New Korkin-Zoloratev inequalities : implementation and numerical data

New Korkin–Zolotarev Inequalities: Implementation and Numerical Data

Stefan H. M. van Zwam*

May 8, 2006

Abstract

This technical report discusses the mathematical details and the implementation of the methods discussed in the accompanying paper [PZ06]. In particular a method to find a finite list of inequalities that certify Korkin–Zolotarev reducedness of a quadratic form is presented. Moreover a semidefinite programming relaxation of the space of KZ-reduced quadratic forms is described in detail, together with a branching strategy to optimize over this space. Finally the implementation of these methods is discussed, together with some hints on how to compile and use the programs.
The two digital appendices, which can be obtained from the SPOR reports website†, contain an implementation of the methods discussed and numerical data that prove the theorems in [PZ06].

Keywords: Lattice, quadratic form, semidefinite programming, optimization, Korkin–Zolotarev reduction, Hermite's constant.

*E-mail: svzwam@tue.nl
†http://www.win.tue.nl/bs/spor/

Contents

Chapter 1

Overview

This technical report serves as an accompaniment to the paper *New Korkin–Zolotarev Inequalities* [PZ06]*. It details some of the techniques described in that paper, provides some of the numerical data, and shows how to verify the proofs of the theorems.

We start our discussion with the mathematical underpinnings. In Chapter 2 we describe a detailed, constructive proof of the finiteness of the set of constraints needed to verify KZ-reducedness of a quadratic form. After that we show in Chapter 3 how these sets can be employed to create a semidefinite programming relaxation. This SDP is described with considerable detail and closely reflects our implementation. Moreover, the branching techniques used to obtain better lower bounds are described. Chapter 4 goes on to show how to turn the floating-point data provided by the implementation of these methods into mathematically rigorous proofs. This is exactly what was used to prove the lower bounds in the paper. The last two sections then show how to verify these proofs. The verification is very straightforward and does not require anything beyond basic rational arithmetic.

Chapters 5 and 6 serve to link the mathematical results to the implementation. The first contains instructions for getting the code to run and describes what can be found in the various source files; the second describes where the various proofs can be found and finishes with a listing of X_n for $n \leq 5$, together with quadratic forms showing necessity of most of them.

The previous sections hinted already at the existence of source code for computer programs and numerical data that prove the lower bounds. Such information can not be presented conveniently on paper. Hence the addition of two digital appendices. The first of these contains the source code, the second the numerical data. Both can be obtained from the SPOR reports website,

`http://www.win.tue.nl/bs/spor/`

Finally, the author has built a website to which updates to the code and new results will be posted. He invites people who find new inequalities to post them to this site.

`http://www.win.tue.nl/kz/`

*Throughout this report we will refer to [PZ06] as "the paper".

Chapter 2

Finding a sufficient set of inequalities

In the excellent survey [RB79] a constructive proof was given showing that a finite number of inequalities suffices to check if a form is KZ-reduced. The proof was very short, but used some rather rough estimates. In this chapter a refined proof is given, resulting in an algorithm that produces much smaller numbers of inequalities - in fact, for forms of up to 4 variables the number of inequalities found by this method is optimal. Novikova [Nov83] gives lists of inequalities for forms of up to 8 variables. In her paper she claims that up to 5 variables, the sets provided are optimal. Unfortunately, her paper does not contain the proofs. The finiteness proof presented here stems from [Zwa05].

Let $f(\boldsymbol{x}) = \boldsymbol{x}^t B \boldsymbol{x}$ be a quadratic form* in n variables with symmetric matrix B. Its Lagrange expansion is

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} A_i (x_i - \sum_{j=i+1}^{n} \alpha_{ij} x_j)^2$$

and the form is said to be *Korkin–Zolotarev reduced* if $|\alpha_{ij}| \leq 1/2$, $\alpha_{i,i+1} \geq 0$, and for all i, A_i is the minimum of $f_{in}(x_i, \ldots, x_n)$ over all $(x_i, \ldots, x_n) \in \mathbb{Z}^{n-i+1} \setminus \{0\}$, where

$$f_{ij}(x_i, \ldots, x_j) = \sum_{k=i}^{j} A_k (x_k - \sum_{l=k+1}^{j} \alpha_{kl} x_l)^2.$$

Observe that

$$f_{ij}(x_i, \ldots, x_j) = f_{in}(x_i, \ldots, x_j, 0, \ldots, 0). \tag{2.1}$$

The following important theorem was proven by Korkin and Zolotarev in [KZ73]. It gives a lower bound on the size of the A_i in a KZ-reduced expansion.

Theorem 2.1 (First KZ-inequality)**.** *In the Lagrange expansion of a KZ-reduced quadratic form the outer coefficients satisfy*

$$A_{i+1} \geq \frac{3}{4} A_i$$

for all $i \in \{1, \ldots, n-1\}$.

Proof. Let f be a KZ-reduced quadratic form. We can use (2.1) and the fact that A_i is the minimum of $f_{in}(x_i, \ldots, x_n)$ to see that

$$A_i \leq f_{i,i+1}(x_i, x_{i+1}).$$

Substituting $(x_i, x_{i+1}) = (0, 1)$ we get

$$A_i \leq A_i(0 - \alpha_{i,i+1})^2 + A_{i+1} \leq \frac{1}{4} A_i + A_{i+1}$$

which gives the desired result. □

*In the paper a quadratic form is indicated with q; what we call $f_{kn}(\boldsymbol{x})$ is called $q_k(\boldsymbol{x})$ there and f_{km} is denoted by q_k^m. The notation in the paper is somewhat more convenient; the notation in this report is closer to historical literature. Other places where this report's notation deviates from the paper will be indicated by footnotes.

Korkin and Zolotarev also proved the following result.

Theorem 2.2 (Second KZ-inequality)**.** *In the Lagrange expansion of a KZ-reduced quadratic form the outer coefficients satisfy*

$$A_{i+2} \geq \frac{2}{3} A_i$$

for all $i \in \{1, \ldots, n-2\}$.

Using this theorem, we can find a finite characterization of KZ-reduced forms:

Theorem 2.3. *For each* $n > 0$ *there are finite sets of vectors* $X_1, \ldots, X_n$ *such that if a quadratic form* f *satisfies*

$$f \text{ is size-reduced,} \tag{2.2}$$

$$f_{ij}(\boldsymbol{x}) \geq A_i \text{ for all } 1 \leq i < j \leq n, \text{ for all } \boldsymbol{x} \in X_{j-i+1}, \tag{2.3}$$

then f *is KZ-reduced*[†].

Note that if we drop the restriction that the X_i are finite, and choose $X_i = \mathbb{Z}^i$, then (2.2) and (2.3) are equivalent to the definition of KZ-reducedness.

Proof. By induction on n. Clearly $X_1 = \emptyset$ suffices, as $f(\boldsymbol{x}) = A_1 x_1^2$ is always KZ-reduced.

If $n = 2$ then $X_2 = \{(0,1)\}$ suffices: from $f(0,1) \geq A_1$ the first KZ-inequality follows. If $|x_2| > 1$ then $f(\boldsymbol{x}) \geq A_2 x_2^2 \geq 3/4 A_1 2^2 > A_1$. If $x_2 = 1$ then size-reducedness implies $|x_1 - \alpha_{12} x_2| \geq \alpha_{12}$ for all x_1; the same holds if $x_2 = -1$. Hence only $x_1 = 0$ needs to be considered. Finally $f(0,1) = f(0,-1)$, so we need only one of the two.

For $n = 3$ we have $X_3 = \{(0,0,1), (0,1,1), (1,1,1)\}$: as in the previous case we have $|x_3| \leq 1$, so we may assume $x_3 = 1$ for all vectors in X_3. Using KZ-reducedness of f_{23} and f_{12} (so the first KZ-inequality holds) we get

$$f(x_1, x_2, 1) \geq A_2 (x_2 - \alpha_{23} x_3)^2 + A_3 \geq A_1 (\frac{3}{4}(x_2 - \alpha_{23})^2 + \frac{9}{16}),$$

so we need $|x_2 - \alpha_{23}| < \sqrt{7/12}$. This can only be true for some $0 \leq \alpha_{23} \leq 1/2$ if $x_2 \in \{0, 1\}$. If $x_2 = 0$ we have

$$f(x_1, 0, 1) \geq A_1 (x_1 - \alpha_{13})^2 + 3/4 A_1$$

Like above we have $|x_1 - \alpha_{13}| \geq \alpha_{13}$ for all x_1, so we only need to add $(0,0,1)$ to the list. Now if $x_2 = 1$ then

$$f(x_1, 1, 1) \geq A_1 (x_1 - \alpha_{12} - \alpha_{13})^2 + 3/4 A_1.$$

For $x_1 < 0$ we have $|x_1 - \alpha_{12} - \alpha_{13}| \geq 1/2$. The same goes for $x_1 \geq 2$. So we need to verify $(0,1,1)$ and $(1,1,1)$.

