J. Nonlinear Funct. Anal. 2018 (2018), Article ID 14 https://doi.org/10.23952/jnfa.2018.14

MATHRES Journal of Nonlinear Functional Analysis
Available online at http://jnfa.mathres.org

AN INTERIOR-POINT ALGORITHM FOR SEMIDEFINITE OPTIMIZATION BASED ON A NEW PARAMETRIC KERNEL FUNCTION

SAJAD FATHI-HAFSHEJANI[1,*], ALIREZA FAKHARZADEH J.[1,2]

[1]Department of Mathematics, Shiraz University of Technology, P.O. Box 71555-313, Shiraz, Iran
[2]Fars Elites Foundation, P.O. Box 71966-98893, Shiraz, Iran

Abstract. In this paper, an interior-point algorithm for Semidefinite Optimization (SDO) problems based on a new parametric kernel function is proposed. By means of some simple analysis tools, we prove that the primal-dual interior-point algorithm for solving SDO problems meets $O\left(\sqrt{n}\log n\log\frac{n}{\varepsilon}\right)$, iteration complexity bound for large-update methods. Numerical results confirm that our new proposed kernel function is doing well in practice in comparison with some existing kernel functions in the literature.

Keywords. Kernel function; Semidefinite optimization; Primal-dual interior-point method; Large-update method.

2010 Mathematics Subject Classification. 90C51, 90C22, 65K05

1. INTRODUCTION

Semidefinite Optimization (SDO) problems are convex optimization problems over the intersection of an affine set and the cone of positive semidefinite matrices. Nowadays, these problems have been changed as one of the most active research areas in mathematical programming due to their wide applications in the real world problems [1]. The polynomial time Interior-Point Methods (IPMs), proposed by Karmarkar [2] in 1984, are the most important class of algorithms that are extended directly from Linear Optimization (LO) problems to SDO problems and are widely used for solving these problems. Roughly speaking, all the IPMs designed for LO have been successfully extended to SDO. The first work in this area was proposed by Nesterov and Nemirovskii [3] and they built up a polynomial complexity of the interior-point algorithms, at least in the theoretical sense, for solving conic problems, including SDO and Second Order Cone Programming problems, using the so called self-concordant barrier functions which consist of the logarithmic barrier function. They showed that the primal-dual IPMs for LO preserves its theoretical behavior when the nonnegativity constrains in LO are replaced by a convex homogeneous and self-dual cone. Later, Nesterov and Todd provided the same terminology for the case when the cone is self-scaled [4].

*Corresponding author.
E-mail addresses: s.fathi@sutech.ac.ir (S. Fathi-Hafshejani), a_fakharzadeh@sutech.ac.ir (A. Fakharzadeh J.).
Received January 6, 2018; Accepted March 26, 2018.

A new primal-dual interior-point algorithm for solving LO problems based on the Self-Regular (SR)-barrier functions was first suggested by Peng, Roos and Terlaky [5]. They obtained the worst case iteration complexity bounds for large and small-update methods as $O\left(\sqrt{n}\log n\log\frac{n}{\varepsilon}\right)$ and $O\left(\sqrt{n}\log\frac{n}{\varepsilon}\right)$, respectively. The same bound as LO were also achieved for SDO problems using the self-regular kernel function [5]. In recent years, many attempts for introducing non SR-kernel functions in order to at least meeting the complexity results of SR-barrier functions have been started, see, e.g., [6, 7, 8, 9] and the references therein. A comparative study on the kernel functions for LO was provided in [10].

Recently, El Ghami *et al.* in [11] introduced a trigonometric kernel function for the first time and analyzed the feasible primal-dual IPMs for LO based on this kernel function. They showed that the primal-dual IPM for solving LO problems enjoys $O\left(n^{\frac{3}{4}}\log\frac{n}{\varepsilon}\right)$ as the worst case iteration complexity. Inspired by this work, Kheirfam in [12] proposed a new kernel function with trigonometric barrier term and analyzed the complexity of large-update primal-dual IPMs for SDO problems based on this kernel function. He achieved the same complexity of LO case in [11] for SDO problems. Recently, El Ghami in [13] proposed a primal-dual IPMs for $P_*(\kappa)$-linear complementarity problems, for $\kappa\geq 0$, based on a kernel function with trigonometric barrier term which was previously proposed for LO in [11]. He achieved the worst case iteration complexity as $O\left((1+2\kappa)n^{\frac{3}{4}}\log\frac{n}{\varepsilon}\right)$ for large-update. In the more recent papers, several trigonometric kernel functions have been suggested [14, 15, 16, 17, 18, 19]. Based on the some of them, the best known worst case complexity results are obtained [14, 16, 17, 19]. For some other related IPMs based on the kernel functions works, we refer to [20, 21, 22, 23, 24].

In this paper, a primal-dual IPM for solving SDO problems based on the new kernel function is presented. By means of some simple analysis tools, we prove that for SDO problems, the worst case iteration complexity of large-update primal-dual IPMs based on the new proposed kernel function enjoys $O\left(\sqrt{n}\log n\log\frac{n}{\varepsilon}\right)$, which matches in order the so far best-known iteration bound for large update IPMs. Moreover, the result is also competitive with some existing results in the literature. Numerical results show that the new proposed kernel function is well promising and performs very well in practice too.

The paper is organized as follows. In Section 2, some elementary concepts of matrix functions, IPMs and the central path for SDO are briefly recalled. In Section 3, we introduce the new kernel function and present some properties of this function. Section 4 is devoted to discuss about the growth behavior of the proximity measure during an inner iteration. An estimation of the step size is provided in Section 5. We also introduce a default value for the step size in this section. The worst case iteration complexity for the primal-dual IPMs based on the new kernel function is given in Section 6. We illustrate the practical performance of the new proposed kernel function in Section 7. Finally, we end up the paper with some concluding remarks in Section 8.

The following notations are used throughout the paper: $\mathbb{R}^n_+$ and $\mathbb{R}^n_{++}$ denote the nonnegative and positive orthants, respectively. $\|.\|$ is used as the Frobenius norm for the matrices and as the Euclidian norm for the vectors. $\mathbb{R}^{m\times n}$ is the space of all $m\times n$ matrices whose components are from $\mathbb{R}$. $\mathbb{S}^n$, $\mathbb{S}^n_+$ and $\mathbb{S}^n_{++}$ denote the cone of symmetric, symmetric positive semidefinite and symmetric positive definite matrices of the order $n\times n$, respectively. E denotes the identity matrix. We use the classical Löwner partial ordering $\succeq$ for symmetric matrices by the mean of $A\succeq B(A\succ B)$ if and only if $A-B$ is positive semidefinite (positive definite). Given $A\in\mathbb{R}^{n\times n}$, $Tr(A)$ stands for the trace of the matrix A. For $A,B\in\mathbb{R}^{m\times n}$, the inner product is defined by $A\bullet B=Tr(AB^T)$. For any symmetric positive definite matrix $Q\in\mathbb{S}^n_{++}$, the expression $Q^{\frac{1}{2}}$ stands for the symmetric square root of Q. For a vector $x\in\mathbb{R}^n$, $\mathtt{diag}(x)$ is a

diagonal matrix whose diagonal entries are the components of the vector x. For $V \in \mathbb{S}^n_{++}$, $\lambda(V)$ denotes the vector of eigenvalues of V arranged in non-increasing order, i.e., $\lambda_1(V) \geq \lambda_2(V) \geq \ldots \geq \lambda_n(V)$.

2. Preliminaries

In this section, we first introduce some useful results regarding the properties of the symmetric matrices. Then some concepts of central path for SDO problems are briefly recalled. The structure of generic primal-dual IPMs based on kernel function is also given in this section.

Now, we recall some well known facts from linear algebra which are essential in our analysis. Consider the SDO problem as:

$$\begin{array}{lll} (P) & \min & C \bullet X \\ & s.t. & A_i \bullet X = b_i, \qquad i = 1, ..., m, \\ & & X \succeq 0, \end{array}$$

along with its dual problem as

$$\begin{array}{lll} (D) & \max & b^T y \\ & s.t. & \sum_{i=1}^{m} y_i A_i + S = C, \\ & & S \succeq 0, \end{array}$$

where $C, A_i \in \mathbb{S}^n$, for $1 \leq i \leq m$ and $b, y \in \mathbb{R}^m$. Furthermore, $X \succeq 0$ (or $X \succ 0$) means that $X \in \mathbb{S}^n_+$ (or $X \in \mathbb{S}^n_{++}$). Throughout the paper, we assume that the matrices A_i are linearly independent. We also assume that both problems (P) and (D) satisfy the interior-point Condition (IPC), that is, there exist $X^0 \succ 0$ and (y^0, S^0) with $S^0 \succ 0$ in the feasible regions of (P) and (D), respectively, i.e.,

$$\begin{aligned} A_i \bullet X^0 &= b_i, \qquad i = 1, ..., m, \\ \sum_{i=1}^{m} y_i^0 A_i + S^0 &= C. \end{aligned}$$

This can be achieved via the so-called self-dual embedding technique introduced by De Klerk in [1]. The following theorem is known as the spectral theorem for symmetric matrices in the literature. One can find a proof of it in [25].

Theorem 2.1. *The real $n \times n$ matrix A is symmetric if and only if there exists an orthogonal matrix $U \in \mathbb{R}^{n \times n}$, i.e., $U^T U = E$, so that $U^T A U = \Lambda$, where Λ is a diagonal matrix whose diagonal entries are the eigenvalues of the matrix A.*

Now, we define the concepts of the matrix function and the real valued matrix function.

