SMALL FLAT SIMPLICES

ANTOINE DEZA

McMaster University, Hamilton, Ontario, Canada
e-mail: deza@mcmaster.ca

LIONEL POURNIN

Université Paris 13, Villetaneuse, France
e-mail: lionel.pournin@univ-paris13.fr

ABSTRACT

Let $\varepsilon_i(d, k)$ be the smallest possible distance between an i-dimensional face P of a d-dimensional lattice simplex contained in $[0, k]^d$ and the opposite $(d-1-i)$-dimensional face Q of that simplex. In this note, we improve an algorithm that computes $\varepsilon_i(d, k)$. This allows to determine and report new values of this quantity for small d and k. Related quantities are reported as well such as the number $\varepsilon_i^u(d, k)$ obtained by replacing the distance between P and Q in the definition of $\varepsilon_i(d, k)$ by the distance of their affine hulls and of the smallest possible distance $\varepsilon(d, k)$ between two disjoint lattice polytopes contained in $[0, k]^d$.

1. Introduction

Consider two positive integers d and k. A lattice (d, k)-polytope is a polytope P contained in the hypercube $[0, k]^d$ such that all the vertices of P have integer coordinates. In the special case when P is a simplex, we will refer to it as a lattice (d, k)-simplex. Now consider an integer i such that $0 \leq i \leq (d-1)/2$. We say that two proper faces of a simplex S are *opposite* when their vertex sets form a partition of the vertex set of S. We will study the smallest possible distance $\varepsilon_i(d, k)$ between an i-dimensional face of a d-dimensional lattice (d, k)-simplex and the opposite $(d-1-i)$-dimensional face of that simplex. As remarked in [7, Section 2], the smallest possible distance $\varepsilon(d, k)$ between two disjoint

d	k					
	2	3	4	5	6	7
4	$\sqrt{209}$	$\sqrt{3022}$	$\mathbf{17\sqrt{65}}$	$\mathbf{\sqrt{81154}}$	$\mathbf{\sqrt{252433}}$	$\mathbf{2\sqrt{169219}}$
5	$\mathbf{\sqrt{2109}}$					

Table 1. The known values of $1/\varepsilon_0(d,k)$ when $d \geq 4$ and $k \geq 2$.

lattice (d,k)-polytopes can be expressed as

$$\varepsilon(d,k) = \min\left\{\varepsilon_i(d,k) : 0 \leq i \leq \frac{d-1}{2}\right\}. \tag{1}$$

This quantity appears when bounding the complexity of von Neumann's alternating projections algorithm [3, 13] and is related to several notions that arise in optimization [2, 8, 9, 10, 11, 12].

Explicit bounds on $\varepsilon(d,k)$ and $\varepsilon_i(d,k)$ in terms of d and k are given in [6, 7] and exact values are provided in [4, 5, 6, 7] when d or k are small. More precisely it is known that $\varepsilon_0(d,1)$ is always equal to $1/\sqrt{d}$ and that

$$\left\{\begin{array}{l}\varepsilon(2,k) = \varepsilon_0(2,k) = \dfrac{1}{\sqrt{(k-1)^2+k^2}},\\[2ex] \varepsilon_0(3,k) = \dfrac{1}{\sqrt{3k^4-4k^3+4k^2-2k+1}},\end{array}\right.$$

when k is at least 2. Moreover, $\varepsilon(3,3) = \varepsilon_1(3,3) = 1/\sqrt{299}$ while

$$\varepsilon(3,k) = \varepsilon_1(3,k) = \frac{1}{\sqrt{2(2k^2-4k+5)(2k^2-2k+1)}}$$

when k is not equal to 3. Therefore, all the values of $\varepsilon(d,k)$ and $\varepsilon_i(d,k)$ are known exactly when d is equal to 2 or 3. When d is at least 4 only a handful

d	k					
	1	2	3	4	5	6
4	$3\sqrt{2}$	$2\sqrt{113}$	$11\sqrt{71}$	$\mathbf{3\sqrt{4710}}$	$\mathbf{29\sqrt{218}}$	$\mathbf{\sqrt{526055}}$
5	$\sqrt{58}$	$\mathbf{\sqrt{5671}}$				
6	$\sqrt{202}$					
7	$\mathbf{\sqrt{746}}$					