For $n > 3$ we must show that if $f_{2,n}$ and $f_{1,n-1}$ are KZ-reduced, then $f(x) \geq A_1$ for all but finitely many $x \in \mathbb{Z}^n$. By (2.1) we can assume $x_n \neq 0$ for all $\boldsymbol{x} \in X_n$. If $x_n > \left((3/2)^{\lfloor (n-1)/2 \rfloor} (4/3)^{(n-1) \bmod 2}\right)^{1/2}$ then

$$f(x_1, \ldots, x_n) \geq A_n x_n^2 \geq (2/3)^{\lfloor (n-1)/2 \rfloor} (3/4)^{(n-1) \bmod 2} A_1 x_n^2 > A_1,$$

where we have used the first and second KZ-inequalities. It follows that we only need to consider $1 \leq |x_n| \leq \left((3/2)^{\lfloor (n-1)/2 \rfloor} (4/3)^{(n-1) \bmod 2}\right)^{1/2}$.

Now assume that the values for $x_{i+1}, \ldots, x_n$ are fixed, say $(x_{i+1}, \ldots, x_n) = (y_{i+1}, \ldots, y_n)$. We can then bound the values of x_i that can possibly be completed into a vector $(y_1, \ldots, y_n)$ with $f(y_1, \ldots, y_n) < A_1$ for some f.

$$f(x_1, \ldots, x_i, y_{i+1}, \ldots, y_n) \geq A_i (x_i - \sum_{j=i+1}^{n} \alpha_{ij} y_j)^2 + \sum_{k=i+1}^{n} A_k (y_k - \sum_{j=k+1}^{n} \alpha_{kj} y_j)^2 \tag{2.4}$$

$$\geq A_i (x_i - \sum_{j=i+1}^{n} \alpha_{ij} y_j)^2 + C \tag{2.5}$$

$$\geq (2/3)^{\lfloor (i-1)/2 \rfloor} (3/4)^{(i-1) \bmod 2} A_1 (x_i - \sum_{j=i+1}^{n} \alpha_{ij} y_j)^2 + C \tag{2.6}$$

[†]The set X_i in this theorem corresponds to the set $\{x \in X_i \mid x_n \neq 0\}$, where the last X_i is the set as defined in the paper.

where C is the minimum over all KZ-reduced forms $\sum_{k=i+1}^{n} A_k(y_k - \sum_{j=k+1}^{n} \alpha_{kj} y_j)^2$ with

$$A_k \geq (2/3)^{\lfloor (k-1)/2 \rfloor} (3/4)^{(k-1) \bmod 2} A_1.$$

Furthermore, introduce constants C_1, C_2, such that

$$C_1 \leq \sum_{j=i+1}^{n} \alpha_{ij} y_j \leq C_2 \text{ for all size-reduced sets of coefficients } \alpha_{kj}.$$

Then it is clear that for

$$x_i > \left((3/2)^{\lfloor (i-1)/2 \rfloor} (4/3)^{(i-1) \bmod 2} (1 - C/A_1) \right)^{1/2} + C_2$$

and

$$x_i < -\left((3/2)^{\lfloor (i-1)/2 \rfloor} (4/3)^{(i-1) \bmod 2} (1 - C/A_1) \right)^{1/2} + C_1,$$

$f(x_1, \ldots, x_i, y_{i+1}, \ldots, y_n) \geq A_1$ for any size-reduced form f whose tail $f_{i+1,n}$ is KZ-reduced. Hence only values for x_i within these bounds need to be considered. □

We will now compute an explicit bound on C. Since $f_{i+1,n}$ is already KZ-reduced, we know

$$f_{i+1,n}(y_{i+1}, \ldots, y_n) \geq A_{i+1}$$

for all $f_{i+1,n}$ and for all $\boldsymbol{y}$. Hence $C \geq A_{i+1}$. Moreover, we can find lower bounds on the individual terms $A_k(y_k - \sum_{j=k+1}^{n} \alpha_{kj} y_j)^2$ since the y_j are fixed, the α_{kj} are bounded, and the first and second KZ-inequalities bound A_k. The bound on C is simply the best of the bounds that follow in this way. Size-reducedness and the fact that the y_j are constant gives bounds on C_1 and C_2.

Note that, since $f(-\boldsymbol{x}) = f(\boldsymbol{x})$, we only need to include the vectors with $x_n > 0$ in X_n. The following theorem, which can be extended to forms in more variables in a straightforward way, can be used to reduce the size of the list further.

Theorem 2.4. *Let $v = (a, x, y, z)$ with $\operatorname{sign}(a) = -\operatorname{sign}(x)$ and $|a| \geq \frac{1}{2}(|y| + |z|)$. There is no quadratic form in 4 variables such that $f(v) < A_1$, given size-reducedness and the fact that f_{24} is KZ-reduced.*

Proof. Without loss of generality, assume $a > 0$ and $x < 0$ (take $-v$ instead of v if this is not satisfied). Then

$$\begin{aligned} f(v) &= A_1(a - \alpha_{12}x - \alpha_{13}y - \alpha_{14}z)^2 + \ldots + A_4 z^2 && (2.7)\\ &\geq A_1(a - \alpha_{12}x - \alpha_{13}y - \alpha_{14}z)^2 + A_2 && (2.8)\\ &\geq A_1 \alpha_{12}^2 + A_2 && (2.9)\\ &\geq A_1, && (2.10) \end{aligned}$$

where (2.8) follows from $f_{24}(x, y, z) \geq A_2$; (2.9) follows from $\alpha_{12} \geq 0$, $\alpha_{13}, \alpha_{14} \in (-1/2, 1/2]$ and $|a| - 1/2|y| - 1/2|z| \geq 0$. Finally, (2.10) follows from $f_{01}(0, 1) \geq A_1$. □

The proofs in this section are all constructive. Hence one can generate a set X_n for any n. Note that in principle the lower bound on C can be sharpened by solving a semidefinite programming problem. The author has chosen not to follow that approach because it would have made the method less transparent. The arguments above determine inclusion in X_n on a case-by-case basis, using only quite general properties implied by KZ-reducedness of the smaller forms. No interaction between different vectors in X_n is considered. The surplus vectors can be proven redundant afterwards using the SDP formulation in a systematic way that fits perfectly in the general framework. See Section 6.3 for $X_n, n \leq 5$.

Chapter 3

The Semidefinite Programming Relaxation and Branching Strategies

3.1 A Semidefinite Relaxation

We start the derivation of the semidefinite programming relaxation by remarking that a quadratic form f can be represented by a symmetric, positive semidefinite matrix B, such that $f(\boldsymbol{x}) = \boldsymbol{x}^t B \boldsymbol{x}$ for all $\boldsymbol{x} \in \mathbb{R}^n$. The matrix B is assumed to have full row rank, which implies that it is actually positive definite.

Definition 3.1. *A* decomposition *of a symmetric $n \times n$ matrix B is a set of matrices $B^1, \ldots, B^n$ satisfying*

$$B^i \text{ is a symmetric } (n-i+1) \times (n-i+1) \text{ matrix}^* \tag{3.1}$$

such that

$$B = \bar{B}^1 + \cdots + \bar{B}^n \tag{3.2}$$

where $\bar{B}^i$ is an $n \times n$ matrix obtained by padding B^i on the top and left with zeroes. To be precise: if $B^i = (B^i_{jk})_{i \le j,k \le n}$, then $\bar{B}^i = (B^i_{jk})_{1 \le j,k \le n}$ with $B^i_{jk} = 0$ for $0 \le \min\{j,k\} < i$.

Lemma 3.2. *A positive definite matrix B has a unique decomposition*

$$B = \bar{B}^1 + \bar{B}^2 + \cdots + \bar{B}^n \tag{3.3}$$

where B^i is positive semidefinite and of rank one. Moreover, the matrix B^i can be found from the Lagrange expansion of the quadratic form associated with B by

$$B^i = A_i(1, -\alpha_{i,i+1}, \ldots, -\alpha_{in})(1, -\alpha_{i,i+1}, \ldots, -\alpha_{in})^t.$$

We will refer to a decomposition as in the lemma as a *Lagrange decomposition.*

For ease of notation, in the remainder of this section n, the dimension of B, will be fixed. We encode the set of all possible coefficients of the ith term of the Lagrange expansion of a form f as

$$\mathcal{A}^i := \big\{ \sqrt{A_i}(1, -\alpha_{i,i+1}, \ldots, -\alpha_{in}) \in \mathbb{R}^{n-i+1} \mid A_i \ge 0,\ \alpha_{i,i+1} \ge 0,\ |\alpha_{ij}| \le 1/2 \quad (j = i+1, \ldots, n) \big\}.$$

This set is clearly a convex cone. The minus signs in front of the α_{ij} are slightly inconvenient, but they are consistent with historical notation. Let $\boldsymbol{\alpha}_i \in \mathcal{A}^i$. Any size-reducedness condition $\alpha_{ij} \le \gamma_{ij}$ can be written as

$$(\gamma_{ij}, 0, \ldots, 0, 1, 0, \ldots, 0)^t \boldsymbol{\alpha}_i \ge 0,$$

where the "1" is in the position corresponding to $-\alpha_{ij}$. Likewise a condition $\alpha_{ij} \ge \beta_{ij}$ can be written as

$$(-\beta_{ij}, 0, \ldots, 0, -1, 0, \ldots, 0)^t \boldsymbol{\alpha}_i \ge 0.$$

*The matrix called B^i in the paper is the same as $\bar{B}^i$ in this report.