Definition 2.2. Let $V \in \mathbb{S}^n_{++}$ and $Q \in \mathbb{R}^{n \times n}$ be an orthogonal matrix that diagonalizes V, i.e. $V = Q^T diag(\lambda(V))Q$. Then, for given real function $\psi(t)$, $t \in \mathbb{R}_+$, the matrix function is defined by

$$\psi(V) = Q^T diag(\psi(\lambda_1(V)), \psi(\lambda_2(V)), \ldots, \psi(\lambda_n(V)))Q. \tag{2.1}$$

Furthermore, the real valued matrix function $\Psi(V) : \mathbb{S}^n_{++} \to \mathbb{R}_+$ induced by the matrix function $\psi(V)$ is defined by:

$$\Psi(V) := Tr(\psi(V)) = \sum_{i=1}^{n} \psi(\lambda_i(V)). \tag{2.2}$$

Suppose that $\psi(t)$ is a twice differentiable function for all $t > 0$. So, the first and second order derivatives of the matrix function $\psi(V)$, are defined as below [5]:

$$\begin{aligned}\psi'(V) &= Q^T diag(\psi'(\lambda_1(V)), \psi'(\lambda_2(V)), \ldots, \psi'(\lambda_n(V)))Q,\\ \psi''(V) &= Q^T diag(\psi''(\lambda_1(V)), \psi''(\lambda_2(V)), \ldots, \psi''(\lambda_n(V)))Q.\end{aligned}$$

Definition 2.3. A matric $M(t)$ is said to be a matrix of functions if each entry of $M(t)$ is a function of t, i.e., $M(t) = [M_{ij}(t)]$.

Remark 2.4. In the following, when we apply the function $\psi(.)$ and its derivatives $\psi'(.)$ and $\psi''(.)$, indeed, we face with matrix functions if the argument is a matrix and with a univariate function if the argument is in $\mathbb{R}_{++}$.

In the next lemma we describe some law about the first derivative of matrix of functions. One can find the proof in [26, 27].

Lemma 2.5. *Suppose that $M(t)$ and $N(t)$ are two matrices of functions, thus, one has:*

$$\frac{d}{dt}Tr(M(t)) = Tr\left(\frac{d}{dt}M(t)\right) = Tr(M'(t)), \tag{2.3}$$

$$\frac{d}{dt}Tr(\psi(M(t))) = Tr\left(\psi'(M(t))M'(t)\right), \tag{2.4}$$

$$\frac{d}{dt}(M(t)N(t)) = \left(\frac{d}{dt}M(t)\right)N(t) + M(t)\left(\frac{d}{dt}N(t)\right) = M'(t)N(t) + M(t)N'(t). \tag{2.5}$$

In the sequel, we provide the central path concept for the SDO problems which is a direct extension of those in LO. Let IPC hold for the SDO problem. The Karush-Kuhn-Tucker (KKT) optimality conditions for both problems (P) and (D) can be expressed as follows:

$$\begin{aligned} A_i \bullet X &= b_i, \qquad i = 1, \ldots, m,\\ \sum_{i=1}^{m} y_i A_i + S &= C,\\ XS &= 0,\\ X \succeq 0, &\qquad S \succeq 0. \end{aligned} \tag{2.6}$$

The key idea of primal-dual IPMs for SDO problems is to replace the last equation in system (2.6) which is so called complementarity condition by the parameterized equation $XS = \mu E$, where $\mu > 0$. If IPC holds, thus the new parameterized system, that is,

$$\begin{aligned} A_i \bullet X &= b_i, \qquad i = 1, \ldots, m,\\ \sum_{i=1}^{m} y_i A_i + S &= C,\\ XS &= \mu E,\\ X \succ 0, &\qquad S \succ 0 \end{aligned} \tag{2.7}$$

has a unique solution for given $\mu > 0$ and it is denoted by $(X(\mu), y(\mu), S(\mu))$. We call $X(\mu)$ the μ-center of (P) and $(y(\mu), S(\mu))$ is known as the μ- center of (D). The set of all solution of system (2.7), that is, $\mathscr{C} := \{(X(\mu), y(\mu), S(\mu)) |\ \mu > 0\}$ defines a homotopy path, which is so called central path of both

problems (P) and (D). The concept of central path for SDO problems was first proposed by Nesterov and Nemirovskii [3]. It should be note that, when the parameter μ goes to zero, the limit of the central path exists. This limit denotes the optimal solution set for the problem.

A direct application of the Newton method to system (2.7) generate the following linear system of equations for the search direction $(\Delta X, \Delta y, \Delta S)$:

$$\begin{aligned} A_i \bullet \Delta X &= 0, \qquad 1 \le i \le m, \\ \sum_{i=1}^{m} \Delta y_i A_i + \Delta S &= 0, \\ X\Delta S + S\Delta X &= \mu E - XS. \end{aligned} \tag{2.8}$$

This system has a unique solution [25], in which ΔX is not necessarily symmetric. Therefore, some symmetrization techniques are used to obtain a symmetric solution for the ΔX, see, e.g., [1, 25]. In this paper, we use the Nesterov-Todd symmetrization scheme [1] which leads us to the NT direction. Let

$$P := X^{\frac{1}{2}}(X^{\frac{1}{2}}SX^{\frac{1}{2}})^{-\frac{1}{2}}X^{\frac{1}{2}} = S^{-\frac{1}{2}}(S^{\frac{1}{2}}XS^{\frac{1}{2}})^{\frac{1}{2}}S^{-\frac{1}{2}},$$

and $D = P^{\frac{1}{2}}$. Let us define the symmetric and positive definite matrix V as follows

$$V := \frac{1}{\sqrt{\mu}} D^{-1} X D^{-1} = \frac{1}{\sqrt{\mu}} DSD. \tag{2.9}$$

This implies that

$$V^2 := \frac{1}{\mu} D^{-1} XSD. \tag{2.10}$$

On the other hand, we define

$$\begin{aligned} \bar{A}_i : &= DA_iD, \qquad 1 \le i \le m, \\ D_X : &= \frac{1}{\sqrt{\mu}} D^{-1} \Delta X D^{-1}, \\ D_S : &= \frac{1}{\sqrt{\mu}} D\Delta SD. \end{aligned} \tag{2.11}$$

Therefore, the new system can be written as below

$$\begin{aligned} \bar{A}_i \bullet D_X &= 0, \qquad 1 \le i \le m, \\ \sum_{i=1}^{m} \Delta y_i \bar{A}_i + D_S &= 0, \\ D_X + D_S &= V^{-1} - V. \end{aligned} \tag{2.12}$$

Again, this system has a unique solution as $(D_X, \Delta y, D_S)$, which is called the NT search direction. Moreover, the third equation in (2.12) is called centering equation. One can easily verify that the right hand side of the centering equation that is $V^{-1} - V$ equals to $-\psi_c'(V)$, where

$$\psi_c(t) := \frac{t^2 - 1}{2} - \log(t), \qquad \forall\, t > 0,$$

is so-called kernel function. Note that the function $\psi_c(t)$ has the following properties:

i): $\psi_c(t)$ is strictly convex function for all $t > 0$.
ii): $\psi_c(1) = \psi_c'(1) = 0$.
iii): $\lim_{t\to 0^+} \psi_c(t) = \lim_{t\to+\infty} \psi_c(t) = +\infty$.

Suppose that $\psi(t)$ is any strictly convex function on $\mathbb{R}_{++}$ and $\psi_c(1) = \psi_c'(1) = 0$. Given the kernel function $\psi(t)$ and its associated matrix function $\psi(V)$, the right-hand side of the third equation in (2.12) is replaced by $-\psi'(V)$.

Note that, the real valued matrix function $\Psi(V) : \mathbf{S}_+^n \to \mathbb{R}_+$ is defined as

$$\Psi(V) := \Psi(X,S,\mu) = Tr(\psi(V)) = \sum_{i=1}^{n} \psi(\lambda_i(V)). \tag{2.13}$$

The new search direction D_X and D_S are derived by solving the following system

$$\begin{aligned} \bar{A}_i \bullet D_X &= 0, \qquad 1 \le i \le m, \\ \sum_{i=1}^{m} \Delta y_i \bar{A}_i + D_S &= 0, \\ D_X + D_S &= -\psi'(V). \end{aligned} \tag{2.14}$$

This system has a unique solution D_X, D_S and Δy [9], which can be used to compute ΔX and ΔS via (2.11). Note that D_X and D_S are orthogonal due to orthogonality of ΔX and ΔS. Moreover,

$$D_X = D_S = 0_{n\times n} \Leftrightarrow \psi'(V) = 0_{n\times n} \Leftrightarrow V = E \Leftrightarrow \Psi(V) = 0 \Leftrightarrow XS = \mu E,$$

which implies that $X = X(\mu)$ and $S = S(\mu)$. Therefore, for $(X,y,S) \neq (X(\mu), y(\mu), S(\mu))$, we have $(\Delta X, \Delta y, \Delta S) \neq 0$. Thus, the new point is obtained as below

$$X_+ = X + \alpha \Delta X, \qquad y_+ = y + \alpha \Delta y, \qquad S_+ = S + \alpha \Delta S, \tag{2.15}$$

where, $\alpha \in (0,1]$ is the so-called *step size* and obtained by a line search strategy.

Summarizing the above arguments, we can describe one step of the IPMs based on the kernel function as follows: Starting with an interior point (X^0, y^0, S^0), $\mu_0 > 0$, an accuracy parameter $\varepsilon > 0$ and the proximity function $\Psi(V)$, induced from the kernel function $\psi(t)$ by $\Psi(V) = \sum_{i=1}^{n} \psi(\lambda_i(V))$, let a good approximation of the μ-center $(X(\mu), y(\mu), S(\mu))$ be known for $\mu > 0$. Then, the parameter μ is decreased by a factor $1-\theta$, for $\theta \in (0,1)$, and set $\mu := (1-\theta)\mu$. In general, this will increase the value of $\Psi(V)$ above τ, i.e. $\Psi(V) > \tau$. Therefore, the subsequent inner iterations are performed in order to bring the values of $\Psi(V)$ back to the situation where we have $\Psi(V) \le \tau$. In any inner iteration, to find the unique search direction, the modified Newton's system is solved. On the other hand, to obtain the search direction, first system (2.14) is solved. Then, the original search directions ΔX and ΔS are computed via (2.11). After obtaining the search direction, the new point is computed. In this case, the matrix V is calculated and the value $\Psi(V)$ is evaluated. This process is repeated until $\Psi(V) \le \tau$. Now, the algorithm performs an outer iteration with reducing the parameter μ. This procedure is repeated until we get the point in which $n\mu < \varepsilon$. In this case, we say that the current X and (y,S) are ε-approximate solutions of the primal and dual problems, respectively. This procedure is outlined in Algorithm 1 [9].