Table 2. The known values of $1/\varepsilon_1(d,k)$ when $d \geq 4$.

d	k = 1	2
5	$\sqrt{55}$	$\boldsymbol{\sqrt{6971}}$
6	$\sqrt{199}$	
7	$\boldsymbol{\sqrt{991}}$	

Table 3. The known values of $1/\varepsilon_2(d,k)$.

of values of $\varepsilon_i(d,k)$ are known other than $\varepsilon_0(d,1)$. A few values have been computed in [6] using an exhaustive enumeration strategy and the other ones in [5, 7] using an algorithm that exploits an algebraic formulation that allows to factor certain symmetries of the problem.

We show how the algorithm from [5] can be improved as a consequence of several results from [7] in order to compute further values of $\varepsilon_i(d,k)$ and $\varepsilon(d,k)$. The new values of $\varepsilon_i(d,k)$ we obtain this way are the bolded entries in Tables 1, 2, and 3. This allows to determine four new values of $\varepsilon(d,k)$.

Theorem 1.1: *The following four equalities hold.*

$$\varepsilon(4,4)=\frac{1}{3\sqrt{4710}},\ \varepsilon(4,5)=\frac{1}{29\sqrt{218}},\ \varepsilon(4,6)=\frac{1}{\sqrt{526055}},\ \varepsilon(5,2)=\frac{1}{\sqrt{6971}}.$$

These values $\varepsilon(d,k)$ correspond to the bolded entries in Table 4.

Our algorithm also allows to compute a quantity related to $\varepsilon_i(d,k)$ that can be defined by considering the distance of the affine hull of P and Q rather than the distance of P and Q: we denote by $\varepsilon_i^u(d,k)$ the smallest possible distance between the affine hull of a i-dimensional face of a d-dimensional lattice (d,k)-simplex and the affine hull of the opposite $(d-1-i)$-dimensional face of that simplex. Here we use the exponent u to mark unboundedness. When d is

d	k = 1	2	3	4	5	6
4	$3\sqrt{2}$	$2\sqrt{113}$	$11\sqrt{71}$	$\boldsymbol{3\sqrt{4710}}$	$\boldsymbol{29\sqrt{218}}$	$\boldsymbol{\sqrt{526055}}$
5	$\sqrt{58}$	$\boldsymbol{\sqrt{6971}}$				
6	$\sqrt{202}$					

Table 4. The known values of $1/\varepsilon(d,k)$ when $d \geq 4$.

d	k						
	1	2	3	4	5	6	7
4	$\sqrt{7}$	$\sqrt{209}$	$\sqrt{3142}$	$\sqrt{\mathbf{20441}}$	$\sqrt{\mathbf{85346}}$	$\sqrt{\mathbf{270937}}$	$\sqrt{\mathbf{714134}}$
5	$\sqrt{19}$	$\sqrt{\mathbf{2329}}$					
6	$\sqrt{59}$						
7	$\sqrt{\mathbf{203}}$						

Table 5. The known values of $1/\varepsilon_0^u(d,k)$ when $d \geq 4$.

equal to 2 or 3, $\varepsilon_i^u(d,k)$ always coincides with $\varepsilon_i(d,k)$ [4, 5, 7] and is therefore completely determined. However, when d is at least 4 none of the known values of $\varepsilon_0^u(d,k)$ are equal to $\varepsilon_0(d,k)$ except when d is equal to 4 and k is equal to 2. The known values of $\varepsilon_0^u(d,k)$ when d is at least 4 are reported in Table 5 where the bolded are obtained using the improved algorithm that we present here while the non-bolded ones have been obtained computationally in [7].

