of the *Transilvania* University of Braşov • Vol 13(62), No. 1 - 2020
Series III: Mathematics, Informatics, Physics, 331-342
https://doi.org/10.31926/but.mif.2020.13.62.1.24

THE EFFECT OF THE STEP-SIZE ON THE NUMERICAL BEHAVIOR OF A PRIMAL-DUAL INTERIOR-POINT ALGORITHM APPLIED TO $P_*(\kappa)-$LINEAR COMPLEMENTARY PROBLEM

Chahinez CHENOUF[1] **and Zakia KEBBICHE**[*,2]

Abstract

Theoretically, the step-size plays a crucial role in the complexity analysis of primal-dual interior-point algorithms. In this paper, we would like to focus on the strategy of how to select the step-size. We propose three choices applied to $P_*(\kappa)-$Linear complementarity problem based on two new kernel functions. The numerical behavior of the primal-dual interior-point algorithm is shown to be improved with these step-size choices. We have significantly reduced the number of the inner iterations and the calculation time of the large-update algorithm.

2000 *Mathematics Subject Classification:* 90C05, 90C33.
Key words: $P_*(\kappa)-$Linear complementarity problem; kernel functions; primal-dual interior-point algorithm; step-size; large-update methods; small-update methods.

1 Introduction

After the appearance of the Karmarkar algorithm in 1984 for linear programming, the interior-point methods experienced a real revolution giving rise to a radical change in the theoretical and numerical study of linear programs. The primal-dual path-following methods introduced by Kojima et al. in 1989 [10] and Monteiro et al. [13] are the most attractive and most of the corresponding algorithms are based on the logarithmic barrier function. In practice, finding a strictly feasible initial solution that is close to the central-path is a major difficulty

[1]Faculty of Sciences, Laboratory of Fundamental and Numerical Mathematics, Departement of Mathematics, University Ferhat Abbas Setif 1, Setif, Algeria, e-mail: chahinezchenouf-math@gmail.com

[2*] *Corresponding author*, Faculty of Sciences, Laboratory of Fundamental and Numerical Mathematics, Departement of Mathematics, University Ferhat Abbas Setif 1, Setif, Algeria, e-mail: kebbichez@yahoo.fr

that is the subject of research. To remedy this difficulty, researchers propose the use of the following alternatives: the weighted path-following method, the path-following method via a kernel function and the infeasible path-following method. In this paper we are interested in the path-following method based on kernel functions. These approaches have the advantage of starting the algorithm with any strictly feasible point and improving the algorithmic complexity of large-update interior-point methods.

In 2002, Peng et al. [14] introduced new barrier functions "self-regular functions" for primal-dual interior-point methods applied to linear problems and recently, researchers in [1] and [4] proposed another new class of interior-point methods always for linear programming based on kernel functions that are not logarithmic and not necessarily self-regular. This helps to improve all the theoretical results obtained so far and to provide the best algorithmic complexity, especially for large-update methods.

The success observed in interior-point methods for linear programming has encouraged researchers to develop interesting extensions to solve other classes of optimization problems.

In 1991, Kojima et al. [11] demonstrated the existence of the central path for the linear complementarity problem (LCP), with a $P_*(\kappa) - matrix$ and gave the same results of complexity of the case of linear programming. In 1995, Miao [12] performed an extension of the Mizuno-Todd-Ye method to the case of a $P_*(\kappa) - (LCP)$. In 2005, Illés and Nagy [7] gave another version of the Mizuno-Tood-Ye algorithm and they gave its algorithmic complexity. Afterwards, the results of complexity continue with the works of El Ghami et al. [3] in 2010, El Ghami [5] in 2013, El Ghami et al. [6] in 2017, Kheirfman et al. [9] in 2018, ...