Algorithm 1: Generic Primal-dual IPM for SDO [9]

Input

a threshold parameter $\tau > 0$;

an accuracy parameter $\varepsilon > 0$;

a fixed barrier update parameter θ, $0 < \theta < 1$;

a strictly feasible pair (X^0, S^0) and $\mu^0 = 1$ such that $\Psi(X^0, S^0, \mu^0) \le \tau$.

begin
$X := X^0; S := S^0; \mu := \mu^0;$
while $n\mu > \varepsilon$ **do**
$\mu := (1-\theta)\mu;$
while $\Psi(X,S,\mu) > \tau$ **do**
solve system (2.14) and use (2.11) to obtain $(\Delta X, \Delta y, \Delta S)$;
determine a step size α;
$X_+ := X + \alpha\Delta X;$
$y_+ := y + \alpha\Delta y;$
$S_+ := S + \alpha\Delta S;$
$V := \frac{1}{\sqrt{\mu}}(D^{-1}XSD)^{\frac{1}{2}}.$
end do
end do
end

Algorithm 1 consists of inner and outer while loops which are called inner and outer iterations, respectively. The total number of iterations is bounded above by the multiplication of the inner and outer iterations and can be described as a function of the dimension n and the accuracy parameter $\varepsilon > 0$. The choice of the barrier update parameter θ plays an important role in theory and practice of IPMs. For a constant θ, let say $\theta = \frac{1}{2}$, the algorithm is called the *large-update* method, while for the θ be dependent n, let say $\theta = \frac{1}{\sqrt{n}}$, the algorithm is named the *small-update* method. It is well known that small-update methods have the best iteration bound in theory while the large update methods are practically efficient [28].

3. The new kernel function and its properties

In this section, a new parametric kernel function is presented. Then some useful properties of this function are provided. Let us define a new kernel function as:

$$\psi(t) = \frac{t^2-1}{2} - \int_1^t \left(\frac{e-1}{e^x-1}\right)^p dx, \qquad p \geq 1. \tag{3.1}$$

To analysis the algorithm, we need the first three derivatives of the function $\psi(t)$. Therefore, we present them here as follows:

$$\psi'(t) = t - \left(\frac{e-1}{e^t-1}\right)^p, \tag{3.2}$$

$$\psi''(t) = 1 + \frac{pe^t(e-1)^p}{(e^t-1)^{p+1}}, \tag{3.3}$$

$$\psi'''(t) = -\frac{pe^t(e-1)^p}{(e^t-1)^{p+1}}\left(\frac{pe^t+1}{e^t-1}\right). \tag{3.4}$$

From (3.1) and (3.2), we conclude that $\psi(1) = \psi'(1) = 0$. On the other hand, we have $\lim_{t\to 0^+}\psi(t) = \lim_{t\to+\infty}\psi(t) = +\infty$. These show that function $\psi(t)$ is indeed a kernel function. Now, we provide some properties of kernel function $\psi(t)$ which are used in the complexity analysis of Algorithm 1 based on this function.

Lemma 3.1. *For all $t > 0$, we have*

$$tpe^t - e^t + 1 > 0.$$

Proof. Let us define the function $g(t)$ by

$$g(t) := tpe^t - e^t + 1.$$

Then, we have

$$g'(t) = (p-1)e^t + tpe^t. \tag{3.5}$$

From the fact that $p \geq 1$, we conclude that $g(t)$ is increasing function for all $t > 0$. Now using the fact that $g(0) = 0$, the proof is completed. □

Lemma 3.2. *Let the function $\psi(t)$ be defined as in (3.1). Then, we have*

i): $\psi''(t) > 1, \quad$ *for all* $t > 0$.
ii): $t\psi''(t) - \psi'(t) > 0, \quad$ *for all* $t > 1$.
iii): $t\psi''(t) + \psi'(t) > 0, \quad$ *for all* $t > 0$.
iv): $\psi'''(t) < 0, \quad$ *for all* $t > 0$.

Proof. To prove the first item, we have

$$\psi''(t) = 1 + \frac{pe^t(e-1)^p}{(e^t-1)^{p+1}} \geq 1,$$

which shows that (i) holds. For the second part of the lemma, one can easily see that

$$t\psi''(t) - \psi'(t) = \left(\frac{e-1}{e^t-1}\right)^p \left(\frac{tpe^t}{e^t-1} + 1\right) > 0.$$

To prove statement (iii), using Lemma 3.1, for all $t > 0$, we have

$$t\psi''(t) + \psi'(t) = 2t + \left(\frac{e-1}{e^t-1}\right)^p \left(\frac{tpe^t - e^t + 1}{e^t-1}\right) > 0.$$

Item (iv) of the lemma can be easily followed by using (3.4). □

The first part of Lemma 3.2 is known as *superconvexity* property and provides a powerful tools for complexity analyzing of Algorithm 1. Moreover, the property mentioned in the third part of Lemma 3.2 is known as *exponential convexity* (or simply *e-convexity*) property of the kernel function $\psi(t)$ which is also crucial in the analysis of the primal-dual IPMs based on kernel functions. This property is equivalent to the convexity of the function $\psi(e^\xi)$. In [5], it has been proved that the e-convexity property is equivalent to the following inequality:

$$\psi(\sqrt{t_1 t_2}) \leq \frac{1}{2}(\psi(t_1) + \psi(t_2)), \qquad \forall\, t_1, t_2 > 0. \tag{3.6}$$

To derive some other properties of this kernel function, we define the norm-based proximity measure $\delta(V)$ as below:

$$\delta := \delta(V) = \frac{1}{2}\|\psi'(V)\| = \frac{1}{2}\sqrt{\sum_{i=1}^{n}(\psi'(\lambda_i(V)))^2} = \frac{1}{2}\|D_X + D_S\|. \tag{3.7}$$

As the matrices D_X and D_S are perpendicular with respect to inner product of the matrices, we conclude that

$$\|D_X + D_S\|^2 = \|D_X\|^2 + \|D_S\|^2.$$

Moreover, from the fact that $\psi(1) = \psi'(1) = 0$, the function $\psi(t)$ can be totally described by its second derivative according to:

$$\psi(t) = \int_1^t \int_1^\xi \psi''(\zeta) d\zeta d\xi, \tag{3.8}$$

Theorem 3.3. *(Proposition 5.2.6 in [29]) Suppose that matrices V_1 and V_2 are symmetric positive definite and Ψ is the real valued matrix function induced by the matrix function ψ. Then, one has*

$$\Psi\left(\left[V_1^{\frac{1}{2}} V_2 V_1^{\frac{1}{2}}\right]^{\frac{1}{2}}\right) \leq \frac{1}{2}\left(\Psi(V_1) + \Psi(V_2)\right).$$

Due to the superconvexity property of the kernel function $\psi(t)$, the following results can be easily proved.

Lemma 3.4. *Let the kernel function $\psi(t)$ be defined as in (3.1). Then, we have*

i): $\frac{1}{2}(t-1)^2 \leq \psi(t) \leq \frac{1}{2}\psi'(t)^2, \qquad$ *for all* $t > 0$.
ii): $\Psi(V) \leq 2\delta(V)^2, \qquad$ *for all* $V \succ 0$.
iii): $\|V\| \leq \sqrt{n} + \sqrt{2\Psi(V)}, \qquad$ *for all* $V \succ 0$.

Proof. Proof of the first part can be found in [10]. Moreover, a similar proof for the second and third statements can be found in [30]. □

4. The growth of the proximity function

This section presents the growth behavior of the proximity function after a μ-update. In the Algorithm 1, for given τ, at the start of an outer iteration and just before updating of the parameter μ, we have $\Psi(V) \leq \tau$. Assume that we are working on large-update methods. Due to the updating strategy of μ, i.e., $\mu := (1-\theta)\mu$, the matrix V is divided by a factor $\sqrt{(1-\theta)}$. This leads the proximity to be increased, in general. The subsequent inner iterations are performed in order to bring the values of $\Psi(V)$ back to the situation where we have $\Psi(V) \leq \tau$. Therefore, the largest value of $\Psi(V)$ occurs just after updating of μ. The following lemma provides an upper bound for the growth of the proximity after a μ update.