Note that

$$(1, 0, \ldots, 0)^t \boldsymbol{\alpha}_i \geq 0.$$

is implied. The set of all vectors of this format (i.e. forming a lower or an upper bound on an α_{ij}) will be denoted[†] by D_i. Note that the elements of D_i depend on the lower and upper bounds β_{ij}, γ_{ij}. Unless specified otherwise, we will assume the default values

$$\begin{aligned} \beta_{ij} &= 0 && \text{if } j = i+1 \\ \beta_{ij} &= -1/2 && \text{if } i+1 < j \leq n \\ \gamma_{ij} &= 1/2. \end{aligned}$$

We then have

$$\mathcal{A}^i = \{\boldsymbol{\alpha}_i \in \mathbb{R}^{n-i+1} \mid \boldsymbol{d}^t \boldsymbol{\alpha}_i \geq 0 \text{ for all } \boldsymbol{d} \in D_i\}. \tag{3.4}$$

Next we define a similar notion for the ith matrix in the Lagrange decomposition of a positive semidefinite matrix B.

$$\mathcal{B}^i := \{\boldsymbol{\alpha}\boldsymbol{\alpha}^t \mid \boldsymbol{\alpha} \in \mathcal{A}^i\}.$$

From Lemma 3.2 we see immediately that a decomposition $B = \bar{B}^1 + \cdots + \bar{B}^n$ is a Lagrange decomposition if and only if $B^i \in \mathcal{B}^i$ for all i. It is clear that if $B^i \in \mathcal{B}^i$, then

- $B^i \in \mathbf{S}_+^{n-i+1}$ (the positive semidefinite cone), and
- $\boldsymbol{d}_1 \boldsymbol{d}_2^t \star B^i \geq 0$ for all $\boldsymbol{d}_1, \boldsymbol{d}_2 \in D_i$.[‡]

The constraints in the second item are all constraints that can be obtained by using one lower or upper bound, or by combining exactly two of the lower- and upper bounds. Of course one can also combine three of the lower and upper bounds, as follows.

Suppose we have constrained variables $-1 \leq x_1, x_2, x_3 \leq 1$. Then the following inequality is valid:

$$\frac{1}{2}(1+x_1)(1+x_2)(1+x_3) + \frac{1}{2}(1-x_1)(1-x_2)(1-x_3) = 1 + x_1x_2 + x_1x_3 + x_2x_3 \geq 0.$$

The inequality remains valid if we replace any of the x_i by $-x_i$. This yields a total of 4 new inequalities, and straightforward enumeration learns that those are all inequalities that can be obtained from combining products of 3 inequalities such that the term $x_1x_2x_3$ cancels. If we select three of the α_{ij} (no two equal), we can translate and scale each of the intervals $\beta_{ij} \leq \alpha_{ij} \leq \gamma_{ij}$ into the interval $[-1, 1]$ and plug them into the inequality above. We can now construct a matrix E^i_{jklm} ($i+1 \leq j < k < l \leq n$; $m = 1, \ldots, 4$ for the four inequalities of this form) with as coefficients -1 in the top left corner (position (i, i)), the coefficient in the inequality of α_{ij} at position (i, j) and the coefficient of $\alpha_{ij}\alpha_{ik}$ at position (j, k). This means we also have that

- $E^i_{jklm} \star B^i \geq 0$ for all E^i_{jklm}.

Similarly, there are quadratic inequalities among the nonnegative combinations of $m > 3$ linear inequalities. For fixed m the extremal inequalities among those can be found using Fourier-Motzkin elimination to eliminate the higher-order terms. As shown in [PR01], there is an M such that all quadratic inequalities implied by our linear inequalities have been generated for an $m < M$. However, the increase in strength of the formulation on account of the third-order inequalities described above was already comparatively small, so this method was not pursued further.

Putting the above together, we have that

$$\mathcal{B}^i \subseteq \{B^i \in \mathbf{S}_+^{n-i+1} \mid \boldsymbol{d}_1 \boldsymbol{d}_2^t \star B^i \geq 0, E^i_{jklm} \star B^i \geq 0 \text{ for all } \boldsymbol{d}_1, \boldsymbol{d}_2 \in D_i, \text{ for all } i+1 \leq j < k < l \leq n\} =: \mathcal{K}^i.$$

The set $\mathcal{K}^i$ is clearly a convex cone, and therefore it can be used as building block for a convex optimization problem. In fact, we can now write down a semidefinite programming problem where we minimize a linear

[†]In the paper the set D_i is not described explicitly, and, similar to the difference in meaning of the matrices B^i, the set D_i in the paper is defined on vectors in $\mathbb{R}^n$. It must also contain more vectors than the D_i in this report, since it must ensure that each vector in the set $\{a \in \mathbb{R}^n \mid ad \geq 0 \text{ for all } d \in D_i\}$ starts with a number of zeroes.

[‡]In this report we write $A \star B = \sum_{i,j} A_{ij} B_{ij} = \mathrm{Tr}(A^t B)$.

function in the entries of the B^i subject to the constraints that each of the B^i is positive semidefinite and a number of linear inequalities in the entries of the B^i:

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n} C^i \star B^i && && \text{(SDP)}\\
\text{subject to} \quad & B^n_{nn} = 1 && && (3.5)\\
& \boldsymbol{d}_1 \boldsymbol{d}_2^t \star B^i \geq 0 && \text{for all } \boldsymbol{d}_1, \boldsymbol{d}_2 \in D_i,\ 1 \leq i \leq n && (3.6)\\
& E^i_{jklm} \star B^i \geq 0 && \text{for all } i+1 \leq j < k < l \leq n,\ m = 1, \ldots, 4, \text{ and } 1 \leq i \leq n && (3.7)\\
& B^i_{ii} \leq \boldsymbol{x}^t \Big(\sum_{k=i}^{i+j-1} \bar{B}^k \Big) \boldsymbol{x} && \text{for all } \boldsymbol{x} \in \bar{X}_j,\ 1 \leq i \leq n-2, \text{ and } 2 \leq j \leq n-i+1 && (3.8)\\
& B^i \succeq \mathbf{0} && (1 \leq i \leq n). && (3.9)
\end{aligned}$$

The set $\bar{X}_j$ is defined as $\{(\mathbf{0}, x) \in \mathbb{Z}^n \mid x \in X_j\}$. Equality (3.5) is needed to remove the scale-invariance from the problem. Without it,for every solution $(B^i)_{i=1}^n$ to the remaining system, we would have a solution $(\zeta B^i)_{i=1}^n$ for $\zeta > 0$. The author does not find this to be the most convenient way of intersecting the cone. In fact, at first the equality $B^1_{11} = 1$ was used. However, that equality did not bound all outer coefficients from above, which leads to numerical problems for some objective functions. The current method does not have that problem.

There are several software packages that can approximate the optimal solution to this problem quite efficiently. Section 3.5 describes how to do this in one specific package. But first we describe how to use a branching process to improve the bounds provided by this formulation.

3.2 Branch and Bound

The linear inequalities of the previous section were carefully crafted to depend on general upper and lower bounds $\beta_{ij} \leq \alpha_{ij} \leq \gamma_{ij}$. Hence the branch and bound process described in the paper can be executed efficiently.

Since we are looking for rank-one matrices, it seems wise to branch on intervals whose corresponding matrix B^i deviates significantly from a rank-one matrix. The way we measure this is the following: for each i, we construct the rank-one matrix $\widetilde{B}^i$ for which

$$\widetilde{B}^i_{jk} = B^i_{ij} B^i_{ik} / B^i_{ii} \quad j, k \in \{i, \ldots, n\}, \tag{3.10}$$

that is, $\widetilde{B}^i$ is the unique symmetric rank-one matrix such that the first row of $\widetilde{B}^i$ equals the first row of B^i.

If B^i is close to rank one, $\widetilde{B}^i$ will not differ too much from it. To determine our branching point, we now select the $j \in \{i+1, \ldots, n\}$ for which

$$\sum_{k=i}^{n} |\widetilde{B}^i_{jk} - B^i_{jk}|$$

is maximal. Intuitively, we select the j that contributes most to the height of the rank of B^i. Maximizing this quantity over all i then gives a good choice for a branching point.