When i is at least 1, all the values of $\varepsilon_i^u(d,k)$ known so far coincide with the corresponding $\varepsilon_i(d,k)$ except for $\varepsilon_1^u(5,2)$. In particular, the known values of $\varepsilon_2^u(d,k)$ when d is at least 4 can be obtained from Table 3. We report the known values of $\varepsilon_1^u(d,k)$ when d is at least 4 in Table 6 but it should be noted that in this table, only $\varepsilon_1^u(5,2)$ differs from the entries of Table 2.

In Section 2, we recall the algebraic model for $\varepsilon_i(d,k)$ and $\varepsilon_i^u(d,k)$ that has been introduced in [5] for the computation of $\varepsilon(d,k)$ and adapted to these two quantities in [7]. Based on this model, we explain in Section 3 how the algorithm from [5] can be adapted and improved thanks to the properties of $\varepsilon_i(d,k)$ and

d	k					
	1	2	3	4	5	6
4	$3\sqrt{2}$	$2\sqrt{113}$	$11\sqrt{71}$	$\mathbf{3}\sqrt{\mathbf{4710}}$	$\mathbf{29}\sqrt{\mathbf{218}}$	$\sqrt{\mathbf{526055}}$
5	$\sqrt{58}$	$\sqrt{\mathbf{6042}}$				
6	$\sqrt{202}$					
7	$\sqrt{\mathbf{746}}$					

Table 6. The known values of $1/\varepsilon_1^u(d,k)$ when $d \geq 4$.

$\varepsilon_i^u(d,k)$ established in [7]. Finally, in Section 4, we discuss some of the pairs of lattice simplices achieving $\varepsilon_i(d,k)$ and $\varepsilon_i^u(d,k)$ that our algorithm outputs.

2. The algebraic model from [7] for $\varepsilon_i^u(d,k)$ and $\varepsilon_i(d,k)$

An algebraic model for the computation of $\varepsilon(d,k)$ is given in [5]. This model has been adapted to the computation of $\varepsilon_i(d,k)$ and $\varepsilon_i^u(d,k)$ in [7]. We recall the later model as it will be the basis on our improved algorithm. Consider two opposite faces P and Q of a d-dimensional lattice (d,k)-simplex such that P has dimension i. Let p^0 to p^i and q^0 to q^{d-i-1} denote the vertices of P and Q. Consider the $d\times(d-1)$ matrix A whose first i column vectors are p^1-p^0 to p^i-p^0 and whose remaining $d-i-1$ column vectors are q^1-q^0 to $q^{d-i-1}-q^0$:

$$A=\left[\begin{array}{ccccccc} p_1^1-p_1^0 & \cdots & p_1^i-p_1^0 & q_1^1-q_1^0 & \cdots & q_1^{d-i-1}-q_1^0 \\ \vdots & & \vdots & \vdots & & \vdots \\ p_d^1-p_d^0 & \cdots & p_d^i-p_d^0 & q_d^1-q_d^0 & \cdots & q_d^{d-i-1}-q_d^0 \end{array}\right]. \tag{2}$$

Observe that if i is equal to 0, then the columns of A are q^1-q^0 to $q^{d-i-1}-q^0$ and in particular, A does not depend on P in that case. It is not hard to see that, as P and Q being opposite faces of a d-dimensional lattice (d,k)-simplex, A^tA is non-singular (see [7]). In fact, by the Cauchy–Binet formula, the determinant of A^tA is positive (see Equation (10) in [7]). Further denote

$$b=q^0-p^0. \tag{3}$$

It is shown in [5] that

$$d\big(\mathrm{aff}(P),\mathrm{aff}(Q)\big)=\|A(A^tA)^{-1}A^tb-b\| \tag{4}$$

which is refined in [7, Section 3] into

$$d\big(\mathrm{aff}(P),\mathrm{aff}(Q)\big)=\sqrt{\frac{\|b\|^2\det(A^tA)-b^tAC^tA^tb}{\det(A^tA)}} \tag{5}$$