Lemma 4.1. *Suppose that $\beta \geq 1$ and $\psi(t)$ is defined as in (3.1). Thus, we have*

$$\psi(\beta t) \leq \psi(t) + \frac{(\beta^2 - 1)t^2}{2}$$

Proof. Let us define $\psi(t)$ as

$$\psi(t) = \frac{t^2 - 1}{2} + r(t),$$

where

$$r(t) = -\int_1^t \left(\frac{e-1}{e^x - 1}\right)^p dx.$$

Then, we have

$$\psi(\beta t) - \psi(t) = \frac{(\beta^2 - 1)t^2}{2} + r(\beta t) - r(t).$$

Since $\beta \geq 1$ and the fact that $r(t)$ is a decreasing function, we conclude that

$$\psi(\beta t) \leq \psi(t) + \frac{(\beta^2-1)t^2}{2}.$$

This completes the proof of lemma. □

Lemma 4.2. *Suppose that* $0 < \theta < 1$ *and* $V_+ = \frac{V}{\sqrt{1-\theta}}$. *Thus, we have*

$$\Psi(V_+) \leq \Psi(V) + \frac{\theta}{2(1-\theta)}\left(2\Psi(V) + 2\sqrt{2n\Psi(V)} + n\right).$$

Proof. Using Lemma 4.1, with $\beta = \frac{1}{\sqrt{1-\theta}}$, we have

$$\Psi(\beta V) \leq \Psi(V) + \frac{1}{2}\sum_{i=1}^{n}(\beta^2-1)\lambda_i(V)^2 = \Psi(V) + \frac{\theta\|V\|^2}{2(1-\theta)}.$$

Now, the proof is easily followed by using the third part of Lemma 3.4. □

5. An estimation for the step size

In this section, a default value for the step size α during an inner iteration of the Algorithm 1 is computed. After an inner iteration, the new point is computed as below:

$$\begin{aligned} X_+ &= X + \alpha\Delta X = X + \alpha\sqrt{\mu}DD_XD = \sqrt{\mu}D(V + \alpha D_X)D, \\ S_+ &= S + \alpha\Delta S = S + \alpha\sqrt{\mu}D^{-1}D_SD^{-1} = \sqrt{\mu}D^{-1}(V + \alpha D_S)D^{-1}, \end{aligned}$$

where D_X, D_S and D are defined by (2.11) and α denotes the step size. Moreover, from (2.10), we conclude that

$$V_+ = \frac{1}{\sqrt{\mu}}(D^{-1}X_+S_+D)^{\frac{1}{2}}. \tag{5.1}$$

It is easily seen that the matrix V_+^2 is unitarily similar to the matrix $X_+^{\frac{1}{2}}S_+X_+^{\frac{1}{2}}$ and therefore to the matrix $\bar{V}_+^2 = (V + \alpha D_X)^{\frac{1}{2}}(V + \alpha D_S)(V + \alpha D_X)^{\frac{1}{2}}$. This implies that both of the matrices have the same eigenvalues. So, the following result can be concluded [9]:

$$\Psi(V_+) = \Psi(\bar{V}_+).$$

Based on the Theorem 3.3, we can conclude that

$$\Psi(\bar{V}_+) \leq \frac{1}{2}\left(\Psi(V + \alpha D_X) + \Psi(V + \alpha D_S)\right),$$

which implies that

$$\Psi(V_+) \leq \frac{1}{2}\left(\Psi(V + \alpha D_X) + \Psi(V + \alpha D_S)\right).$$

Let us define the univariate functions $f(\alpha)$ and $f_1(\alpha)$ as follows

$$\begin{aligned} f(\alpha) &:= \Psi(V_+) - \Psi(V) = \Psi(\bar{V}_+) - \Psi(V), \\ f_1(\alpha) &:= \frac{1}{2}\left[\Psi(V + \alpha D_X) + \Psi(V + \alpha D_S)\right] - \Psi(V). \end{aligned} \tag{5.2}$$

Then, we have $f(\alpha) \leq f_1(\alpha)$. Moreover, the function $f_1(\alpha)$ is a convex function with respect to α due to the fact that Ψ is a convex function and its arguments in the first two terms are linear with respect to α. To proceed, we need the first two derivatives of the function $f_1(\alpha)$ with respect to α. We have

$$\begin{aligned} f_1'(\alpha) &= \frac{1}{2} Tr\left(\psi'(V+\alpha D_X)D_X + \psi'(V+\alpha D_S)D_S\right), \\ f_1''(\alpha) &= \frac{1}{2} Tr\left(\psi''(V+\alpha D_X)D_X^2 + \psi''(V+\alpha D_S)D_S^2\right). \end{aligned} \tag{5.3}$$

For notational convenience, in the sequel, we use $\delta := \delta(V)$. By taking $\alpha = 0$ in (5.3) and using the third equation of system (2.14), the following result can be concluded

$$f_1'(0) = \frac{1}{2} Tr\left(\psi'(V)(D_X + D_S)\right) = \frac{1}{2} Tr(-\psi'(V)^2) = -2\delta^2, \tag{5.4}$$

where the last equality is obtained from (3.7). In what follows, we are going to introduce conditions on the step size α in which the condition $f_1'(\alpha) < 0$ holds. This allows us to conclude that the function $f(\alpha)$ is a decreasing function using the fact that $f(0) = f_1(0) = 0$ and $f(\alpha) \leq f_1(\alpha)$. The following lemma provides an upper bound for the second derivative of the function $f_1(\alpha)$. One can find its proof in [6].

Lemma 5.1. *Let f_1 be defined as in (5.2). Then, the second derivative of the function f_1 with respect to α satisfies the following inequality*

$$f_1''(\alpha) \leq 2\delta^2 \psi''(\lambda_n(V) - 2\alpha\delta).$$

Our aim in introducing a suitable step size is that it should be chosen so that X_+ and S_+ are feasible and $f(\alpha)$ decreases sufficiently. The procedure for selecting the largest possible step size is almost a "word-by-word" extension of the LO case in [31]. Therefore, we omit the proof of the following results here and refer the reader to [31, 32] and the references therein for their proof.

Lemma 5.2. *Inequality $f_1'(\alpha) \leq 0$ holds if α satisfies the following inequality*

$$-\psi'(\lambda_n(V) - 2\alpha\delta) + \psi'(\lambda_n(V)) \leq 2\delta. \tag{5.5}$$

Lemma 5.3. *Assume that $\rho : [0,\infty) \to (0,1]$ is the inverse of the function $-\frac{1}{2}\psi'(t)$ in the interval $(0,1]$. Then, the largest possible value for the α satisfying (5.5) is given by*

$$\overline{\alpha} = \frac{1}{2\delta}(\rho(\delta) - \rho(2\delta)). \tag{5.6}$$

Lemma 5.4. *Let $\overline{\alpha}$ be defined as in (5.6). Then, we have*

$$\overline{\alpha} \geq \frac{1}{\psi''(\rho(2\delta))}. \tag{5.7}$$

Due to Lemmas 5.3 and 5.4, we consider the following value for the step size in the further analysis as the default value

$$\tilde{\alpha} = \frac{1}{\psi''(\rho(2\delta))}. \tag{5.8}$$

It is easily seen that $\tilde{\alpha} \leq \overline{\alpha}$. Our next lemma provides the amount of decrease in the proximity function during an inner iteration.

Lemma 5.5. *(Lemma 5.2 in [6]) For the step size α satisfying $\alpha \leq \bar{\alpha}$, we have*

$$f(\alpha) \leq -\alpha\delta^2. \tag{5.9}$$

Lemma 5.6. *Let $\Psi(V) \geq 1$, and ρ and $\tilde{\alpha}$ be defined as in Lemma 5.3 and (5.8), respectively. Then, one has*

$$f(\tilde{\alpha}) \leq -\frac{\delta^2}{\psi''(\rho(2\delta))} \leq \Theta\left(-\frac{\delta^{\frac{p-1}{p}}}{p}\right). \tag{5.10}$$

Proof. By using Lemma 5.5 and the fact that $\tilde{\alpha} \leq \overline{\alpha}$, we conclude that $f(\tilde{\alpha}) \leq -\tilde{\alpha}\delta^2 = -\frac{\delta^2}{\psi''(\rho(2\delta))}$, which implies the first inequality.

To complete proof, we need the inverse function $t = \rho(\delta)$ of $-\frac{1}{2}\psi'(t)$ for $t \in (0,1]$. To this end, we derive t from the equation

$$-\left(t - (\frac{e-1}{e^t-1})^p\right) = 2s.$$

From the fact that $t \in (0,1]$, we conclude that

$$(\frac{e-1}{e^t-1})^p \leq 2s+1. \tag{5.11}$$

Putting $t = \rho(2\delta)$, we conclude that $4\delta = -\psi'(t)$. These imply that

$$(\frac{e-1}{e^t-1})^p \leq 4\delta+1 \Rightarrow \frac{1}{e^t-1} \leq \frac{(4\delta+1)^{\frac{1}{p}}}{e-1} \leq (4\delta+1)^{\frac{1}{p}}. \tag{5.12}$$

Now, from (5.11) and (5.12), we obtain

$$\tilde{\alpha} = \frac{1}{\psi''(t)} = \frac{1}{1+pe^t\frac{(e-1)^p}{(e^t-1)^{p+1}}} \geq \frac{1}{1+3p(4\delta+1)^{\frac{p+1}{p}}} = \Theta\left(\frac{1}{p\delta^{\frac{p+1}{p}}}\right).$$

This implies that

$$f(\tilde{\alpha}) \leq -\tilde{\alpha}\delta^2 \leq -\frac{\delta^2}{\psi''(\rho(2\delta))} \leq \Theta\left(-\frac{\delta^{\frac{p-1}{p}}}{p}\right).$$

This completes the proof of lemma. □

Note that, using the second part of Lemma 3.4, we have

$$f(\tilde{\alpha}) \leq \Theta\left(-\frac{\delta^{\frac{p-1}{p}}}{p}\right) \leq \Theta\left(-\frac{\Psi^{\frac{p-1}{2p}}}{p}\right). \tag{5.13}$$

6. Iteration complexity

In this section, we derive the worst case iteration bound for the Algorithm 1 based on the proximity measure Ψ induced from the kernel function ψ defined by (3.1). As mentioned before, we utilize the value of $\tilde{\alpha}$, defined by (5.8), as a default value for the step size during an inner iteration. In this manner, from Lemma 4.2, we have

$$\Psi(V_+) \leq \Psi(V) + \frac{\theta}{2(1-\theta)}\left(2\Psi(V) + 2\sqrt{2n\Psi(V)} + n\right), \tag{6.1}$$

right after updating the parameter μ to $(1-\theta)\mu$, for $\theta \in (0,1)$. As we are interested to work in large neighborhood of the central path, from now on, we assume that $\tau = O(n) \geq 1$. Besides, in the large update methods we let $\theta = \Theta(1)$.