In our work, first we present a primal-dual interior-point (central path) method based on a kernel function with a trigonometric barrier term for $P_*(\kappa) - (LCP)$. It is an extension of a method studied for the first time for the case of a linear program in 2016 [2]. The convergence of the corresponding algorithm is demonstrated and its complexity is given for both small and large-update methods in [6]. Then, we would like to focus on the strategy of how to select the step-size. Our aim is to find out the suitable step-size which guarantees not only the iteration still feasible, but also the decreases of the number of the inner iteration and thus improving the computational behavior of the algorithm. Theoretically, the step-size plays a crucial role in the complexity analysis of primal-dual algorithms but it is very small during each inner iteration, which requires a very large number of iterations and a large calculation time. In practical computation we discover that it may accelerate the iteration process by enlarging the step-size. So, we propose two other choices: dynamic and practical step-sizes. Numerical tests show that algorithms with practical and dynamic step-size are more efficient than these with theoretical one.

The paper is organized as follows: in section 2, we define the linear complementarity problem. In section 3, we briefly review the basic concepts of path-following methods for linear complementarity problem such as the central path and new search directions. We also present the generic algorithm for $P_*(\kappa) - (LCP)$.

The complexity analysis is performed in section 4. Numerical results based on $P_*(\kappa) - (LCP)$ with different choices of the step-size are obtained in section 5. Finally, section 6 contains some concluding remarks and directions for future research.

Some notations used throughout the paper are as follows. First, $\mathbb{R}^n, \mathbb{R}^n_+, \mathbb{R}^n_{++}$ denote the $n-$dimensional Euclidean space with the inner product $\langle ., . \rangle$, the set of nonnegative vectors and the set of positive vectors, respectively. $\|.\|$ denotes the 2-norm for vectors. For x, $s \in \mathbb{R}^n_+$, xs denote the componentwise (or Hadamard) product of the vectors x and s. Furthermore, e denotes the all-one vector of length

n, $X = diag(x)$ the $n \times n$ diagonal matrix with components of vector $x \in \mathbb{R}^n_{++}$ are diagonal entries.

2 Position of the problem

Since its appearance, complementarity has attracted the interest of several researchers and its importance can be measured by the crucial role it plays in the resolution of several problems in different fields: linear programming, convex quadratic programming, variational inequalities, mechanics, ...

The linear complementarity problem noted (LCP), is written in the form:

$$(LCP) \begin{cases} \text{Find } x, s \in \mathbb{R}^n \text{ such that} \\ s = Mx + q \\ x^t s = 0 \\ (x, s) \geq 0, \end{cases} \qquad (1)$$

where $M \in \mathbb{R}^{n \times n}$ is a square matrix of order n and $q \in \mathbb{R}^n$.

Existing methods to solve (LCP) are extensions of methods designed for linear programming.

3 Path-following (Central path) method

This approach is proposed by Kojima, Mizuno and Yoshise in 1991, it is an

extension of the path-following method applied to linear programming.

We consider the (LCP) defined by (1) and we noted by:

$S = \left\{(x, s) \in \mathbb{R}^n_+ \times \mathbb{R}^n_+ : s = Mx + q\right\}$ the set of feasible solutions of (LCP).

$S_{int} = \left\{(x, s) \in \mathbb{R}^n_{++} \times \mathbb{R}^n_{++} : s = Mx + q\right\}$ the set of strictly feasible solutions of (LCP).

Assumptions: Without loss of generality, it is assumed that there is $(x^0, s^0) > 0$, such that $s^0 = Mx^0 + q$ with M is $P_*(\kappa) - matrix$.

Definition 1. *: The class $P_*(\kappa)$ is introduced by Kojima et al. [11] in 1991.*

Let Y be an open convex subset of $\mathbb{R}^n$ and $\kappa \geq 0$, we say that M is $P_*(\kappa) -$ matrix on Y if and only if $(1+4\kappa) \sum_{i \in J_+(x)} x_i (Mx)_i + \sum_{i \in J_-(x)} x_i \,(Mx)_i \geq 0,$

for all $x \in Y$, and $J_+(x) = \{i \in J : x_i\ (Mx)_i \geq 0\}$ and $J_-(x) = \{i \in J : x_i\ (Mx)_i < 0\}$, in addition, M is called a P_*- matrix if it is $P_*(\kappa) -$ matrix for some $\kappa \geq 0$. Note that the class P_* is the union of all $P_*(\kappa) -$ matrices for $\kappa \geq 0$ and contains the class of positive semidefinite matrices by choosing $\kappa = 0$.