where C is the cofactor matrix of A^tA. We shall denote by $A|b$ the $d\times d$ matrix obtained by appending b as an additional column on the right of A. Consider the set $\mathcal{A}_i(d,k)$ of all the augmented matrices $A|b$ obtained via (2) and (3) from an i-dimensional face P of a d-dimensional lattice (d,k)-simplex and its opposite $(d-i-1)$-dimensional face Q. As some matrix in $\mathcal{A}_i(d,k)$ arises from two faces whose distance of affine hulls is $\varepsilon_i^u(d,k)$, we get the following.

LEMMA 2.1: *For any positive d and k and any i such that $0 \leq i \leq (d-1)/2$,*

$$\varepsilon_i^u(d,k) = \min\left\{ \|A(A^tA)^{-1}A^tb - b\| \right\}$$

where the minimum is over all the augmented matrices $A|b$ in $\mathcal{A}_i(d,k)$.

Lemma 2.1 makes it possible compute $\varepsilon_i^u(d,k)$ by enumerating $\mathcal{A}_i(d,k)$ up to symmetry. By [7, Corollary 2.3] and [5, Remark 2], the same strategy turns out to work for $\varepsilon_i(d,k)$ as well. Indeed, let P denote an i-dimensional face of some d-dimensional lattice (d,k)-simplex and Q denote the $(d-i-1)$-dimensional face of that simplex opposite P. According to [7, Corollary 2.3] if the distance between P and Q is equal to $\varepsilon_i(d,k)$, then that distance is necessarily equal to the distance between the affine hulls of P and Q. Moreover, by [5, Remark 2], it can be checked using just the augmented matrix $A|b$ associated to P and Q via (2) and (3) whether the distance between P and Q coincides with the distance of their affine hulls: this is the case if and only if the first i coordinates of the vector $(A^tA)^{-1}A^tb$ are non-negative and sum to at most 1 while its last $d-i-1$ coordinates are non-positive and sum to at least -1. We obtain the following analogue of Lemma 2.1 for $\varepsilon_i(d,k)$ as a consequence.

LEMMA 2.2: *For any positive d and k and any i such that $0 \leq i \leq (d-1)/2$,*

$$\varepsilon_i(d,k) = \min\left\{ \|A(A^tA)^{-1}A^tb - b\| \right\}$$

where the minimum is over the augmented matrices $A|b$ in $\mathcal{A}_i(d,k)$ such that

(i) *the first i coordinates of the vector $(A^tA)^{-1}A^tb$ are all non-negative and the sum of these coordinates is at most 1 while*

(ii) *the last $d-i-1$ coordinates of the vector $(A^tA)^{-1}A^tb$ are all non-positive and the sum of these coordinates is at least -1.*

3. An improved algorithm

Lemmas 2.1 and 2.2 reduce the computation of $\varepsilon_i^u(d,k)$ and $\varepsilon_i(d,k)$ to evaluating a formula over all the augmented matrices $A|b$ in $\mathcal{A}_i(d,k)$ and to checking the signs and two partial sums of the coordinates to the vector $(A^tA)^{-1}A^tb$. In fact, not all matrices of $\mathcal{A}_i(d,k)$ need to be considered in that computation. Indeed, given an i-dimensional face P of some d-dimensional lattice (d,k)-simplex and the opposite $(d-i-1)$-dimensional face Q, many different matrices $A|b$ can be obtained via (2) and (3) depending on the way the vertices of P and Q are labeled. Moreover, applying an isometry of $[0,k]^d$ to P and Q will not change their distance or the distance of their affine hulls. The strategy of the algorithm presented in [5, Section 4] is to take advantage of these symmetries by building a list $\mathcal{L}$ of the possible rows of an augmented matrix $A|b$ from $\mathcal{A}_i(d,k)$. By construction an element of $\mathcal{L}$ is a row vector of the form