Usually we try to find a lower bound on the minimum. Then it only makes sense to branch in the leaf of the current branching tree that minimizes the objective function: our lower bound is as good as the lowest value among the leaves.

Other branching strategies have been considered (see also [Zwa05]), but this was by far the most efficient one among those. However, observation of the branching process learns that there are sometimes long stretches in which the optimum does not improve significantly. This indicates that there might still be room for improved strategies. One possibility is outlined in the next section.

3.2.1 A different branching strategy

The observation in the previous paragraph suggests a different approach to the branching process: by spending more time in each node, the total number of nodes that are visited can remain relatively small. A first idea would be to compute the improvement in the lower bound for each possible choice of an interval, and then select the one that improves the objective function the most. More concretely, it selects that branching point where the minimum objective function of the two resulting leaves is largest. Experiments show that especially

in the first few steps much progress is made, but soon the algorithm finds itself in a point where none of the possibilities show any significant improvement. Then several secondary strategies can be employed, such as selecting the branching point where the maximum objective function of the two resulting intervals increases most, or reverting to the strategy of the previous section. Yet another option would be to consider the possible branchings a few levels deep. However, computation time will increase significantly in that case.

3.3 A note on the sets X_i

Since we try to find lower bounds on certain minima, we do not need to use exactly the set X_i as constructed in the proof of Theorem 2.3. Any subset or superset will also yield a valid lower bound. However, if the subset is too small the lower bound will not be tight. If the set is too big the SDP will have more inequalities and hence it will take the computer longer to find a solution (in particular, each inequality gives rise to a dual variable). For forms in more variables it might be a wise idea to use a cutting plane approach: start with a small number of inequalities and add ones that are violated if deemed necessary.

3.4 Strategies for finding quadratic forms

For some of the problems we consider we are interested in finding an optimal quadratic form. This is a nontrivial task, as the B^i in the primal solution returned by the interior point algorithm will usually not have rank 1. We describe two ways of creating a quadratic form from this solution. The first way is simply replacing each B^i by the $\widetilde{B}^i$ from (3.10). The second way follows by observing that, regardless of the rank of the B^i, $\sum_{i=1}^n B^i$ will be a quadratic form, from which we can compute the Lagrange expansion.

Both methods will not give a KZ-reduced form for most of the time, but sometimes, especially as the branching process gets closer to the optimum, they do find a form that is almost KZ-reduced. After some manual tweaking such a form may yield a reasonably good upper bound on the optimum. All quadratic forms listed in the paper and this report were obtained in this way.

Finding a way to obtain such forms completely automatically is still a significant challenge.

3.5 Description of the SDP in MaxDet syntax

The MaxDet software [VBW] solves the following problem.

$$\begin{aligned} \text{minimize} \quad & c^t y + \log\det G(y)^{-1} && \text{(MD)} \\ \text{subject to} \quad & G_0 + y_1 G_1 + \cdots + y_m G_m \succ 0 && (3.11) \\ & F_0 + y_1 F_1 + \cdots + y_m F_m \succeq 0. && (3.12) \end{aligned}$$

The dual to this problem is

$$\begin{aligned} \text{maximize} \quad & \log\det W - G_0 \star W - F_0 \star Z + l && \text{(MP)} \\ \text{subject to} \quad & G_k \star W + F_k \star Z = c_k \qquad k = 1, \ldots, m && (3.13) \\ & W = W^t \succ 0, Z = Z^t \succeq 0. && (3.14) \end{aligned}$$

Since the software package Rigorous Maxdet [SV, SV06], which we used to make our proofs rigorous, was based on MaxDet, we will cast the SDP in this format.

Clearly every constraint in Problem (SDP), say constraint k, can be expressed as $\sum_i F_k^i \star B^i \geq c_k$ for some matrices F_k^i and some c_k. If we introduce slack variables we can transform those constraints into equalities:

$$\sum_{i=1}^{n} F_k^i \star B^i - s_k = c_k$$

where we require $s_k \geq 0$. Any feasible solution $(B^1, \ldots, B^n, s_1, \ldots, s_m)$ can be written as the following block

diagonal matrix:

$$Z := \begin{bmatrix} \boxed{B^1} & & & & & & \\ & \boxed{B^2} & & & & & \\ & & \ddots & & & & \\ & & & \boxed{B^n} & & & \\ & & & & \boxed{s_1} & & \\ & & & & & \ddots & \\ & & & & & & \boxed{s_m} \end{bmatrix}.$$

In this matrix, all entries outside the blocks are 0. This matrix is clearly positive semidefinite if and only if all of its blocks are positive semidefinite. For a 1×1 block this reduces to the requirement that it is nonnegative.

In the same way the matrices making up a constraint can be grouped:

$$F_k := \begin{bmatrix} \boxed{F_k^1} & & & & & & \\ & \boxed{F_k^2} & & & & & \\ & & \ddots & & & & \\ & & & \boxed{F_k^n} & & & \\ & & & & \boxed{0} & & \\ & & & & & \ddots & \\ & & & & & & \boxed{-1} \\ & & & & & & & \ddots \\ & & & & & & & & \boxed{0} \end{bmatrix}$$

where the -1 is in the position of the k'th slack variable in Z. Finally we write the target function as the

following matrix:

$$F_0 := \begin{bmatrix} C^1 & & & & & \\ & C^2 & & & & \\ & & \ddots & & & \\ & & & C^n & & \\ & & & & 0 & \\ & & & & & \ddots \\ & & & & & & 0 \end{bmatrix}.$$

Since we are not interested in determinant optimization, we can define $W := [1]$, $G_0 := [1]$ and $G_k = [0]$. With that final step we have cast our problem into the formulation used by MaxDet. Note that if we are looking for a feasible solution to our original problem, we must find a solution to the dual of the MaxDet formulation. Likewise, if we are looking for a lower bound on our minimum, we have to minimize the primal problem of the MaxDet formulation. The optimal solution to the MaxDet-formulation of the problem is minus the optimal solution to our original problem. We will refer to the minimization problem of the MaxDet-formulation as the *dual* problem and to the maximization problem (i.e. our original problem) as the *primal* problem throughout this report.

Chapter 4

Making the lower bounds rigorous

The methods described in the previous chapter give tools to generate numerical evidence that certain lower bounds hold. It is not entirely trivial to turn this numerical evidence into mathematically rigorous proofs. In this chapter we talk about the steps that need to be taken to achieve this.

Once the proofs have been generated, their verification is straightforward. The last two sections of this chapter summarize this process.

4.1 From floating-point numbers to rational numbers

The software used to find the primal and dual solutions uses fast routines for computing with floating-point numbers. These numbers have limited precision (roughly 13 or 14 digits, usually) and hence all computations are approximate. Any result returned by such software should therefore be interpreted as "within this precision, it seems that this answer is a reasonable approximation". As a first step towards rigorous proofs, one should get rid of these floating-point numbers. For the constraint matrices F_k and the right-hand side vectors c_k this is rather straightforward: all the information needed to generate these matrices is expressed in rational numbers. In fact, we will usually scale them to ensure that all entries are integral (but see Section 4.3).

Next we consider the dual solution returned by the software. There is no way of knowing how to scale this vector beforehand. It is not even clear that the optimal vector is rational! However, one can use continuous fraction expansions to get rational numbers that are very close to the desired number. This is the method that we employ.

4.2 Ensuring strict feasibility

Suppose that for a certain interval set the SDP is described by $F_0, \ldots, F_m$ and $c_0, \ldots, c_m$, all with integer entries, and that we have a rational approximation y of a feasible dual vector. That is, y is close to a vector $\widetilde{y}$ satisfying

$$F_0 + \widetilde{y}_1 F_1 + \cdots + \widetilde{y}_m F_m \succeq 0.$$

How can we verify if y itself satisfies this relation, and, crucially, can we ensure beforehand that y will satisfy it? If we had used MaxDet for our computations, both questions could have been answered easily. MaxDet will return a solution to your problem that is *strictly* feasible. Then the approximation by rational numbers – if close enough – will also be strictly feasible. The first question then reduces to checking positive *definiteness* of a matrix, and this can be accomplished by checking positiveness of the determinant of the top left 1×1 submatrix, then the top left 2×2 submatrix, up to the determinant of the whole matrix.