Proposition 1. *[11]: If M is a $P_* -$ matrix, then*
$M\ ' = \begin{pmatrix} -M & I \\ S & X \end{pmatrix}$ is a nonsingular matrix for all diagonal and positive matrices X and S in $\mathbb{R}^{n\times n}$.

Corollary 1. *[6]: Let M a $P_* -$ matrix and $x, s \in R^n_{++}$. Then for all $a \in R^n$ the system*

$$\begin{cases} -M \triangle x + \triangle s = 0 \\ S \triangle x + X \triangle s = a \end{cases} \qquad (2)$$

has a unique solution $(\triangle x, \triangle s)$.

3.1 Introduction of new directions

The basic idea of these methods is to replace the 2 nd equation in (1) with the nonlinear equation $sx = \mu e$ for $\mu > 0$.

So we get the following parametric system

$$\begin{cases} s = Mx + q, \\ sx = \mu e, \\ (x, s) \geq 0. \end{cases} \qquad (3)$$

Since M is a $P_*(\kappa) - matrix$ and (1) is strictly feasible, then system (3) has a unique solution $(x(\mu), s(\mu))$ for each $\mu > 0$.

$(x(\mu), s(\mu))$ is called $\mu-$center of (3), the set of $\mu-$centers ($\mu > 0$) defines the central path of (3). If $\mu \to 0$, the limit of the central path exists and satisfies the condition of complementarity and belongs to the set of solutions of (1), [11].

Let (x, s) be a strictly feasible point and $\mu > 0$. We define the vector

$$\upsilon = \sqrt{\tfrac{xs}{\mu}} \qquad (4)$$

Note that the pair (x, s) coincides with the $\mu-$center $(x(\mu), s(\mu))$ if and only if $\upsilon = e$.

The search directions $\triangle x, \triangle s$ are given by the system

$$\begin{cases} -M \triangle x + \triangle s = 0 \\ s \triangle x + x \triangle s = -\mu \upsilon \bigtriangledown \Phi(\upsilon), \end{cases} \qquad (5)$$

where $\Phi(\upsilon)$ is a regular strictly convex function defined for $\upsilon > 0$ with $\Phi(e) = 0$ and minimal in $\upsilon = e$.

Since M is a $P_* - matrix$, system (5) defines $(\triangle x\ , \triangle s)$ in a unique way for all $x > 0$ and $s > 0$.

We define also the vector p by

$$p = \sqrt{\tfrac{x}{s}}, \qquad (6)$$

and we note by

$$\overline{M} = PMP \ , P = diag(p) \text{ and } V = diag(\upsilon) \quad \text{with} \quad \upsilon = \sqrt{\tfrac{xs}{\mu}},$$
$$d_x = \tfrac{\upsilon \triangle x}{x} \quad , \qquad d_s = \tfrac{\upsilon \triangle s}{s}. \qquad (7)$$

System (5) can be reformulated as follows

$$\begin{cases} -\overline{M}d_x + d_s = 0 \\ d_x + d_s = -\nabla\Phi(\upsilon). \end{cases} \qquad (8)$$

From the solution d_x and d_s, the vectors $\triangle x$ and $\triangle s$ can be calculated using (7).