$$(x_1-x_0,\ldots,x_i-x_0,y_1-y_0,\ldots,y_{d-i-1}-y_0,y_0-x_0)$$

where x and y are lattice points in $[0,k]^i$ and $[0,k]^{d-i-1}$, respectively, while x_0 and y_0 are two integers in $\{0,\ldots,k\}$. A first reduction is provided [5, Proposition 6] whereby, for any augmented matrix $A|b$ in $\mathcal{A}_i(d,k)$,

$$\|A(A^tA)^{-1}A^tb-b\|\geq\|\overline{A}(\overline{A}^t\overline{A})^{-1}\overline{A}^t\overline{b}-\overline{b}\|>0$$

where $\overline{A}|\overline{b}$ denotes an augmented matrix obtained by dividing each row of $A|b$ by its greatest common divisor and by negating a subset of these rows. Hence, when several rows in $\mathcal{L}$ are multiple of one another, we only need to keep in $\mathcal{L}$ the one whose first non-zero coordinate is positive and whose coordinates are relatively prime. Since the rows of an augmented matrix $A|b$ in $\mathcal{A}_i(d,k)$ are distinct (otherwise A^tA would be singular), we only need to consider the

$$\binom{|\mathcal{L}|}{d}$$

matrices $A|b$ obtained from d different rows in $\mathcal{L}$. By construction, the matrices obtained this way will exhaust, up to symmetry, the augmented matrices from $\mathcal{A}_i(d,k)$. Note that some of the augmented matrix $A|b$ built this way from $\mathcal{L}$ may not belong to $\mathcal{A}_i(d,k)$ because for some of them A^tA may be singular or the norm of $A(A^tA)^{-1}A^tb-b$ may be equal to 0.

The algorithm from [5, Section 4] consists in building $\mathcal{L}$ taking advantage of the reduction provided by [5, Proposition 6] and then in considering all the augmented matrices $A|b$ formed from d different rows in $\mathcal{L}$. The smallest non-zero value of $\|A(A^tA)^{-1}A^tb-b\|$ over all of the obtained matrices for which A^tA is non-singular is then computed and output which, by Lemma 2.1 provides $\varepsilon_i^u(d,k)$. The efficiency of this algorithm can be significantly improved thanks to the expression of $\|A(A^tA)^{-1}A^tb-b\|$ obtained by using (4) and (5) jointly. Indeed, denote by ε the smallest non-zero value of $\|A(A^tA)^{-1}A^tb-b\|$ obtained before some augmented matrix $A|b$ is considered by the algorithm. According to (4) and (5), if $\|A(A^tA)^{-1}A^tb - b\|$ is non-zero, then

$$\|A(A^tA)^{-1}A^tb - b\| \geq \frac{1}{\sqrt{\det(A^tA)}} \tag{6}$$

because the numerator of the fraction under the square root in the right-hand side of (5) is an integer. As a consequence, if the right hand side of (6) is at least ε, or equivalently if the determinant of A^tA is at most $1/\varepsilon^2$, then the value of $\|A(A^tA)^{-1}A^tb-b\|$ is either zero or at least ε. Since comparing the determinant of A^tA with $1/\varepsilon^2$ can be done without using b, this allows the algorithm to discard together all of the augmented matrices that only differ from $A|b$ in the last column. Therefore our revised strategy is the following: once the list $\mathcal{L}$ of all the possible rows of $A|b$ has been built and reduced using [5, Proposition 6],