MaxDet is not the best SDP solver available nowadays. First of all, it requires a strictly feasible dual solution to even start working, and it is not clear how to obtain that from the description above. This problem can be circumvented by the introduction of a new variable, but that will create a new constraint with index $m+1$ in the primal solution. This constraint should be such that it is satisfied by all KZ-reduced quadratic forms, which leads to a huge number c_{m+1}. This, in turn, seriously hampers the numerical stability of MaxDet. Moreover, MaxDet gives up sooner than other packages with a simple error message. By comparison, SeDuMi [Stu99] always returns a solution and some information on the quality, even if this solution is quite far from feasibility, leaving it up to the recipient to deal with it. Finally, MaxDet seems to be quite a bit slower. These reasons

moved the author to import the SDP into SeDuMi, solve the problem there, and translate the solution back to the MaxDet format for further processing. The cost of this is that SeDuMi will return a dual solution that is not strictly feasible, and in fact this solution is usually slightly infeasible.

How can we still obtain strict feasibility? This can be accomplished by modifying the problem a bit before offering it to the solver. We replace F_0 by $F_0 - \varepsilon I$ for some small ε and solve the resulting SDP. If the computed dual solution has eigenvalues that are slightly negative but more than $-\varepsilon$, they will still be strictly positive for the original problem. This means we can use these dual solutions in our approach as described above.

4.3 Numerical stability

Even then, we need to be careful. As intervals get split more often, the numerical stability of the corresponding SDP decreases, and eventually the solution returned will no longer have eigenvalues within the bound imposed by the previous section. There is no way to recover from such instabilities in the rounding phase, so one must monitor the returned dual solutions throughout the branching process, and deny branching on choices that yield instable results.

Somewhat higher stability can be achieved by scaling each of the F_k so that all numbers are roughly of the same size. It is easy to scale the obtained dual vectors to correspond to the original, unscaled problem. But this will only postpone the problems of the previous paragraph.

There are several strategies for dealing with stability problems. One can change the parameters of the solver to make it try harder (but take longer), one can put restrictions on the minimum width an interval should have, and so on. When looking at the implementation the reader will notice many places in which the method can be finetuned.

4.4 Testing KZ-reducedness of a quadratic form

In order to test whether a quadratic form is KZ-reduced, one needs to compute its Lagrange expansion, check whether the inner coefficients are size-reduced, and whether $\min\{f_{in}(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathbb{Z}^{n-i+1}, \boldsymbol{x} \neq \boldsymbol{0}\} = A_i$ for all $i \in \{1, \ldots, n-1\}$. One way to do this is to check whether $f_{in}(\boldsymbol{x}) \geq A_i$ for all $\boldsymbol{x} \in X_{n-i+1}$, where this last set is the one computed by the methods in Chapter 2.

If the form was intended to show necessity for a particular vector $\boldsymbol{x}$, it is not KZ-reduced. In this case, one needs to verify KZ-reducedness, whether f_{2n} is KZ-reduced, whether $f_{1n}(\boldsymbol{x}) < A_1$, and whether $f_{1n}(\boldsymbol{y}) \geq A_1$ for all $\boldsymbol{y} \in X_n \setminus \{\boldsymbol{x}\}$.

4.5 Verifying a lower bound

This verification is not much more involved than the previous one. A lower bound consists of a series of interval sets whose union is the space of size-reduced inner coefficients of Lagrange expansions, coupled with a feasible dual vector. A verifier therefore needs to accomplish the following two tasks:

1. Check if the union of all interval sets is indeed the whole space of size-reduced inner coefficients,
2. Verify the feasibility of the dual vector for each interval set.

The second task is accomplished by computing, for each dual vector, the SDP formulation corresponding to the interval set, establishing the positive semidefiniteness of $F_0 + \sum_{i=1}^{m} y_i F_i$, and finally evaluating cy to obtain the lower bound. As described in Section 4.2 we will in fact provide dual vectors that are *strictly* feasible, so that we only have to establish positive definiteness of $F_0 + \sum_{i=1}^{m} y_i F_i$, which is relatively straightforward.

To verify the proofs of the theorems in the paper accompanying this report, one should ascertain that the computed SDP formulation is correct. That is: given an interval set, do all KZ-reduced quadratic forms, with inner coefficients in this interval set, satisfy the constraints of the corresponding SDP formulation?

Chapter 5

Programs

The computer code and certificates are published as a digital appendix to this report and are available from the SPOR reports website of the department of Mathematics and Computer Science of the Technische Universiteit Eindhoven,

http://www.win.tue.nl/bs/spor/

In addition, the author will strive to keep a version of the code available from the following website:

http://www.win.tue.nl/kz/

On this website updated versions of the code will appear, as well as new results.

This chapter tries to guide users through the process of getting this code to work, and tries to give some hints that help reading the files.

5.1 Installation

The difficulty of getting the code to work depends on which parts of the code you are going to use. If you are only interested in the verification of our claims, this should be rather straightforward: you only need to compile some C++ programs, as detailed in the next subsection. The code that verifies a quadratic form was written in Mathematica, and should work with most versions of that software. If you want to prove your own lower bounds, you will need to use MATLAB, and you will have to compile some C++ files using MATLAB's "mex" compiler. The details of this can be found in the remainder of this section.

5.1.1 How to compile the code for `genlists` and `verifydual`

After decompressing the contents of the archive into a directory, here's what you do (assuming the presence of a Unix-like "make" utility*):

- Obtain and compile NTL by Victor Shoup [Sho].
- Change paths and settings in the file `Makefile` to reflect program locations. Specifically, look for the lines starting with `CXXFLAGS` and `LIBS`.
- Type `make` from the command prompt

Typically, in the second step you specify include directories so that your compiler can find NTL. If you compiled NTL without GMP, you need to remove the option `-lgmp`. Make sure that you don't change any tabs in the file.

To test your programs, type

```
./genlists 3 > dim3.txt 2> dim3-redundants.txt
```

*Windows users may want to consider installing Cygwin, see http://www.cygwin.com/

(assuming a Bash-like shell, where you can route the standard error and standard output separately - otherwise the results are unpredictable, and it is safest to change the source code by removing line 51 of `genlists.cpp`), and

```
./verifydual < certs/kz3proof.txt
```

Note that `verifydual` will not work if you put it in your path - it uses its calling string ('./verifydual' in the examples above) to deduce the location of the directory `xrows` ('./xrows' in the examples above).

5.1.2 How to interface between the MATLAB and C++ parts of the code

First, make sure the programs in the previous section work (specifically, make sure NTL is available). Then the basic process is this:

- Obtain and install SeDuMi [Stu99]
- Start MATLAB
- Configure mex (`mex -setup`)
- Change paths and settings in `compileAkz.m` to reflect program locations
- change into the directory containing the files
- run `compileAkz.m`

To test the system, run the following commands in MATLAB:

```
akz(3,[-1,0,0],'kz3test',3,[],[],-1);
buildCerts(3,[-1,0,0],'kz3test');
```

and then, from the command prompt,

```
./verifydual < kz3testproof.txt
```

Unfortunately the process is not as smooth as suggested here. The next paragraphs will describe some hurdles that the author has run into, sorted by operating system. Note that in principle the code uses no platform-specific constructs. All issues below are caused by external factors.

Linux The following applies to systems with GCC 3.4 or newer and MATLAB 7.0.4 (aka R14SP2). If GCC is older (3.3) or MATLAB newer, you probably don't have to deal with what follows. Only resort to this if you get errors when running the compiled code. This affects SeDuMi as well as the code in this project.

The version of MATLAB mentioned above has, for some reason, several outdated libraries (most notably `libgcc_s.so.1`) hidden in its directory structure. These will conflict with GCC 3.4. Several solutions exist. The easiest one is to type the following command just before starting MATLAB (assuming the Bash shell - other shells may have different ways of setting environment variables):

```
export LD_PRELOAD=/lib/libgcc_s.so.1
```

This will make sure the linker links against the right version of the library. Note that you have to do this every time you want to run the code. You may want to consider adding it to your environment permanently, but the author is not sure if this could cause unwanted side-effects. Another way is to install an older version of gcc (version 3.3, for example) alongside your new one and modify `.matlab/R14/mexopts.sh` to point to gcc33 and g++33 as compilers. This directory `.matlab` can be found in the directory from which you run MATLAB. The author did not test this second method.

MATLAB will now run the created files just fine, but it will crash when you exit.

Microsoft Windows XP If you are unfortunate enough to be doing all this in Windows, there are more hurdles along the road. In fact, the author only succeeded in getting the code to run at all after a lot of effort, and even then only for one compiler. This section shows his attempts, so that you can save yourself some time. Moreover, the compiled `.dll` files for use with MATLAB 7.0.4 have been included for convenience, so you don't have to compile the code yourself. Note that no compiled version of the programs in the previous section was included, since running a "black box" is not very illuminating in terms of verifying a proof. For experimentation and generation of certificates, however, using a black box does not have to be an objection at all.