At the beginning of an outer iteration, and right before updating of μ, we have $\Psi(V) \leq \tau$. Based on (6.1), the value of $\Psi(V)$ exceeds the threshold τ after updating of μ. Therefore, we should compute the

number of inner iterations that are needed to return the value of the proximity back to the case in which we have $\Psi(V) \leq \tau$. Let the value of the proximity after μ-update be denoted by Ψ_0, and the subsequent values by Ψ_j, for $j = 1, \ldots, L-1$, where L is the total number of inner iterations in an outer iteration. Inequality (6.1) together with $\Psi(V) \leq \tau = O(n)$ implies that

$$\Psi_0 \leq \tau + \frac{\theta}{2(1-\theta)} \left(2\tau + 2\sqrt{2n\tau} + n \right) = O(n). \tag{6.2}$$

Now, using (5.13) and the fact that, in an inner iteration, we have $\Psi_j > \tau \geq 1$, the decrease of Ψ in each inner iteration can be computed by

$$\Psi_{j+1} \leq \Psi_j - \kappa \Delta \Psi_j \qquad j = 0, 1, \ldots, L-1, \tag{6.3}$$

where κ is some positive constant and $\Delta\Psi_j$ is defined by

$$\Delta \Psi_j = \frac{\Psi_j^{\frac{p-1}{2p}}}{p}. \tag{6.4}$$

Now, we state the number of inner iterations during an outer iteration. For this purpose, the following technical lemma plays an important role.

Lemma 6.1 (Lemma 1.3.1 in [5])**.** *Given $\alpha \in [0,1]$ and $t \geq -1$, one has*

$$(1+t)^\alpha \leq 1 + \alpha t.$$

The following theorem provides the worst case upper bound for the total number of inner iterations during an outer iteration. To prove the theorem, without loss of generality, we assume that at the present iteration (j-th iteration) $0 < \kappa < p\Psi_j^{\frac{p+1}{2p}}$ [5].

Theorem 6.2. *Let μ be updated by $\mu := (1-\theta)\mu$ and $\tau \geq 1$. Then, the number of inner iterations that are required to return the iterations back to the situation where $\Psi(V) \leq \tau$ is bounded above by*

$$L \leq 1 + \frac{2p^2}{\kappa(p-1)} \Psi_0^{\frac{p+1}{2p}}. \tag{6.5}$$

Proof. Using (6.3), for all $j = 0, 1, \ldots, L-1$, we have

$$0 \leq \Psi_{j+1}^{\frac{p+1}{2p}} \leq \left(\Psi_j - \frac{\kappa}{p} \Psi_j^{\frac{p-1}{2p}} \right)^{\frac{p+1}{2p}} = \Psi_j^{\frac{p+1}{2p}} \left(1 - \frac{\kappa}{p} \Psi_j^{-\frac{p+1}{2p}} \right)^{\frac{p+1}{2p}}. \tag{6.6}$$

$$\tag{6.7}$$

By taking $\alpha = \frac{p+1}{2p}$ in Lemma 6.1, we conclude that the following inequality

$$\Psi_{j+1}^{\frac{p+1}{2p}} \leq \Psi_j^{\frac{p+1}{2p}} \left(1 - \frac{\kappa(p+1)}{2p^2} \Psi_j^{-\frac{p+1}{2p}} \right) = \Psi_j^{\frac{p+1}{2p}} - \frac{\kappa(p+1)}{2p^2}, \tag{6.8}$$

where the last inequality is obtained from Lemma 6.1. By subsequently using (6.6), we obtain

$$\Psi_{j+1}^{\frac{p+1}{2p}} \leq \Psi_0^{\frac{p+1}{2p}} - \frac{j\kappa(p+1)}{2p^2}.$$

Letting $j = L-1$, one has

$$0 \leq \Psi_L^{\frac{p+1}{2p}} \leq \Psi_0^{\frac{p+1}{2p}} - \frac{(L-1)\kappa(p+1)}{2p^2},$$

which shows that

$$L \leq 1 + \frac{2p^2}{\kappa(p-1)} \Psi_0^{\frac{p+1}{2p}}.$$

This completes the proof of the theorem. □

Using (6.2), we have $\Psi_0 = O(n)$. Now, from Theorem 6.2, we obtain the following upper bound for the total number of inner iterations during an outer iteration

$$L \leq \left\lceil 1 + \frac{2p^2}{\kappa(p-1)} \Psi_0^{\frac{p+1}{2p}} \right\rceil = \left\lceil O\left(p\, n^{\frac{p+1}{2p}}\right) \right\rceil. \tag{6.9}$$

On the other hand, for given accuracy parameter $\varepsilon > 0$ and $\theta = \Theta(1)$, the total number of outer iterations for getting $n\mu \leq \varepsilon$ are bounded above by $O\left(\frac{1}{\theta} \log \frac{n}{\varepsilon}\right)$, see Lemma I.36 in [28]. Therefore, an upper bound for the total number of iterations of Algorithm 1 is obtained by multiplying the total number of inner and outer iterations. By omitting the integer bracket in (6.9) which does not change the order of complexity, we then derive the following total number of iterations to get an ε-solution, i.e., a solution that satisfies $n\mu \leq \varepsilon$, as follows:

$$O\left(p\, n^{\frac{p+1}{2p}} \log \frac{n}{\varepsilon}\right).$$

This bound matches with the so far best-known iteration bound for large update IPMs.

7. Numerical results

In this section, we give some numerical results to see some advantages of using the mentioned new kernel function. The results are obtained by performing of Algorithm1 with the proposed kernel function on some test problems given in [9, 33, 34, 35]. Also the obtained results are compared with the one generated by other eight famous kernel function which presented in Table 1. All experiments are implemented in MATLAB 7.10.0 (R2010a) environment on a PC with CPU 3.1 GHz and 12Gb RAM memory and double precision format. In all tables, "Iter", "Time" and "gap" stand for the number of iterations, CPU time (in second) and the value of $C \bullet X - b^T y$, respectively. Furthermore, $\psi_{new,p}$ stands for the new proposed kernel function with different values p. Note that, each test example is denoted by "Exam." in all tables. To implement the algorithm, the following parameters are used:

$$\begin{aligned} \varepsilon &= 10^{-8}, \qquad \tau = 1, \qquad \mu^0 = 1, \\ \theta &\in \{0.05, 0.4, 0.6, 0.95\}. \end{aligned}$$

Example 7.1. Let us consider the following SDO problem [9]:

$$A_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -2 & -1 \\ 0 & -1 & 1 & -1 & -2 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0 & 0 & -2 & 2 & 0 \\ 0 & 2 & 1 & 0 & 2 \\ -2 & 1 & -2 & 0 & 1 \\ 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 2 \end{bmatrix}, \quad A_3 = \begin{bmatrix} 2 & 2 & -1 & -1 & 1 \\ 2 & 0 & 2 & 1 & 1 \\ -1 & 2 & 0 & 1 & 0 \\ -1 & 1 & 1 & -2 & 0 \\ 1 & 1 & 0 & 0 & -2 \end{bmatrix};$$

TABLE 1. Considered kernel functions

$\psi_i(t)$	References
$\psi_1(t)=\frac{t^2-1}{2}-\log(t)$	[28]
$\psi_2(t)=\frac{t^2-1}{2}-(t-1)e^{\frac{1}{t}-1}$	[30]
$\psi_3(t)=\frac{t^2-1}{2}+\frac{\frac{1}{t}-1}{2}-\frac{t-1}{2}$	[5]
$\psi_4(t)=\frac{t^2-1}{2}+\frac{6}{\pi}\tan(h(t)),\quad h(t)=\frac{1-t}{2+4t}\pi$	[11]
$\psi_5(t)=\frac{t^2-1}{2}+\frac{4}{\pi}\cot(h(t)),\quad h(t)=\frac{\pi t}{1+t}$	[12]
$\psi_6(t)=\frac{t^2-1}{2}-\log(t)+\frac{1}{8}\tan^2(h(t)),\quad h(t)=\frac{1-t}{2+4t}\pi$	[18]
$\psi_7(t)=\frac{t^2-1}{2}-\int_1^t e^{3(\tan(h(x))-1)}dx,\quad h(x)=\frac{\pi}{2+2x}\pi$	[17]
$\psi_8(t)=\frac{t^2-1}{2}+\frac{4}{3p\pi}\tan^{3p}(h(t))-\frac{4}{3p\pi},\quad h(t)=\frac{\pi}{2+2t}\pi$	[16]

$$C=\begin{bmatrix} 3 & 3 & -3 & 1 & 1\\ 3 & 5 & 3 & 1 & 2\\ -3 & 3 & -1 & 1 & 2\\ 1 & 1 & 1 & -3 & -1\\ 1 & 2 & 2 & -1 & -1 \end{bmatrix},\quad b=\begin{bmatrix} -2\\ 2\\ -2 \end{bmatrix}.$$

To solving this test problem, the algorithm is started by initial strictly feasible primal and dual solution $X=E, S=E$ and $y=(1,1,1)^T$. The optimal solutions of the primal and dual problems (P) and (D) are:

$$X^*=\begin{bmatrix} 0.0704 & -0.0717 & 0.0133 & 0.0695 & -0.1465\\ -0.0717 & 0.0753 & -0.0162 & -0.0636 & 0.1631\\ 0.0133 & -0.0162 & 0.0105 & -0.0124 & -0.0703\\ 0.0695 & -0.0636 & -0.0124 & 0.1695 & 0.0178\\ -0.1465 & 0.1631 & -0.0703 & 0.0178 & 0.5829 \end{bmatrix};$$

$$S^*=\begin{bmatrix} 1.4249 & 0.5669 & -0.0204 & -0.4045 & 0.2125\\ 0.5669 & 1.0918 & 0.3289 & 0.2125 & -0.1216\\ -0.0204 & 0.3289 & 1.1921 & 0.2125 & 0.0459\\ -0.4045 & 0.2125 & 0.2125 & 0.2912 & -0.1420\\ 0.2125 & -0.1216 & 0.0459 & -0.1420 & 0.0991 \end{bmatrix},\quad y^*=\begin{bmatrix} 0.8580\\ 1.0960\\ 0.7875 \end{bmatrix}.$$

Algorithm 1 is performed on the above mentioned SDO problem with the new proposed kernel function $\psi(t)$ and the eight kernel functions listed in Table 1.