---

Algorithm 1: Computation of $\varepsilon_i^u(d,k)$ and $\varepsilon_i(d,k)$

Compute $\mathcal{L}$ and then $\mathcal{A}$ for the given values of i, d, and k
$\varepsilon^u \leftarrow 1$ and $\varepsilon \leftarrow 1$
for *each matrix A formed of d distinct rows from $\mathcal{A}$* **do**
    if $\det(A^tA) > 1/\varepsilon^2$ **then**
        for *each vector b such that $A|b$ is formed of d rows from $\mathcal{L}$* **do**
            $\delta \leftarrow \|A(A^tA)^{-1}A^tb - b\|$
            if *δ is positive* **then**
                if *δ is less than ε^u* **then**
                    $\varepsilon^u \leftarrow \delta$
                end
                if *$A|b$ satisfies Assertions (i) and (ii) in Lemma 2.2* **then**
                    if *δ is less than ε* **then**
                        $\varepsilon \leftarrow \delta$
                    end
                end
            end
        end
    end
end
$\varepsilon_i^u(d,k) \leftarrow \varepsilon^u$ and $\varepsilon_i(d,k) \leftarrow \varepsilon$

---

we remove the last coordinate of every row vector in $\mathcal{L}$ in order to build the list $\mathcal{A}$ of all the possible row vectors of the matrix A. Then, we consider all the matrices A formed from d different rows in $\mathcal{A}$ in a sequence. Only if the determinant of A^tA is greater than the smallest value of $1/\|A(A^tA)^{-1}A^tb-b\|^2$ obtained so far do we consider the vectors b such that the augmented matrix $A|b$ is formed from d rows in $\mathcal{L}$. In practice the number of augmented matrices $A|b$ that correspond to a matrix A is exponential in the dimension. For instance, when computing $\varepsilon_2^u(5,2)$ and $\varepsilon_2(5,2)$, the number of rows in $\mathcal{L}$ is equal to 300 while the number of rows in $\mathcal{A}$ is 162. Hence, instead of

$$\binom{300}{5} = 19\,582\,837\,560$$

augmented matrices $A|b$, the algorithm only reviews

$$\binom{162}{5} = 873\,642\,672$$

matrices A and for all but a few of them, it does not need to consider the corresponding augmented matrices $A|b$ because the determinant of A^tA is not large enough. The substantial speedup allows to compute all the new values of $\varepsilon_i^u(d,k)$ and $\varepsilon_i(d,k)$ reported in the introduction. This results in Algorithm 1 and, by the above discussion we have the following.

PROPOSITION 3.1: *Algorithm 1 correctly computes $\varepsilon_i^u(d,k)$ and $\varepsilon_i(d,k)$.*

In the proposed implementation, ε^u and ε are the two variables that will be updated as the algorithm proceeds to eventually reach $\varepsilon_i^u(d,k)$ and $\varepsilon_i(d,k)$, respectively. During the execution, these two variables can only decrease (in Lines 9 and 13) and they are initially set to 1 in Line 1, as

$$\varepsilon_i^u(d,k) \leq \varepsilon_i(d,k) \leq 1.$$

Note that in Line 5 of Algorithm 1, there is no need to actually look at $\mathcal{L}$ as in practice, for each row l contained in $\mathcal{A}$, we store the set $\mathcal{B}_l$ of the integers r such that the row vector $l|r$ belongs to $\mathcal{L}$. The possible vectors b such that $A|b$ is formed from d rows vectors in $\mathcal{L}$ are then generated as the elements of the cartesian product of the sets $\mathcal{B}_l$ when l ranges over the rows of A.

Finally, let us observe that the test in Line 4 of Algorithm 1 does not result in a significant improvement when computing $\varepsilon_0^u(d,1)$. Indeed, in that case, it is known that $\varepsilon_0(d,1)$ is equal to $1/\sqrt{d}$ for all d (see [6, Lemma 2.4]) while

$$\varepsilon_0^u(d,1) \leq \frac{4^{d+o(d)}}{\sqrt{d}^d}$$

as d goes to infinity (see [1, Theorem 3.2.2]). In fact, Table 5 shows that in low dimensions, $\varepsilon_0^u(d,1)$ is already significantly smaller than $\varepsilon_0(d,1)$. Since computing $\varepsilon_0(d,1)$ is not needed, one can determine $\varepsilon_0^u(d,1)$ faster by replacing ε with ε^u in Line 4 of Algorithm 1 and by removing Lines 11 to 15.