MATLAB comes with a built-in compiler (lcc) for C programs. This is of no use for C++ code. Hence you need an alternative. *MSVC++* was not available to the author. The possibility of using *Cygwin* in conjunction with *Gnumex* seems promising but leads to MATLAB crashes (though the Gnumex website claims that it works if you install GCC 3.2 instead of the current version, something the author didn't try). Gnumex also provides support for *MingW*. However, trying to link NTL to the mex code leads to a long sequence of "undefined references", concluding with an error. *Open Watcom* (the author tried 1.3 and 1.4) requires MATLAB to be installed in a path without spaces. More importantly, it failed to compile NTL. *Borland C++ Builder 6* initially does not seem to compile the code for NTL (due to an "ambiguous template statement"). This can be resolved by replacing every occurrence of "`negate`" by "`NTL::negate`". You also have to specify a "page size" of 128 in the TLib pane of the Project Options dialog. But after that the programs would compile on the author's system.

Finally note that MATLAB 7.1 (R14SP3) has changed the extension of Mex files from .dll to .mexw32 (or something similar). Also the internals seem to have changed, but the author reverted to 7.0.4 before he could look into that in more detail.

Other systems The author does not have access to MATLAB on other systems, so you are on your own there.

5.2 Understanding the code

Table 5.1 contains a short description of the contents of each source file.

Filename	Purpose
`rig_maxdet.h`, `rig_maxdet.cpp`	The code for Rigorous Maxdet was originally written by Achill Schürmann and Frank Vallentin [SV]. This version of the code is nothing but a rearrangement of a small subset of their code that handles the rounding of the floating-point dual solutions.
`QQ.h`, `structs.h`, `structs.cpp`	General definitions: a class for dealing with rational numbers (a slight modification of the file by the same name included with Rigorous Maxdet), definitions of the data structures and a few useful functions.
`genlists.cpp`	Implementation of the method described in Chapter 2. Originally written by Rudi Pendavingh, modified to use `QQ.h`.
`xrows.h`, `xrows.cpp`, `constraintmatrix.h`, `constraintmatrix.cpp`	The generation of the SDP, as described in Chapter 3.
`xrows/*`	Sufficient sets X_n, for $n \leq 6$. Read by `xrows.cpp`.
`constraintMatrix.m`, `mlconstraintMatrix.cpp`, `resize.m`	Generation of the SDP in SeDuMi input format. The file with prefix `ml` provides the interface between the C++ code and MATLAB. The function resize is needed to scale the dual vector so that it corresponds to a valid dual vector in the C++ code.
`akz.m`	This function performs the branching process.
`specialBaseConstr.m`, `splitInterval.m`, `checkQF.m`, `Split.m`, `branchPosOf.m`, `posOf.m`, `toVec.m`, `toExpan2.m`, `toExpan.m`, `toQF.m`, `verifyExpan.m`, `vecToExpan.m`	some auxiliary functions used by `akz.m` and (for some) `buildCerts.m`.
`buildCerts.m`, `mlbuildCerts.cpp`	Convert certificates that were obtained numerically by `akz` into rational, and hence verifiable, certificates.
`verifydual.h`, `verifydual.cpp`	Verify certificates produced by `buildCerts.m`.
`doComputations.m`, `toVer4Mat.m`, `preSplit.m`, `findRedundants.m`, `setpars.m`	Some miscellaneous small MATLAB functions to perform specific tasks.
`VerifyQF.ma`	Mathematica code to verify KZ-reducedness of a given quadratic form.
`Makefile`, `compileAkz.m`	Commands to build the applications from the source code.
`mlbuildCerts.dll`, `mlconstraintMatrix.m`	Binaries to run the Matlab part of the code under Windows.
`readme.txt`	The contents of the previous section.
`gpl.txt`	The software license file. The programs are distributed under the GNU General Public License, which roughly means you can do whatever you want with the software, but you must distribute all source code of any published program based on this code, and you cannot impose other restrictions on redistribution.

Table 5.1: *Files included in the first digital appendix*

Chapter 6

Data

Clearly the long lists of arbitrary-length integers returned by the methods can not be reproduced completely in this report without increasing the number of pages to a very long integer. For that reason we only present forms violating exactly one inequality in dimensions 3, 4, and 5, and a certificate showing that $A_3 \geq 2/3 - \epsilon$. For certificates of the other claims, please refer to the table in Section 6.2.

As in the previous chapter, the full set of data justifying all claims made by the author on this subject can (at present) be found on the author's website. Of course, the code in the previous chapter can be used to generate the data.

6.1 A lower bound on A_3

Listing 6.1, included for illustrative purposes, shows that

$$-A_1 \geq -843000657631516509143/562000363888803840000 A_3,$$

which proves $A_3 \geq (2/3 - 10^{-7})A_1$. It is formatted in just the way that `verifydual` (see Section 4.5) would accept it. The first line contains the dimension n, the length l of the function to be optimized (either 1 or n), the coefficients of the function to be optimized (an index into X_n if $l = 1$, so that $\boldsymbol{x}^t B \boldsymbol{x}$ is minimized for some $\boldsymbol{x} \in X_n$; if $l = n$ the encoding of a linear combination of the outer coefficients), and the number of interval sets that follow. The branching process will store after each step one of the resulting two nodes at the position of the old node, say position p, and append the other node at the end of the list. To this new node a number is assigned indicating the position of the old node. From these numbers the branching tree can be reconstructed efficiently, which makes it much easier to verify if the whole space of size-reduced inner coefficients is covered. For each interval first this number is given. Then $n - 1$ lines follow with the lower and upper bounds on the inner coefficients. The first of these contains $\alpha_{12}, \ldots, \alpha_{1n}$, the second $\alpha_{23}, \ldots, \alpha_{2n}$, and so on. Finally a feasible vector for the dual of the SDP formulation corresponding to this set of intervals is given. The objective function value for this vector gives a lower bound on the optimum.

For each claim in [PZ06] the author has generated a similar file. However, the choice of intervals that yielded Listing 6.1 was done manually, which results in a remarkably small number of cases.

Listing 6.1: *kz3proof.txt*

```
3 3 -1 0 0 5
-1
([0,1]/2 [-1,1]/2 )
([0,1]/3 )
[1124000727777607680000 168600114529111450419 4 -1121876369050
    -28839286301604983202 -1121868061349 -1121551682833 -1122296051898
    -45088341867299091499 -182893934821874982533 -28883778760249735391
    -28883726140064361798 -1122528289156 -1121572431760 -1122265881955
    -1121001531756 -52583182269215526943 -1121364561815 -1124209355371
    -29295687455260988084 -1122026955208 -62407377676133975052
    -158037923433915576612 -1122728309347 -1123037917653 -1078002233873913458246
    -126533570284055758935 -42066543682030725483 5 -1130259012615 -1129184165878]
```

```
0
([0,1]/3 [-1,1]/2 )
([2,3]/6 )
[112400072777607680000 1686001315263033018286 -1131766127953
    -8007243724317368232 -1123544584111 -1117767310959 -1128994902997
    -4924118409369827747 -23670073964055503871 -2483764878494666199
    -2483757670936094630 -1120068312000 -1116842763299 -1121668727849
    -1116172591135 -2307919237777 -1118227601018 -1136388229368
    -33984366741992692875 -1133593518833 -77175403775879940974
    -106516020767478518008 -1134043806619 -1142295019962 -1264500971530211447949
    -168600130779333025559 2 -1447916522325 -1438570407581 -1134451237817]
1
([2,3]/6 [-3,-2]/6 )
([2,3]/6 )
[112400072777607680000 1686001128294977930318 -1109687524947
    -1427913396099726486 -1111634375216 -1112208598604 -1371013309404062077
    -2556173866114035144 -17969139635901795390 -2256521723472136186
    -27288740000230012753 -1111417305920 -1131780896283 -2126120083788688293
    -1112653510373 -17749612873370311697 -2413526249720062684 -1115331821915
    -1112286309204 -21632282082254242 88 -1110295093736 -27153350668531702069
    -3558899657895480088 -1112231043589 -259841370365692514292
    -693141380465462009078 -254960113112609899238 -737899630255294619852
    -1116346085952]
2
([2,3]/6 [2,3]/6 )
([2,3]/6 )
[112400072777607680000 1686001128584050707717 -1120249266262 -1116721178824
    -1124932432579 -1116760091170 -1124446366054 -1117038131124
    -14683744879110447572 -1117222765932 -108344710738224526 97 -1129109210493
    -7331916259544113868 -1125526117220 -1117081378806 -139832390454220797 13
    -1127967455991 -1121713026785 -1129963302184 -1117127548853 -1135806586551
    -23416684576507255203 -1117051512776 -1122256801284 -306219405614213418528
    -727719695027412506155 -255780961137930345869 -1128835239822
    -702500467921278518793]
2
([2,3]/6 [-2,2]/6 )
([2,3]/6 )
[112400072777607680000 1686001200141997688519 -1119778197259 -1131449315366
    -1122115624322 -1116913047475 -1118877769699 -1132912106255 -2514815752686
    -1118543334881 -1148050334017 -1121867541457 -1118463612287 -1118521757540
    -1115707022151 -175625145869098340 03 -1116376155741 -1121451425785
    -37032324082703375566 -1120389365770 -77764917613660471060
    -103468154102475206443 -1122988007389 -1122735720487 -1200767770500
    -421500308687042842055 -126450088575730699627 3 -1189469894945 -1134257661722]
```

6.2 Correspondence between claims and files

Table 6.1 describes which files in the second digital appendix contain the proofs of the various claims made in the paper and in this report.