3.2 The generic primal-dual interior-point algorithm

```
Data
A proximity function $\Phi(\upsilon)$, a threshold parameter $\tau > 0$, a parameter of precision
$\varepsilon > 0$ and an update parameter $\theta$, $0 < \theta < 1$.
Initialization: $k = 0; \mu = \mu^0; x = x^0; s = s^0; \upsilon^0 = \sqrt{\frac{x^0 s^0}{\mu^0}}$.
**While** $n\mu^k > \varepsilon$ do
Start (outer iteration)
$\mu^{k+1} = (1-\theta)\mu^k$ ;
While $\Phi(\upsilon) > \tau$ **do**
Start ((inner iteration)
- Solve system (8) to determine $(d_x, d_s)$ then $(\triangle x, \triangle s)$ ;
- Calculate the step-size $\alpha$ and let $x = x + \alpha \triangle x$, $s = s + \alpha \triangle s$ , $\upsilon = \sqrt{\frac{xs}{\mu}}$;
end (inner iteration),
end (outer iteration).
**End.**
```

4 Complexity of the algorithm

Small-update methods: If we take $\theta = O(\frac{1}{\sqrt{n}})$ as a reduction factor of μ and

$\tau = O(1)$, first, we ensure that we stay in a neighborhood of the central path and, on the other hand, we get an algorithmic complexity of order $O(\sqrt{n}\log\frac{n}{\varepsilon})$ for linear optimization. It is the best complexity obtained today. Algorithms that use a reduction θ depending on n are called small-update algorithms.

Large-update methods: By using $\tau = O(n)$ and $\theta = O(1)$ we obtain a better numerical efficiency in practice. The theoretical complexity is however less good, it is of order $O(n \log \frac{n}{\varepsilon})$ for linear optimization. Algorithms that use a reduction θ independent of n are called large-update algorithms.

The parameters τ and θ and the step-size α must be chosen in such a way that the number of iterations is as small as possible. This number depends on the choice of function Φ " the proximity function". Recently, researchers are trying to improve the number of iterations and thus improving the algorithmic complexity, based on new choices of the kernel function Ψ.

In the following table, we give some examples of kernel functions with complexity results that correspond to large-update methods. For small-update methods, the complexity results are the same for all functions, namely $O((1+2\kappa)\sqrt{n} \log \frac{n}{\varepsilon})$ which is the best complexity until today for this type of methods [6].

i	Kernel functions $\Psi_i(t)$	Large-update methods
1	$\frac{t^2-1}{2} - \log t$	$O((1+2\kappa)n \log \frac{n}{\varepsilon})$
2	$\frac{t^2-1}{2} + \frac{t^{1-q}-1}{q(q-1)} - \frac{q-1}{q}(t-1)$	$O((1+2\kappa)qn^{\frac{q+1}{2q}} \log \frac{n}{\varepsilon})$
3	$\frac{1}{2}(t-\frac{1}{t})^2$	$O((1+2\kappa)n^{\frac{2}{3}} \log \frac{n}{\varepsilon})$
4	$\frac{t^2-1}{2} + e^{\frac{1}{t}-1} - 1$	$O((1+2\kappa)\sqrt{n} \log^2 n \log \frac{n}{\varepsilon})$
5	$\frac{t^2-1}{2} - \int_1^t e^{\frac{1}{\xi}-1} d\xi$	$O((1+2\kappa)\sqrt{n} \log^2 n \log \frac{n}{\varepsilon})$
6	$\frac{t^2-1}{2} + \frac{t^{1-q}-1}{(q-1)}, q > 1$	$O((1+2\kappa)qn^{\frac{q+1}{2q}} \log \frac{n}{\varepsilon})$
7	$t - 1 + \frac{t^{1-q}-1}{(q-1)}, q > 1$	$O((1+2\kappa)nq \log \frac{n}{\varepsilon})$
8	$\frac{t^2-1}{2} + \frac{6}{\pi} \tan(\pi \frac{1-t}{4t+2})$	$O((1+2\kappa)n^{\frac{3}{4}} \log \frac{n}{\varepsilon})$
9	$\frac{t^2-1}{2} + \frac{4}{p\pi}(\tan^p(\frac{\pi}{2t+2}) - 1)$, $p \geq 2$	$O(p(1+2\kappa)n^{\frac{2+p}{2(1+p)}} \log \frac{n}{\varepsilon}$

5 Numerical results

To prove the efficiency of a new kernel function and evaluate its influence on the computational behavior of an algorithm, it is necessary to perform some numerical tests.