Example 7.2. (Example 1 in [33]) For this example, we have

$$C(i,j)=-1,\qquad \forall\, i,j\in\{1,2\};$$

$$A_1=\begin{bmatrix} 1 & -1\\ -1 & 1 \end{bmatrix},\quad A_2=E,\qquad b=\begin{bmatrix} 1\\ 1 \end{bmatrix};$$

$$X^0=diag(0.5,0.5),\quad y^0=\begin{bmatrix} 0\\ -3 \end{bmatrix},\quad S^0=\begin{bmatrix} 2 & -1\\ -1 & 2 \end{bmatrix}.$$

The optimal solution of this example is:

$$X^* = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}, \quad S^* = 10^{-8} \begin{bmatrix} 0.397 & 0 \\ 0 & 0.397 \end{bmatrix}, \qquad y^* = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, \quad C \bullet X = b^T y = -1.$$

Example 7.3. (Example 2 in [33]) We have
$C = diag(5,8,8,5)$, $A_4 = E$, $b = (1,1,1,2)^T$ and A_k , $k = 1,2,3$ are defined as:

$$A_k(i,j) = \begin{cases} 1 & if\ i = j = k\ or i = j = k+1; \\ -1 & if\ i = k,\ j = k+1\ or\ i = k+1,\ j = k; \\ 0 & otherwise. \end{cases}$$

$$X^0 = \frac{1}{2}E, \quad y^0 = (1.5, 1.5, 1.5, 1.5)^T, \quad S^0 = \begin{bmatrix} 2 & 1.5 & 0 & 0 \\ 1.5 & 3.5 & 1.5 & 0 \\ 0 & 1.5 & 3.5 & 1.5 \\ 0 & 0 & 1.5 & 2 \end{bmatrix}.$$

The optimal solution of this example has the following information:

$$X^* = \begin{bmatrix} 0.7500 & 0 & 0 & -0.0032 \\ 0 & 0.2500 & -0.2500 & 0 \\ 0 & -0.2500 & 0.2500 & 0 \\ -0.0032 & 0 & 0 & 0.7500 \end{bmatrix}, \qquad S^* = \begin{bmatrix} 0.0000 & 0 & 0 & 0 \\ 0 & 1.5000 & 1.5000 & 0 \\ 0 & 1.5000 & 1.5000 & 0 \\ 0 & 0 & 0 & 0.0000 \end{bmatrix},$$

$$y^* = [0, 1.5, 0, 5]^T, \qquad C \bullet X = b^T y = 11.5.$$

The numerical results of performing the algorithm based on the kernel functions denoted in Table 1 for solving Examples 7.1–7.3 are demonstrated in Table 2. Furthermore, the results for the new kernel function with difference values p and θ, are presented in Table 3. To compare the behavior of any kernel functions, the averages iterations and CPU times are calculated for these tables. For the new kernel functions, the average of obtained results are denoted in below of the second part of Table 3. Based on the obtained results in Average rows, we can conclude that the kernel functions ψ_6, ψ_7 and ψ_8 have the best results between the kernel functions listed in Table 1. Therefore, for performing the algorithm for other examples, we use these kernel functions. Based on the obtained results in Tables 2 and 3, we conclude that the new kernel function with $p = 1$ has the best results than the other kernel functions.

Our next goal in this section is to perform the algorithm for solving two examples that have variant free. For Example 7.4, we perform the algorithm with $m \in \{10, 25, 50, 100\}$. Moreover, for the Example 7.5, the algorithm is executed with $m \in \{100, 250\}$. For two test problems, we have performed the algorithm with kernel functions ψ_6, ψ_7, ψ_8 and ψ_{new} which have the best results in the previous examples.

TABLE 2. Number of iterations for the Example 7.1.

		ψ_1	ψ_2	ψ_3	ψ_4	ψ_5	ψ_6	ψ_7	ψ_8
$\theta = 0.05$	Iter	116	185	84	38	54	33	32	25
Exam. 1	gap	1.02E-8	1.0E-8	1.10E-8	1.34E-8	1.26E-8	1.48E-8	1.26E-8	2.23E-8
	CPU	3.04	3.15	0.26	0.17	0.20	0.16	0.15	0.14
	Iter	84	126	80	48	58	44	43	44
Exam. 2	gap	1.53E-8	1.30E-8	1.36E-8	1.95E-8	1.72E-8	2.37E-8	3.19E-8	2.79E-8
	CPU	0.29	0.49	0.31	0.27	0.28	0.22	0.23	0.26
	Iter	157	777	130	64	83	56	55	57
Exam. 3	gap	1.17E-8	9.74E-9	1.20E-8	1.83E-8	1.49E-8	2.14E-8	2.17E-8	1.83E-8
	CPU	0.48	1.63	0.41	0.31	0.35	0.26	0.28	0.29
$\theta = 0.4$	Iter	39	77	39	38	39	26	26	20
Exam. 1	gap	7.93E-9	6.86E-9	7.93E-9	7.95E-9	7.93E-9	9.92E-9	9.92E-9	1.14E-8
	CPU	0.11	0.19	0.11	0.11	0.11	0.10	0.08	0.07
	Iter	64	78	61	41	48	38	38	38
Exam. 2	gap	1.20E-8	9.54E-9	1.06E-8	2.07E-8	1.41E-8	2.22E-8	2.22E-8	2.22E-8
	CPU	0.15	0.17	0.15	0.13	0.17	0.12	0.12	0.12
	Iter	91	196	79	52	64	48	47	48
Exam. 3	gap	1.22E-8	9.10E-9	1.14E-8	1.49E-8	1.26E-8	1.85E-8	1.55E-8	1.85E-8
	CPU	0.21	0.40	0.19	0.14	0.16	0.12	0.12	0.13
$\theta = 0.6$	Iter	42	43	42	22	22	21	21	21
Exam. 1	gap	9.29E-9	9.35E-9	9.29E-9	1.35E-8	1.35E-8	1.25E-8	1.25E-8	1.25E-8
	CPU	0.14	0.12	0.12	0.07	0.07	0.08	0.07	0.07
	Iter	53	64	53	42	42	38	38	38
Exam. 2	gap	1.41E-8	1.19E-8	1.41E-8	1.75E-8	1.75E-8	2.43E-8	1.92E-8	2.43E-8
	CPU	0.13	0.15	0.14	0.12	0.15	0.10	0.11	0.11
	Iter	71	154	66	46	55	44	44	44
Exam. 3	gap	8.65E-9	7.12E-9	8.95E-9	1.40E-8	9.70E-9	1.40E-8	1.40E-8	1.40E-8
	CPU	0.16	0.32	0.17	0.12	0.14	0.11	0.12	0.11
$\theta = 0.95$	Iter	21	28	21	18	21	17	17	17
Exam. 1	gap	4.36E-9	4.00E-9	4.36E-9	4.82E-9	4.36E-9	7.28E-9	7.28E-9	7.28E-9
	CPU	0.06	0.08	0.07	0.07	0.07	0.07	0.06	0.06
	Iter	43	47	43	35	38	35	35	35
Exam. 2	gap	2.23E-9	2.11E-9	2.23E-9	4.03E-9	3.14E-9	4.03E-9	4.03E-9	4.03E-9
	CPU	0.09	0.12	0.09	0.08	0.09	0.07	0.07	0.08
	Iter	49	71	46	38	42	37	36	37
Exam. 3	gap	3.67E-9	3.18E-9	3.76E-9	6.14E-9	4.47E-9	5.26E-9	7.39E-9	5.26E-9
	CPU	0.12	0.16	0.11	0.09	0.10	0.09	0.09	0.09
Average	Iter	69.16	153.83	62	40.16	47.66	36.41	36	35.33
	CPU	0.41	0.58	0.17	0.14	0.15	0.12	0.12	0.12

TABLE 3. Number of iterations for the new kernel function on the Examples 7.1 and 7.2.

θ		Exam. 1			Exam. 2			Exam. 3		
		$\psi_{new,1}$	$\psi_{new,2}$	$\psi_{new,2.5}$	$\psi_{new,1}$	$\psi_{new,2}$	$\psi_{new,2.5}$	$\psi_{new,1}$	$\psi_{new,2}$	$\psi_{new,2.5}$
	Iter	25	25	25	38	45	52	40	47	61
.05	gap	2.23E-8	2.23E-8	2.23E-8	2.14E-8	3.22E-8	1.79E-8	3.39E-8	2.62E-8	1.86E-8
	CPU	0.13	0.14	0.14	0.24	0.27	0.29	0.26	0.27	0.30
	Iter	20	20	20	36	42	49	37	40	51
0.4	gap	1.14E-8	1.14E-8	1.14E-8	3.22E-8	2.22E-8	1.33E-8	2.64E-8	2.56E-8	1.62E-8
	CPU	0.07	0.07	0.08	0.12	0.12	0.14	0.11	0.11	0.13
	Iter	21	21	22	34	38	42	36	43	43
0.6	gap	1.35E-8	1.35E-8	1.35E-8	3.48E-8	2.75E-8	1.75E-8	1.74E-8	1.40E-8	1.45E-8
	CPU	0.07	0.07	0.07	0.10	0.11	0.12	0.10	0.11	0.11
	Iter	15	15	15	33	35	36	34	34	38
0.95	gap	1.00E-8	1.00E-8	1.00E-8	4.03E-9	6.78E-9	3.96E-9	8.00E-9	3.36E-8	2.08E-8
	CPU	0.05	0.06	0.06	0.07	0.09	0.08	0.09	0.08	0.10
	Iter	30.75	33.75	33.83						
Average										
	CPU	0.11	0.12	0.13						