6.3 Minimality of the sets of inequalities

To show that a set X_n is minimal, one has to show first that it is a subset of a sufficient set (for example, a subset of a set generated by the method in Chapter 2). For all vectors x that are in the difference between the

Data file	Claim
`kz3proof.txt`	$A_3 \geq (2/3 - 10^{-7})A_1$
`redcheck*proof.txt`	Redundancy in X_5 of the vectors in Table 6.9.
`kz4proof.txt`	$-25A_1 - 36A_2 + 48A_3 + 40A_4 \geq -7 \cdot 10^{-6} A_4$
`kz5_1proof.txt`	$-5A_1 + 2A_4 + 8A_5 \geq -3 \cdot 10^{-4} A_5$
`kz5_2proof.txt`	$-4A_1 - 3A_3 + 4A_4 + 8A_5 \geq -5 \cdot 10^{-5} A_5$

Table 6.1: *Where to find the proofs*

minimal and sufficient set, one has to show redundancy. This can be done by showing that

$$\min\left\{f(x) \mid f_{2n} \text{ is KZ-reduced}, f \text{ is size-reduced}, f(y) \geq A_1 \text{ for all } y \in X_n \setminus \{x\}\right\} \geq A_1.$$

Strict inequality can be established using the SDP described in Chapter 3. Finally, for the remaining vectors one must show that not a single one can be missed. That means one has to find a form

$$f \in \left\{f \mid f_{2n} \text{ is KZ-reduced}, f \text{ is size-reduced}, f(y) \geq A_1 \text{ for all } y \in X_n \setminus \{x\}, f(x) < A_1\right\}.$$

Unfortunately, the word "strict" above is a problem. Some of the vectors that are conjectured to be redundant will have their minimum equal to 1. Since we are using numerical approximations, we can only show a lower bound that is slightly below 1. For these vectors necessity will remain undecided by our current methods. In dimension 5 there are 8 such vectors among those generated by `genlists`. The results for dimensions 3, 4, and 5 are shown in Tables 6.2–6.10.

Note that the quadratic forms in Tables 6.2–6.8 were obtained by rounding and manually tweaking solutions provided by the algorithm. Rounding primal solutions is a much more delicate process than rounding dual solutions, and the software accompanying this report does not provide an automatic way of doing this.

Vector	$(0,0,1)$	$(0,1,1)$	$(1,1,1)$
Certificate	$\begin{bmatrix} 720 & -360 & 0 \\ -360 & 720 & -129 \\ 0 & -129 & 540 \end{bmatrix}$	$\begin{bmatrix} 720 & -360 & 360 \\ -360 & 720 & -450 \\ 360 & -450 & 720 \end{bmatrix}$	$\begin{bmatrix} 720 & -360 & -360 \\ -360 & 720 & -90 \\ -360 & -90 & 720 \end{bmatrix}$

Table 6.2: *The vectors in X_3^* and their certificates.*

$(-1,-1,0,1)$

$$\begin{bmatrix} 720 & -360 & -360 & 360 \\ -360 & 720 & 0 & 90 \\ -360 & 0 & 720 & -347 \\ 360 & 90 & -347 & 795 \end{bmatrix}$$

$(0,-1,0,1)$

$$\begin{bmatrix} 720 & -360 & 360 & -360 \\ -360 & 720 & -360 & 450 \\ 360 & -360 & 720 & -351 \\ -360 & 450 & -351 & 795 \end{bmatrix}$$

$(0,0,0,1)$

$$\begin{bmatrix} 720 & -360 & 350 & 0 \\ -360 & 720 & -266 & -31 \\ 350 & -266 & 722 & -81 \\ 0 & -31 & -81 & 540 \end{bmatrix}$$

$(0,1,0,1)$

$$\begin{bmatrix} 720 & -360 & -360 & 360 \\ -360 & 720 & 0 & -450 \\ -360 & 0 & 720 & -193 \\ 360 & -450 & -193 & 795 \end{bmatrix}$$

$(1,1,0,1)$

$$\begin{bmatrix} 720 & -360 & 360 & -360 \\ -360 & 720 & -360 & -90 \\ 360 & -360 & 720 & -294 \\ -360 & -90 & -294 & 795 \end{bmatrix}$$

$(-1,-1,1,1)$

$$\begin{bmatrix} 1024 & -480 & 480 & 160 \\ -480 & 1025 & -225 & 325 \\ 480 & -225 & 1025 & -325 \\ 160 & 325 & -325 & 1033 \end{bmatrix}$$

$(0,-1,1,1)$

$$\begin{bmatrix} 1024 & -512 & -480 & -160 \\ -512 & 1024 & 240 & 464 \\ -480 & 240 & 1025 & -325 \\ -160 & 464 & -325 & 1025 \end{bmatrix}$$

$(-1,0,1,1)$

$$\begin{bmatrix} 720 & -360 & 360 & 360 \\ -360 & 720 & -180 & -180 \\ 360 & -180 & 720 & -90 \\ 360 & -180 & -90 & 720 \end{bmatrix}$$

$(0,0,1,1)$

$$\begin{bmatrix} 720 & -360 & 360 & -360 \\ -360 & 720 & -180 & 180 \\ 360 & -180 & 720 & -450 \\ -360 & 180 & -450 & 720 \end{bmatrix}$$

$(1,0,1,1)$

$$\begin{bmatrix} 720 & -360 & -360 & -360 \\ -360 & 720 & 88 & 234 \\ -360 & 88 & 728 & -94 \\ -360 & 234 & -94 & 721 \end{bmatrix}$$

$(0,1,1,1)$

$$\begin{bmatrix} 720 & -360 & 180 & 180 \\ -360 & 720 & -360 & -360 \\ 180 & -360 & 720 & -90 \\ 180 & -360 & -90 & 720 \end{bmatrix}$$

$(1,1,1,1)$

$$\begin{bmatrix} 720 & -360 & -180 & -180 \\ -360 & 720 & -180 & -180 \\ -180 & -180 & 720 & -90 \\ -180 & -180 & -90 & 720 \end{bmatrix}$$

Table 6.3: *The 12 vectors in X_4^* and their certificates.*

$(-1,-1,-1,0,1)$

$$\begin{bmatrix} 720 & -360 & -180 & 360 & 180 \\ -360 & 720 & -180 & -180 & 180 \\ -180 & -180 & 720 & -360 & 90 \\ 360 & -180 & -360 & 720 & -88 \\ 180 & 180 & 90 & -88 & 720 \end{bmatrix}$$

$(0,-1,-1,0,1)$

$$\begin{bmatrix} 720 & -360 & 180 & -360 & -180 \\ -360 & 720 & -360 & 180 & 360 \\ 180 & -360 & 720 & -360 & 90 \\ -360 & 180 & -360 & 720 & -113 \\ -180 & 360 & 90 & -113 & 720 \end{bmatrix}$$

$(-1,0,-1,0,1)$

$$\begin{bmatrix} 720 & -360 & -360 & 180 & 360 \\ -360 & 720 & 180 & 180 & -180 \\ -360 & 180 & 720 & -360 & 90 \\ 180 & 180 & -360 & 720 & -50 \\ 360 & -180 & 90 & -50 & 720 \end{bmatrix}$$

$(0,0,-1,0,1)$

$$\begin{bmatrix} 720 & -360 & 360 & 180 & 360 \\ -360 & 720 & -180 & 180 & -180 \\ 360 & -180 & 720 & -180 & 450 \\ 180 & 180 & -180 & 720 & -50 \\ 360 & -180 & 450 & -50 & 720 \end{bmatrix}$$

$(1,0,-1,0,1)$

$$\begin{bmatrix} 720 & -360 & 360 & 180 & -360 \\ -360 & 720 & -180 & 180 & 180 \\ 360 & -180 & 720 & -180 & 90 \\ 180 & 180 & -180 & 720 & -230 \\ -360 & 180 & 90 & -230 & 720 \end{bmatrix}$$

$(0,1,-1,0,1)$

$$\begin{bmatrix} 10000 & -5000 & -4659 & 2310 & 1719 \\ -5000 & 10000 & 2329 & -4660 & -4609 \\ -4659 & 2329 & 10000 & -4990 & 3114 \\ 2310 & -4660 & -4990 & 10001 & -730 \\ 1719 & -4609 & 3114 & -730 & 10001 \end{bmatrix}$$