In this part, we consider a kernel function studied recently in 2017 defined in [6] by: $\Psi_1(t) = (\frac{t^2-1}{2} + \frac{4}{\pi p}[\tan^p(\frac{\pi}{2t+2}) - 1], p \geq 2)$.

To solve the problem numerically, we used the software MATLAB. We have taken $\varepsilon = 10^{-6}$, $\theta = 0,99$, $p = 2$ and $\tau = 10$ as examples of fixed size and $\tau = n$ as examples with a variable size corresponding to large-update methods.

Step-size α is chosen in two different ways:

1 /The theoretical choice: We take the default step-size defined in [6] by: $\alpha = \frac{1}{(1+2\kappa)(9+4\pi p)(8\delta+2)^{\frac{p+2}{p+1}}} = \frac{1}{(1+2\kappa)(9+8\pi)(8\delta+2)^{\frac{4}{3}}}$.

2/The dynamic choice: Due to the fact that the theoretical value of the step-size α is very small in each iteration, we choose a dynamic step proposed in [16] in order to enlarge the step-size and improve the computational behavior of the algorithm, so we put:$\alpha = \begin{cases} 2\widetilde{\alpha} \text{ if } \|\triangle x\| \geq n \\ 5\widetilde{\alpha} \text{ if } 1 \leq \ \|\triangle x\| < n \\ 10\widetilde{\alpha} \text{ if } \|\triangle x\| < 1, \end{cases}$

where n is the size of the problem and $\widetilde{\alpha}$ is the theoretical value of the step-size.

5.1 Examples of fixed size

We consider the following $p_*(0) - LCP$:

Example 1. $M = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}$ *and* $q = \begin{pmatrix} -4 \\ -5 \\ -1 \end{pmatrix}$.

The starting point is:

$x^0 = \begin{pmatrix} 0.9918 & 2.0082 & 0.0475 \end{pmatrix}^t$ *and* $s^0 = \begin{pmatrix} 0.0395 & 0.0558 & 2.0952 \end{pmatrix}^t$.

The solution is: $x^* = \begin{pmatrix} 1 & 2 & 0 \end{pmatrix}^t$.

Example 2. $M = \begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 0 & 1 \\ 1 & 0 & 1 & 2 \\ -1 & -1 & -2 & 0 \end{pmatrix}$ *and* $q = \begin{pmatrix} -8 \\ -6 \\ -4 \\ 3 \end{pmatrix}$.

The starting point is:

$x^0 = \begin{pmatrix} 2.4742 & 0.4992 & 0.0073 & 2.5639 \end{pmatrix}^t$

$s^0 = \begin{pmatrix} 0.0187 & 0.0365 & 3.6093 & 0.0120 \end{pmatrix}^t$.

The solution is: $x^* = \begin{pmatrix} 2.5 & 0.5 & 0 & 2.5 \end{pmatrix}^t$.

Example 3. $M = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 \end{pmatrix}$ *and* $q = \begin{pmatrix} -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \end{pmatrix}$

The starting point is:

$x^0 = \begin{pmatrix} 3.6563 & 6.2520 & 7.8097 & 8.3347 & 7.8253 & 6.2743 & 3.6722 \end{pmatrix}^t$,

$s^0 = \begin{pmatrix} 0.0597 & 0.0400 & 0.0321 & 0.0347 & 0.0415 & 0.0512 & 0.0703 \end{pmatrix}^t$.

The solution is: $x^* = \begin{pmatrix} 3.5 & 6 & 7.5 & 8 & 7.5 & 6 & 3.5 \end{pmatrix}^t$.