4. Some optimal configurations

When $2 \leq k \leq 7$, $\varepsilon_0^u(4,k)$ is achieved as the distance of the point P and the affine hull of the tetrahedron Q, where

$$P = \left[\begin{array}{c} 1 \\ 1 \\ 2 \\ 2 \end{array}\right] \text{ and } Q = \text{conv}\left\{\left[\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \end{array}\right], \left[\begin{array}{c} k \\ 0 \\ k \\ k-1 \end{array}\right], \left[\begin{array}{c} 0 \\ k \\ k-1 \\ k \end{array}\right], \left[\begin{array}{c} k-1 \\ k \\ 0 \\ 0 \end{array}\right]\right\}$$

which yields

$$d\big(P, \text{aff}(Q)\big) = \frac{1}{\sqrt{8k^6 - 16k^5 + 20k^4 - 20k^3 + 15k^2 - 6k + 1}}.$$

In contrast, the pairs P and Q that our algorithm finds whose distance achieves $\varepsilon_0(4,k)$ do not show such an obvious behavior. Indeed, $\varepsilon_0(4,4)$ is achieved as the distance between P and Q where

$$P = \left[\begin{array}{c} 1 \\ 2 \\ 3 \\ 3 \end{array}\right] \text{ and } Q = \text{conv}\left\{\left[\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \end{array}\right], \left[\begin{array}{c} 4 \\ 4 \\ 4 \\ 1 \end{array}\right], \left[\begin{array}{c} 1 \\ 4 \\ 0 \\ 4 \end{array}\right], \left[\begin{array}{c} 0 \\ 1 \\ 4 \\ 4 \end{array}\right]\right\},$$

$\varepsilon_0(4,5)$ as the distance between P and Q where

$$P = \left[\begin{array}{c} 2 \\ 2 \\ 3 \\ 4 \end{array}\right] \text{ and } Q = \text{conv}\left\{\left[\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \end{array}\right], \left[\begin{array}{c} 5 \\ 5 \\ 4 \\ 0 \end{array}\right], \left[\begin{array}{c} 0 \\ 3 \\ 5 \\ 5 \end{array}\right], \left[\begin{array}{c} 5 \\ 0 \\ 0 \\ 4 \end{array}\right]\right\},$$

$\varepsilon_0(4,6)$ as the distance between P and Q where

$$P=\begin{bmatrix}1\\1\\1\\2\end{bmatrix} \text{ and } Q=\operatorname{conv}\left\{\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}6\\6\\5\\0\end{bmatrix},\begin{bmatrix}0\\6\\6\\5\end{bmatrix},\begin{bmatrix}5\\0\\0\\6\end{bmatrix}\right\},$$

and $\varepsilon_0(4,7)$ as the distance between P and Q where

$$P=\begin{bmatrix}2\\3\\4\\4\end{bmatrix} \text{ and } Q=\operatorname{conv}\left\{\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}7\\7\\6\\0\end{bmatrix},\begin{bmatrix}0\\2\\7\\7\end{bmatrix},\begin{bmatrix}6\\7\\0\\7\end{bmatrix}\right\}.$$

Likewise, the lattice segment P and the lattice triangle Q that our algorithm finds whose distance achieves $\varepsilon_1(4,k)$ do not show an obvious behavior either. Indeed, $\varepsilon_1(4,4)$ is achieved as the distance between P and Q where

$$P=\operatorname{conv}\left\{\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}3\\4\\4\\4\end{bmatrix}\right\} \text{ and } Q=\operatorname{conv}\left\{\begin{bmatrix}4\\4\\3\\0\end{bmatrix},\begin{bmatrix}0\\3\\4\\3\end{bmatrix},\begin{bmatrix}4\\1\\0\\4\end{bmatrix}\right\},$$