Example 7.4. (Example 4 in [33]) This problem has the following data
$C = -E$, A_k, $k = 1,2,3,...,m$ are denoted as:

$$A_k(i,j) = \begin{cases} 1 & if\ i = j = k; \\ 1 & if\ i = j\ and\ i = k+m; \\ 0 & otherwise. \end{cases}$$

$$X^0 = \begin{cases} 1.5 & i \le j; \\ 0.5 & i > j; \end{cases} \qquad y^0(i) = -2,\ i = 1,2,3,...,m;$$

$$S^0 = E, \quad b(i) = 2,\ i = 1,2,3,...,m.$$

The obtained results of performed Algorithm for $m \in \{10,25,50,100\}$, are shown in Tables 7–4. Note that, the averages of number iterations and CPU times for this example are showed in Table 7

Example 7.5. (Example cube in [34]) This example has the following information
$n = 2m$, $m \in \{100,250\}$, $b = (2,2,2,...,2)^T$, A_k, $k = 1,2,3,...,m$ are defined as:

$$A_k(i,j) = \begin{cases} 1 & if\ i = j = k\ \ or\ i = j = k+m; \\ 25 & if\ i = j = k+1 \quad or\ i = j = k+m+1; \\ -5 & if\ i = k, j = k+1\ or\ i = k+m,\ j = k+m+1; \\ -5 & if\ i = k+1,\ j = k\ or\ i = k+m+1,\ j = k+m; \\ 0 & otherwise. \end{cases}$$

$$C = -2E, \quad y^0 = (0,0,...,0)^T, \quad S^0 = X^0 = E.$$

TABLE 4. Number of iterations for the Example 7.4 with $m = 10$.

θ		ψ_i					
		ψ_6	ψ_7	ψ_8	$\psi_{new,1}$	$\psi_{new,2}$	$\psi_{new2.5}$
	Iter	48	48	51	23	33	37
0.05	gap	1.65E-8	2.01E-8	4.56E-9	1.79E-8	2.22E-8	1.89E-8
	CPU	6.23	4.35	5.23	2.15	2.85	3.12
	Iter	42	32	32	19	25	31
0.4	gap	2.23E-8	2.01E-8	8.35E-9	1.87E-8	7.53E-9	1.14E-8
	CPU	1.39	0.90	0.91	0.64	0.81	0.85
	Iter	38	23	23	18	18	23
0.6	gap	2.13E-8	5.42E-9	4.34E-9	1.94E-8	1.94E-8	3.12E-8
	CPU	0.97	0.67	0.67	0.57	0.57	0.75
	Iter	24	21	21	16	18	21
0.95	gap	8.99E-9	1.45E-9	9.75E-9	2.68E-8	2.09E-8	9.75E-10
	CPU	0.67	0.60	0.58	0.52	0.57	0.67

TABLE 5. Number of iterations for the Example 7.4 with $m = 25$.

θ		ψ_i					
		ψ_6	ψ_7	ψ_8	$\psi_{new,1}$	$\psi_{new,2}$	$\psi_{new2.5}$
	Iter	67	65	66	23	33	44
0.05	gap	5.13E-9	5.44E-9	3.76E-9	1.21E-8	1.34E-8	1.37E-8
	CPU	74.12	76.38	74.14	38.19	41.00	53.11
	Iter	39	47	38	22	34	37
0.4	gap	8.48E-9	2.15E-8	6.04E-9	1.48E-8	3.19E-9	5.19E-9
	CPU	41.19	48.28	44.21	28.32	36.92	38.73
	Iter	31	37	28	20	22	31
0.6	gap	8.86E-9	3.58E-9	5.41E-9	2.01E-8	6.12E-9	1.34E-8
	CPU	35.95	41.19	37.31	19.54	22.43	32.88
	Iter	22	23	21	16	19	24
0.95	gap	2.08E-8	1.54E-9	4.12E-9	1.07E-8	5.98E-9	3.57E-9
	CPU	24.83	27.60	26.54	16.07	18.84	25.96

The numerical results from applying the algorithm for Example 7.5 are denoted in Tables 8–9. Moreover, we compute the averages number of iterations and CPU times for Example 7.5 and denote in Table 9.

Example 7.6. (Some test problems form SDPLIB collection [35]) Now, we are going to present some numerical results by considering test problems from SDPLIB collection. The algorithm is started with initial point as $X^0 = S^0 = E$ and $y^0 = 0$. We have performed the algorithm for the new proposed kernel function with $p = 1$ and kernel functions ψ_6, ψ_7, ψ_8. The related numerical results for $\theta = 0.99$ are demonstrated in Table 10. Moreover, in this table we denote the size problems with m and n respectively.

TABLE 6. Number of iterations for the Example 7.4 with $m = 50$.

θ		ψ_i					
		ψ_6	ψ_7	ψ_8	$\psi_{new,1}$	$\psi_{new,2}$	$\psi_{new2.5}$
	Iter	69	68	69	24	33	47
0.05	gap	3.73E-9	4.12E-9	3.73E-9	2.12E-8	1.48E-8	8.13E-9
	CPU	168.53	143.01	181.82	92.15	102.17	123.44
	Iter	42	51	42	23	35	39
0.4	gap	6.45E-9	1.73E-8	6.45E-9	6.52E-9	7.13E-9	7.02E-9
	CPU	92.32	108.45	90.79	74.19	80.43	82.16
	Iter	35	36	35	21	23	32
0.6	gap	8.48E-9	8.01E-9	8.48E-9	1.85E-8	2.89E-9	1.65E-8
	CPU	83.19	91.29	82.38	53.72	59.93	72.14
	Iter	27	30	27	16	19	24
0.95	gap	1.08E-9	1.06E-9	1.08E-9	1.34E-8	7.47E-9	4.49E-9
	CPU	62.39	71.06	65.36	36.92	42.53	54.61

TABLE 7. Number of iterations for the Example 7.4 with $m = 100$.

θ		ψ_i					
		ψ_6	ψ_7	ψ_8	$\psi_{new,1}$	$\psi_{new,2}$	$\psi_{new2.5}$
	Iter	74	73	74	25	36	48
0.05	gap	1.35E-8	8.12E-9	4.23E-9	1.32E-8	1.94E-8	1.01E-8
	CPU	2875.96	2736.65	2816.43	1012.51	1365.14	1993.00
	Iter	54	54	54	23	32	44
0.4	gap	3.64E-9	6.95E-9	6.34E-9	7.12E-9	6.43E-9	7.02E-9
	CPU	938.01	973.72	901.50	913.13	1051.42	1243.88
	Iter	43	43	43	21	29	37
0.7	gap	6.12E-9	6.12E-9	6.12E-9	3.14E-9	5.49E-9	4.36E-9
	CPU	583.25	602.93	536.79	544.56	792.67	921.89
	Iter	24	24	24	16	23	24
0.95	gap	8.99E-9	8.99E-9	8.99E-9	2.68E-9	4.43E-9	3.93E-9
	CPU	301.34	328.12	356.32	302.13	412.65	543.45
	Iter	42.4	42.2	40.5	20.4	27.0	33.9
Average							
	CPU	330.27	328.19	326.00	195.83	251.76	324.42

Based on the obtained results of performing the algorithm for solving the above examples, the following results are concluded:

- The results in Tables 2 and 3 show that the new kernel function with $p = 1$ has the better results than the other kernel functions. This function reduces the the number of iterations and CPU time about 13% and 8% than the similar results obtained by using the function ψ_8 respectively.

TABLE 8. Number of iterations for the Example 7.5 with $m = 100$.

θ		ψ_i					
		ψ_6	ψ_7	ψ_8	$\psi_{new,1}$	$\psi_{new,2}$	$\psi_{new2.5}$
	Iter	73	75	73	34	41	48
0.05	gap	1.89E-8	2.01E-8	1.89E-8	3.42E-9	6.12E-9	8.73E-9
	CPU	2514.98	2634.52	2698.78	983.07	1151.80	1291.87
	Iter	59	62	59	26	38	39
0.4	gap	2.04E-8	1.81E-8	2.04E-8	4.13E-9	5.68E-9	7.12E-9
	CPU	1975.24	1914.34	1992.14	375.16	412.09	589.61
	Iter	44	44	44	23	24	29
0.6	gap	1.35E-8	1.35E-8	1.35E-8	4.12E-9	5.43E-9	4.83E-9
	CPU	1736.87	1749.59	1801.16	312.78	328.49	390.12
	Iter	26	26	26	16	18	23
0.95	gap	8.92E-9	8.92E-9	8.92E-9	2.56E-8	1.83E-3	8.99E-9
	CPU	1022.45	978.59	1012.43	194.85	227.99	300.199

TABLE 9. Number of iterations for the Example 7.5 with $m = 250$.

θ		ψ_i					
		ψ_6	ψ_7	ψ_8	$\psi_{new,1}$	$\psi_{new,2}$	$\psi_{new2.5}$
	Iter	82	84	82	47	49	58
0.05	gap	1.35E-8	2.12E-8	1.35E-8	2.13E-8	1.75E-8	4.12E-9
	CPU	29832.45	30124.37	29342.14	18372.16	19012.34	21685.34
	Iter	58	56	58	35	48	52
0.4	gap	4.15E-9	3.73E-9	4.15E-9	1.47E-8	2.12E-8	1.73E-8
	CPU	16345.13	19453.74	17546.56	9473.184	12452.63	13659.00
	Iter	47	49	47	28	41	45
0.6	gap	6.13E-9	5.73E-9	6.13E-9	2.01E-8	1.49E-8	6.13E-9
	CPU	11234.91	12876.45	11387.67	5735.29	6345.15	8191.69
	Iter	28	28	28	23	23	29
0.95	gap	4.76E-9	4.76E-9	4.76E-9	6.34E-9	6.34E-9	4.12E-9
	CPU	5489.34	5723.68	5198.72	3812.65	4276.87	5213.56
	Iter	52.1	53.0	52.1	29.0	35.3	40.4
Average							
	CPU	8768.92	9431.91	8872.45	4957.39	5521.92	6415.17

- For Example 7.4, the obtained results by Tables 4–7, show that the new kernel function with $p = 1$ reduces the the the number of iterations and CPU time about 49% and 39% than the kernel function ψ_8 respectively.