Example 4. *We consider the following matrix*

$$M = \begin{pmatrix} 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \text{ and } q = \begin{pmatrix} -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \end{pmatrix}.$$

The starting point is:

$$(x^0)^t = \begin{pmatrix} 0.0009 & 0.0009 & 0.0009 & 0.0009 & 0.0009 & 0.0009 & 0.0009 & 0.0009 \\ & 0.0009 & 0.0009 & 0.0009 & 0.0009 & 0.0009 & 0.0009 & 1.0009 \end{pmatrix},$$

$$(s^0)^t = \begin{pmatrix} 1.0261 & 1.0243 & 1.0225 & 1.0198 & 1.0189 & 1.0171 & 1.0153 & 1.0135 \\ & 1.0108 & 1.0099 & 1.0081 & 1.0063 & 1.0045 & 1.0027 & 0.0009 \end{pmatrix}.$$

The solution is:

$$x^* = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}^t.$$

In the tables of results, n represents the size of the example, (Outer) represents the number of outer iterations, (Inner) represents the number of inner iterations and (T(s)) represents the calculation time in seconds. We summarize our results in the following table:

	Theoretical choice			Dynamic choice		
Examples	*Outer*	*Inner*	$T(s)$	*Outer*	*Inner*	$T(s)$
Example 1	5	58375	42.891960	5	6028	6.299861
Example 2	5	25459	36.219801	5	2542	4.553663
Example 3	5	39594	47.917731	5	3959	5.763560
Example 4	5	61277	114.108174	5	6123	14.543124

Table1: Choice of the step-size for kernel function 1 with fixed size.

5.2 Examples of variable size

Example 5. *We consider the following $p_*(0) - LCP$ with*

$$m_{ij} = \begin{cases} 4 \text{ if } i = j \\ -1 \text{ if } |i-j| = 1 \\ 0 \text{ if not} \end{cases} \quad \text{and } q = \begin{pmatrix} -1 \\ . \\ . \\ -1 \end{pmatrix} \text{ for } \begin{cases} 1 \le i \le n. \\ 1 \le j \le n. \end{cases}$$

	Theoretical choice			Dynamic choice		
n	*Outer*	*Inner*	$T(s)$	*Outer*	*Inner*	$T(s)$
7	5	39558	44.563197	5	3841	5.466096
15	5	61272	135.943155	5	6118	13.859415
25	5	83043	338.890412	5	8301	30.255119
50	5	127377	1226.104891	5	12720	102.328240
75	$--$	$--$	$--$	5	18166	318.978979

Table 2 : Choice of the step-size for kernel function 1 with the variable size.

5.3 Example of a nonmonotonic (LCP)

Example 6. *We consider a $P_*(\kappa)-$ LCP, with*

$$M = \begin{pmatrix} 0 & 1+4\kappa & 0 \\ -1 & 0 & 0 \\ 0 & 0 & c \end{pmatrix} \text{ and } q = \begin{pmatrix} 0.01 \\ 0.501 \\ -0.49 \end{pmatrix} \text{ with } c \geq 0 \text{ and } \kappa \geq 0.$$

We have the table:

	Theoretical choice			Dynamic choice		
κ	*Outer*	*Inner*	$T(s)$	*Outer*	*Inner*	$T(s)$
0	5	62837	51.851148	5	6278	4.372369
0.2	5	84248	59.693772	5	8407	5.834019
0.25	5	89093	62.967999	5	8909	6.427401
0.9	5	113218	88.492977	5	11262	8.627475

Table 3 : Choice of the step-size for the kernel function 1 with differents values of κ .

5.4 A practical choice of the step size

It is a less expensive procedure than the linear search introduced in [8]. We have the conditions:

$$\begin{cases} x + \alpha_x^+ dx > 0 \\ s + \alpha_s^+ ds > 0 \end{cases}$$

Which give: $\alpha_x^+ = \beta\alpha_x$ and $\alpha_s^+ = \beta\alpha_s$ such as $0 < \beta < 1$, or

$$\alpha_x = \begin{cases} \min\limits_{i\in I}(-\frac{x_i}{dx_i}) \text{ with } I = \{i : dx_i < 0\} \\ 1 \qquad \text{elsewhere.} \end{cases}$$

$$\alpha_s = \begin{cases} \min\limits_{i\in I}(-\frac{s_i}{ds_i}) \text{ with } I = \{i : ds_i < 0\} \\ 1 \qquad \text{elsewhere.} \end{cases}$$

We take $\alpha_k = \min(\alpha_x^+, \alpha_s^+)$. So the new iterate is: $(x^+, s^+) = (x, s) + \alpha_k(dx, ds)$.