$\varepsilon_1(4,5)$ is achieved as the distance between P and Q where

$$P=\operatorname{conv}\left\{\begin{bmatrix}0\\1\\0\\0\end{bmatrix},\begin{bmatrix}3\\5\\5\\5\end{bmatrix}\right\} \text{ and } Q=\operatorname{conv}\left\{\begin{bmatrix}0\\0\\1\\5\end{bmatrix},\begin{bmatrix}5\\5\\0\\1\end{bmatrix},\begin{bmatrix}0\\4\\5\\0\end{bmatrix}\right\},$$

and $\varepsilon_1(4,6)$ as the distance between P and Q where

$$P=\operatorname{conv}\left\{\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}5\\6\\6\\6\end{bmatrix}\right\} \text{ and } Q=\operatorname{conv}\left\{\begin{bmatrix}0\\0\\5\\6\end{bmatrix},\begin{bmatrix}6\\1\\0\\0\end{bmatrix},\begin{bmatrix}0\\6\\2\\1\end{bmatrix}\right\}.$$

When d is at least 5, we will not report the pairs of simplices that the algorithm provides whose distance achieves $\varepsilon_i(d,k)$ or $\varepsilon_i^u(d,k)$. However, let us remark that all the known values of $\varepsilon_0(d,2)$ are achieved as the distance between a $(d-1)$-dimensional lattice simplex and the lattice point at the center of the hypercube $[0,2]^d$, which may be true for all d.

References

[1] Noga Alon and Văn H. Vũ, *Anti-Hadamard matrices, coin weighing, threshold gates, and indecomposable hypergraphs*, Journal of Combinatorial Theory, Series A **79** (1997), 133–160.

[2] Amir Beck and Shimrit Shtern, *Linearly convergent away-step conditional gradient for non-strongly convex functions*, Mathematical Programming **164** (2017), 1–27.

[3] Gábor Braun, Sebastian Pokutta and Robert Weismantel, *Alternating linear minimization: revisiting von Neumann's alternating projections*, arXiv:2212.02933 (2022).

[4] Antoine Deza, Zhongyuan Liu and Lionel Pournin, *Kissing polytopes in dimension* 3, Experimental Mathematics, *to appear*.

[5] Antoine Deza, Zhongyuan Liu and Lionel Pournin, *Small kissing polytopes*, Vietnam Journal of Mathematics **53** (2025), no. 4, 901–913.

[6] Antoine Deza, Shmuel Onn, Sebastian Pokutta, and Lionel Pournin, *Kissing polytopes*, SIAM Journal on Discrete Mathematics **38** (2024), no. 4, 2643–2664.

[7] Antoine Deza and Lionel Pournin, *Flat simplices and kissing polytopes*, arXiv:2601.03183 (2026).

[8] David H. Gutman and Javier F. Peña, *The condition number of a function relative to a set*, Mathematical Programming **188** (2021), 255–294.

[9] Simon Lacoste-Julien and Martin Jaggi, *On the global linear convergence of Frank–Wolfe optimization variants*, Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS), 2015, pp. 496–504.

[10] Javier F. Peña, *Generalized conditional subgradient and generalized mirror descent: duality, convergence, and symmetry*, arXiv:1903.00459 (2019).

[11] Javier F. Peña and Daniel Rodríguez, *Polytope conditioning and linear convergence of the Frank–Wolfe algorithm*, Mathematics of Operations Research **44** (2019), no. 1, 1–18.

[12] Luis Rademacher and Chang Shu, *The smoothed complexity of Frank–Wolfe methods via conditioning of random matrices and polytopes*, Mathematical Statistics and Learning **5** (2022), 273–310.

[13] John von Neumann, *On rings of operators. Reduction theory*, Annals of Mathematics **50** (1949), no. 2, 401–485.