TABLE 10. Number of iterations of performing Algorithm 1 on the some test problems from SDPLIB collection.

name problem	m	n		$\psi_{new,1}(t)$	$\psi_6(t)$	$\psi_7(t)$	$\psi_8(t)$
control1	21	15	Iter	57	67	69	63
			CPU	23.54	27.43	28.54	25.54
control2	66	30	Iter	59	59	61	60
			CPU	332.23	345.76	341.34	312.45
hinf1	13	14	Iter	23	23	24	23
			CPU	65.12	64.13	59.14	68.65
hinf2	13	16	Iter	26	26	28	26
			CPU	75.43	74.54	76.12	79.54
hinf10	21	18	Iter	34	35	34	31
			CPU	98.45	102.76	99.67	90.12
hinf11	31	22	Iter	43	45	45	46
			CPU	87.43	98.45	89.34	92.13
hinf12	43	24	Iter	54	55	55	55
			CPU	112.76	126.43	119.76	126.65
hinf13	57	30	Iter	55	56	55	57
			CPU	98.54	106.43	99.65	101.76
arch0	174	355	Iter	61	63	63	63
			CPU	678.76	686.54	694.34	712.12
arch2	174	355	Iter	60	61	61	61
			CPU	534.12	564.54	576.65	556.34
arch4	174	355	Iter	62	65	65	65
			CPU	453.34	499.45	454.98	461.34
arch8	174	355	Iter	63	63	61	64
			CPU	654.45	645.35	601.65	663.23
gpp100	101	100	Iter	35	37	37	37
			CPU	76.54	78.98	77.13	779.45
Average			Iter	48.61	50.38	50.61	50.07
			CPU	253.13	263.13	255.25	259.02

- From Tables 8 and 9, the best results are derived by performing algorithm with the new kernel function ψ_1. Based on this function, the number of iterations and CPU time are reduced about 44% and 43% than the similar results obtained by using the function ψ_6 respectively.
- From Table 10, we conclude that the new kernel function has the better results than the other kernel functions.

8. Concluding remarks

In this paper, a large-update primal-dual IPM based on a new family of kernel functions is presented for solving SDO problems. Our method performs in a large neighborhood of the central path. Under some mild conditions and by using two crucial properties of the kernel function, i.e., the e-convexity and superconvexity properties, we show that the large update IPMs for SDO problems based on the new proposed kernel functions enjoy the iteration bound as $O\left(pn^{\frac{p+1}{2p}}\log\frac{n}{\varepsilon}\right)$, for $p \geq 1$, in the worst case. Letting $p = O(\log n)$ implies that the worst case iteration bound is $O\left(\sqrt{n}\log n\log\frac{n}{\varepsilon}\right)$, which matches with the so far best-known iteration bound for large-update IPMs. The numerical results show that the new proposed kernel function is well promising in practice in comparison with some other considered kernel functions.

References

[1] E. De Klerk, Aspects of Semidefinite Programming, Interior Point Algorithms and Selected Applications, Dordrecht, Kluwer Academic, 2002.

[2] N.K. Karmarkar, A new polynomial-time algorithm for linear programming, Combinatorica, 4 (1984), 373-395.

[3] Y.E. Nesterov, A.S. Nemirovskii, Interior-Point Polynomial Algorithms in Convex Programming, Volume 13 of Studies in Applied and Numerical Mathematics. SIAM, Philadelphia, 1994.

[4] Y.E. Nesterov, M.J. Todd, Primal-dual interior-point methods for self-scaled cone, SIAM J. Optim. 8 (1998), 324-364.

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

[6] M.R. Peyghami, An interior-point approach for semidefinite optimization using new proximity functions, Asia-Pacific J. Oper. Res. 26 (2009), 365-382.

[7] Z.G. Qian, Y.Q. Bai, G.Q. Wang, Complexity analysis of interior-point algorithm based on a new kernel function for semidefinite optimization, J. Shanghai Univ. Engl. Ed. 12 (2008), 388-394.

[8] G.Q. Wang, Y.Q. Bai, A class of polynomial primal-dual interior-point algorithms for semidefinite optimization, J. Shanghai Univ. Engl. Ed. 10 (2006), 198-207.

[9] G.Q. Wang, Y.Q. Bai, C. Roos, Primal-dual interior-point algorithm for semidefinite optimization based on a simple kernel function, J. Math. Model. Algorithms 4 (2005), 409-433.

[10] Y.Q. Bai, M. El Ghami, C. Roos, A comparative study of kernel functions for primal-dual interior-point algorithms in linear optimization, SIAM J. Optim. 15 (2004), 101-128.

[11] M. El Ghami, Z.A. Guennoun, S. Boula, T. Steihaug, Interior-point methods for linear optimization based on a kernel function with a trigonometric barrier term, J. Comput. Appl. Math. 236 (2012), 3613-3623.

[12] B. Kheirfam, Primal-dual interior-point algorithm for semidefinite optimization based on a new kernel function with trigonometric barrier term, Numer. Algorithms 61 (2012), 659-680.

[13] M. El Ghami, Primal-dual algorithms for $P_*(\kappa)$ linear complementarity problems based on a kernel-function with trigonometric barrier term, Optim. Theory Decis Mak Oper. Res. Appl. 31 (2013), 331–349.

[14] M. Bouafia, D. Benterki, A. Yassine, An efficient primal-dual interior-point method for linear programming problems based on a new kernel function with a trigonometric barrier term, J. Optim. Theory Appl. 170 (2016), 528-545.

[15] X. Li, M. Zhang, Interior-point algorithm for linear optimization based on a new trigonometric kernel function, Oper. Res. Lett. 43 (2015), 471-475.

[16] M.R. Peyghami, S. Fathi-Hafshejani, S. Chen, A prima dual interior-point method for semidefinite optimization based on a class of trigonometric barrier functions, Oper Res Lett. 44 (2016), 319-323.

[17] M.R. Peyghami, S. Fathi Hafshejani, Complexity analysis of an interior-point algorithm for linear optimization based on a new poriximity function, Numer. Algorithms 67 (2014), 33-48.

[18] M.R. Peyghami, S. Fathi Hafshejani, L. Shirvani, Complexity of interior-point methods for linear optimization based on a new trigonometric kernel function, J. Comput. Appl. Math. 255 (2014), 74-85.

[19] S. Fathi-Hafshejani, M. Fatemi, M.R. Peyghami, An interior-point method for $P_*(\kappa)$-linear complementarity problem based on a trigonometric kernel function, J. Appl. Math. Comput. 48 (2015), 111-128.

[20] S. Fathi-Hafshejani, H. Mansouri, M.R. Peyghami, A large-update primal-dual interior-point algorithm for second-order cone optimization based on a new proximity function, Optimization, 65 (2016), 1477-1496.

[21] M. El Ghami, Primal-dual algorithm for semidefinit optimization problems based on generalized trigonometric barrier function, Int. J. Pure Appl. Math. 114 (2017), 797-818.

[22] M.R. Peyghami, S. Fathi-Hafshejani, An interior point algorithm for solving convex quadratic semidefinite optimization problems using a new kernel function, Iranian J. Math. Sci. Inform. 12 (2017), 131-152.

[23] S. Fathi-Hafshejani, H. Mansouri, M.R. Peyghami, An interior-point algorithm for $P_*(\kappa)$-linear complementarity problem based on a new trigonometric kernel function, J. Math. Model. 5 (2017), 171-197.

[24] S. Fathi-Hafshejani, A. Fakharzadeh J., M.R. Peyghami, A unified complexity analysis of interior point methods for semidefinite problems based on trigonometric kernel functions, Optimization, 67 (2018), 113-137.

[25] H. Wolkowicz, R. Saigal, L. Vandenberghe, Handbook of Semidefinite Programming: Theory, algorithms and applications, Boston (MA), Kluwer Academic. 2000.

[26] R.A. Horn, C.R. Johnson, Topics in Matrix Analysis, Cambridge, Cambridge University Press, 1991.

[27] J. Peng, C. Roos, T. Terlaky, Self-regular functions and new search directions for linear and semi-definite optimization, Math Program. 93 (2002), 129-171.

[28] C. Roos, T. Terlaky, J-P. Vial, Theory and Algorithms for Linear Optimization: An interior-point Approach, New York NY, Springer, 2005.

[29] J. Peng, New design and analysis of interior-point Methods, PhD Thesis, Universal Press, 2001.

[30] M.W. Zhang, A large-update interior-point algorithm for convex quadratic semi-definite optimization based on a new kernel function, Acta Math Sinica Engl. Ser. 28 (2012), 2313-2328.

[31] Y.Q. Bai, M. El Ghami, C. Roos, A new efficient large-update primal-dual interior-point methods based on a finite barrier, SIAM J. Optim. 13 (2003), 766-782.

[32] K. Amini, M.R. Peyghami, An interior-point algorithm for linear optimization based on a new kernel function, South East Asian Bull. Math. 29 (2005), 651-667.

[33] I. Touil, D. Benterki, A. Yassine, A feasible primal-dual interior-point method for linear semidefinite programming, J. Comput. Appl. Math. 312 (2017), 216-230.

[34] J.P. Crouzeix, B. Merikhi, A logarithm barrier method for semidefinite programming, RAIRO-Oper Res. 42 (2008), 123-139

[35] B. Borchers, SDPLIB 1.2, A library of semidefinite programming test problems, Optim Methods Softw. 11 (1999), 683-690.