For this choice, we consider two kernel functions defined by:

- **Kernel function 1** [6], (2017) : $\Psi_1(t) = (\frac{t^2-1}{2} + \frac{4}{\pi p}[\tan^p(\frac{\pi}{2t+2}) - 1], p \geq 2)$.
- **Kernel function 2** [9], (2018)**:** $\Psi_2(t) = \frac{t^2-1}{2} + \frac{4}{\pi}\cot(\frac{\pi t}{1+t})$.

Applying this procedure, we obtain the results summarized in the following tables:

	Kernel function 1			Kernel function 2		
Examples	$Outer$	$Inner$	$T(s)$	$Outer$	$Inner$	$T(s)$
Example 1	5	8	0.058775	5	10	0.058207
Example 2	5	6	0.049446	5	72	0.108315
Example 3	5	8	0.056408	5	25	0.083541
Example 4	5	8	0.055684	5	108	0.194179

Table 4: A practical choice of the step-size α for examples with fixed size.

	Kernel function 1			Kernel function 2		
n	$Outer$	$Inner$	$T(s)$	$Outer$	$Inner$	$T(s)$
7	5	8	0.109394	5	107	0.051988
25	5	8	0.432846	5	59	0.633441
50	5	8	1.422137	5	77	0.940066
150	6	8	13.619962	6	629	26.217568
250	6	8	68.230574	6	117	30.151152

Table 5: A practical choice of the step-size α for examples with variable size.

Kernel function 1				Kernel function 2		
κ	$Outer$	$Inner$	$T(s)$	$Outer$	$Inner$	$T(s)$
0	5	11	0.040181	5	6	0.062437
1/5	5	10	0.055514	5	11	0.076077
1/4	5	10	0.061765	5	11	0.077246
0.9	5	6	0.059457	5	11	0.044084

Table 6: A practical choice of the step-size α with a $P_*(\kappa) - (LCP)$.

6 Conclusion

In this paper, we have presented a primal-dual interior-point algorithm for $P_*(\kappa) - LCP$ based on two recent kernel functions and we have given numerical results for this algorithm with different choices of the step-size. We have reached the following conclusions:

- Kernel functions play a very important role in the analysis of interior-point algorithms since they improve their complexities.

- The complexity of the primal-dual interior-point methods for linear programming based on the logarithmic barrier function is of order $O(n \log \frac{n}{\epsilon})$ for the large-update algorithms, whereas, with these new features, the complexity is getting better at $O(\sqrt{n} \log n \log \frac{n}{\epsilon})$. However, the complexity of small-update algorithm is always of order $O(\sqrt{n} \log \frac{n}{\epsilon})$.

- The best result of complexity for the large-update and small-update methods applied to $P_*(\kappa) - LCP$, can be achieved, namely: $O((1+2\kappa)\sqrt{n} \log n \log \frac{n}{\varepsilon})$ for large-update methods and $O((1+2\kappa)\sqrt{n} \log \frac{n}{\varepsilon})$ for small-update methods.

- For $k = 0$, we find the same results of complexity obtained in the case of linear programming.

- In practice, the theoretical value of step-size α is very small in each iteration, which requires a very large number of iterations and calculation time.

- Algorithm with dynamic step-size work faster than the algorithm with the theoretical one during the inner iterations.

- The practical choice has significantly reduced the number of iterations and the calculation time of the algorithm than the dynamic choice.

In our future study,

- We intend to generalize these results to other classes of optimization problems.

- Finding new kernel functions will be of a great importance.

- It is necessary to insist on the analytic structure of the kernel function in order to reduce the number of inner iterations and thus improve the algorithmic complexity.

References

[1] Bai, Y.Q., El Ghami, M. and Roos, C., *A comparative study of kernel functions for primal-dual interior-point algorithms in linear optimization*, SIAM Journal on Optimization **15** (2004), no. 1, 101–128.

[2] Bouafia, M, Benterki, D. and Yassine, A. *An efficient primal-dual interior point method for linear programing problems based on a new kernel function with a trigonometric barrier term*, J. of Optimization Theory and Applications **170** (2016), no. 2, 528–545; doi: 10.1007/s10957-016-0895-0.

[3] El Ghami, M. and Steihaug, T., *Kernel-function based primal-dual algorithms for $P_*(\kappa)-$ linear complementarity problems*, International J. RAIRO-Operations Research **44** (2010), no. 3, 185–205.

[4] El Ghami, M., Guennoun, Z. A., Bouali S. and Steihaug, T., *Primal-dual interior-point methods for linear optimization based on a kernel function with trigonometric barrier term*, J. of Computational and Applied Mathematics, **236** (2012), no. 15, 3613–3623.

[5] El Ghami, M., *Primal-dual algorithms for $P_*(\kappa)-$ linear complementarity problems based on kernel function with trigonometric barrier term*, Optimization Theory, Decision Making, and Operations Research Applications **2013** (2013), 331–349.

[6] El Ghami, M. and Wang, G.Q., *Interior-point methods for $P_*(\kappa)-$ linear complementarity problem based on generalized trigonometric barrier function*, Shanghai University of Engineering Science Shanghai, 201620, P.R. China, (2017).

[7] Illés, T. and Nagy, M., *The Mizuno-Todd-Ye predictor-corrector algorithm for sufficient matrix linear complementarity problem*, Alkalmaz. Mat. Lapok **22** (2005), no. 1, 41–61.

[8] Keraghel, A., *Etude adaptative et comparative des principales variantes dans l'algorithme de Karmarkar*, Thèse de Doctorat, Université Joseph Fourier –Grenoble I, France, 1989.

[9] Kheirfam, B. and Haghighi M., *An infeasible interior-point method for the P-matrix linear complementarity problem based on a trigonometric kernel function with full-Newton step*, Communication in Combinatorics and Optimization, **3** (2018), no. 1, 51-70.

[10] Kojima, M., Megiddo, N., Noma, T. and Yoshise, A., *A primal-dual interior point algorithm for linear programming*, In: N. Megiddo (Ed), Progress in Mathematical programming, Interior-Point Related Methods, **10** Springer Verlag, New York, 1989, 29–47.

[11] Kojima, M., Megiddo, N., Noma, T. and Yoshise, A., *A unified approach to interior point algorithms for linear complementarity problems*, Volume **538** of Lecture Notes in Computer Science, Springer Verlag, Berlin, 1991.

[12] Miao, J., *A quadratically convergent $O((1+\kappa)\sqrt{n}$ l-iteration algorithm for the $P^*(\kappa)-$matrix linear complementarity problem*, Mathematical Programming, **69** (1995), 355–368.

[13] Monteiro, R.D.C. and Adler, I., *Interior path-following primal-dual algorithms. Part I: Linear programming*, Mathematical Programming, **44** (1989), 27–41.

[14] Peng, J. Roos, C. and Terlaky, T., *Self-regularity : a new paradigm for primal-dual interior-point algorithms*, Princeton University Press, Princeton, NJ, 2002.

[15] Peng, J., Roos, C. and Terlaky, T., *Self-regular functions and new search directions for linear and semidefinite optimization*, Mathematical Programming, **93** (2002), 129–171.

[16] Qian, Z.G. and Bai, Y.Q., *Primal-dual interior-point algorithms with dynamic step-size based on kernel fonctions for linear programming*, Journal of Shanghai University **9** (2005), no. 5, 391-396.