$(1,1,-1,0,1)$

$$\begin{bmatrix} 100000 & -47057 & 47057 & -26531 & -16372 \\ -47057 & 100000 & -22144 & 50003 & -31224 \\ 47057 & -22144 & 100000 & -51413 & 31224 \\ -26531 & 50003 & -51413 & 102984 & -49665 \\ -16372 & -31224 & 31224 & -49665 & 101000 \end{bmatrix}$$

$(-1,-1,0,0,1)$

$$\begin{bmatrix} 720 & -360 & 360 & -354 & 360 \\ -360 & 720 & -360 & 51 & 90 \\ 360 & -360 & 720 & -333 & 237 \\ -354 & 51 & -333 & 720 & -306 \\ 360 & 90 & 237 & -306 & 795 \end{bmatrix}$$

$(0,-1,0,0,1)$

$$\begin{bmatrix} 10000 & -5000 & -5000 & 537 & -5000 \\ -5000 & 10000 & 0 & 3322 & 6250 \\ -5000 & 0 & 10000 & -4782 & 1321 \\ 537 & 3322 & -4782 & 10037 & -919 \\ -5000 & 6250 & 1321 & -919 & 11042 \end{bmatrix}$$

$(0,0,0,0,1)$

$$\begin{bmatrix} 10000 & -5000 & -2590 & -2266 & 0 \\ -5000 & 10000 & -2410 & 4876 & 69 \\ -2590 & -2410 & 10000 & -2462 & -768 \\ -2266 & 4876 & -2462 & 10000 & -1604 \\ 0 & 69 & -768 & -1604 & 7500 \end{bmatrix}$$

$(0,1,0,0,1)$

$$\begin{bmatrix} 720 & -360 & 360 & -270 & 360 \\ -360 & 720 & -358 & 363 & -450 \\ 360 & -358 & 720 & -364 & 269 \\ -270 & 363 & -364 & 728 & -330 \\ 360 & -450 & 269 & -330 & 797 \end{bmatrix}$$

$(1,1,0,0,1)$

$$\begin{bmatrix} 1024 & -512 & 512 & 480 & -512 \\ -512 & 1024 & -448 & 48 & -128 \\ 512 & -448 & 1040 & 122 & -160 \\ 480 & 48 & 122 & 8575/8 & -384 \\ -512 & -128 & -160 & -384 & 1184 \end{bmatrix}$$

Table 6.4: *The 52 certified vectors in X_5^* and their certificates.*

$(-1,-1,1,0,1)$

$$\begin{bmatrix} 100000 & -47057 & 47057 & -19925 & 16372 \\ -47057 & 100000 & -22144 & -29032 & 31224 \\ 47057 & -22144 & 100000 & -48304 & -31224 \\ -19925 & -29032 & -48304 & 100780 & -18220 \\ 16372 & 31224 & -31224 & -18220 & 101000 \end{bmatrix}$$

$(0,-1,1,0,1)$

$$\begin{bmatrix} 1000000 & -500000 & -460197 & 227732 & -155816 \\ -500000 & 1000000 & 230098 & -479008 & 452908 \\ -460197 & 230098 & 1000000 & -498908 & -322401 \\ 227732 & -479008 & -498908 & 1017860 & -187553 \\ -155816 & 452908 & -322401 & -187553 & 1010000 \end{bmatrix}$$

$(-1,0,1,0,1)$

$$\begin{bmatrix} 720 & -360 & 360 & 180 & 360 \\ -360 & 720 & -180 & 180 & -180 \\ 360 & -180 & 720 & -180 & -90 \\ 180 & 180 & -180 & 720 & 220 \\ 360 & -180 & -90 & 220 & 720 \end{bmatrix}$$

$(0,0,1,0,1)$

$$\begin{bmatrix} 720 & -360 & -360 & 180 & 360 \\ -360 & 720 & 180 & 180 & -180 \\ -360 & 180 & 720 & -360 & -450 \\ 180 & 180 & -360 & 720 & 220 \\ 360 & -180 & -450 & 220 & 720 \end{bmatrix}$$

$(1,0,1,0,1)$

$$\begin{bmatrix} 720 & -360 & -360 & 180 & -360 \\ -360 & 720 & 180 & 180 & 180 \\ -360 & 180 & 720 & -360 & -90 \\ 180 & 180 & -360 & 720 & 40 \\ -360 & 180 & -90 & 40 & 720 \end{bmatrix}$$

$(0,1,1,0,1)$

$$\begin{bmatrix} 720 & -360 & 180 & 180 & 180 \\ -360 & 720 & -360 & 180 & -360 \\ 180 & -360 & 720 & -360 & -90 \\ 180 & 180 & -360 & 720 & 43 \\ 180 & -360 & -90 & 43 & 720 \end{bmatrix}$$

$(1,1,1,0,1)$

$$\begin{bmatrix} 720 & -360 & -180 & 360 & -180 \\ -360 & 720 & -180 & -180 & -180 \\ -180 & -180 & 720 & -360 & -90 \\ 360 & -180 & -360 & 720 & -17 \\ -180 & -180 & -90 & -17 & 720 \end{bmatrix}$$

$(-2,-1,-1,1,1)$

$$\begin{bmatrix} 720 & -360 & -360 & 360 & 360 \\ -360 & 720 & 0 & -180 & 90 \\ -360 & 0 & 720 & -180 & -30 \\ 360 & -180 & -180 & 720 & -90 \\ 360 & 90 & -30 & -90 & 975 \end{bmatrix}$$

$(-1,-1,-1,1,1)$

$$\begin{bmatrix} 720 & -360 & -360 & -360 & 360 \\ -360 & 720 & 0 & 315 & -45 \\ -360 & 0 & 720 & 135 & 15 \\ -360 & 315 & 135 & 720 & -399 \\ 360 & -45 & 15 & -399 & 840 \end{bmatrix}$$

$(0,-1,-1,1,1)$

$$\begin{bmatrix} 1024 & -512 & 256 & -512 & 256 \\ -512 & 1024 & -512 & 256 & 64 \\ 256 & -512 & 1024 & -128 & 352 \\ -512 & 256 & -128 & 1024 & -512 \\ 256 & 64 & 352 & -512 & 1072 \end{bmatrix}$$

$(1,-1,-1,1,1)$

$$\begin{bmatrix} 10000 & -5000 & 5000 & -5000 & -5000 \\ -5000 & 10000 & -5000 & 5000 & 5000 \\ 5000 & -5000 & 10000 & -3334 & 0 \\ -5000 & 5000 & -3334 & 10000 & 0 \\ -5000 & 5000 & 0 & 0 & 11667 \end{bmatrix}$$

$(-1,0,-1,1,1)$

$$\begin{bmatrix} 10000 & -5000 & -5000 & 5000 & 0 \\ -5000 & 10000 & 0 & 0 & -3750 \\ -5000 & 0 & 10000 & -3334 & 4583 \\ 5000 & 0 & -3334 & 10000 & -4583 \\ 0 & -3750 & 4583 & -4583 & 10208 \end{bmatrix}$$

Table 6.5: *The 52 certified vectors in X_5^* and their certificates (continued).*

$(0,0,-1,1,1)$

$$\begin{bmatrix} 1000 & -500 & -500 & -500 & 0 \\ -500 & 1000 & 0 & 500 & -375 \\ -500 & 0 & 1000 & 150 & 458 \\ -500 & 500 & 150 & 1000 & -458 \\ 0 & -375 & 458 & -458 & 1021 \end{bmatrix}$$

$(1,0,-1,1,1)$

$$\begin{bmatrix} 1000 & -500 & 500 & -500 & 0 \\ -500 & 1000 & -500 & 500 & -375 \\ 500 & -500 & 1000 & -334 & 458 \\ -500 & 500 & -334 & 1000 & -458 \\ 0 & -375 & 458 & -458 & 1021 \end{bmatrix}$$

$(-1,1,-1,1,1)$

$$\begin{bmatrix} 1024 & -512 & -512 & 512 & 512 \\ -512 & 1024 & 160 & -568 & -640 \\ -512 & 160 & 1036 & -217 & 176 \\ 512 & -568 & -217 & 4603/4 & 28 \\ 512 & -640 & 176 & 28 & 1408 \end{bmatrix}$$

$(0,1,-1,1,1)$

$$\begin{bmatrix} 720 & -360 & -360 & -180 & 180 \\ -360 & 720 & 180 & -180 & -360 \\ -360 & 180 & 720 & 90 & 180 \\ -180 & -180 & 90 & 720 & -180 \\ 180 & -360 & 180 & -180 & 855 \end{bmatrix}$$

$(1,1,-1,1,1)$

$$\begin{bmatrix} 720 & -360 & 360 & 180 & -180 \\ -360 & 720 & -180 & -360 & -180 \\ 360 & -180 & 720 & 90 & 180 \\ 180 & -360 & 90 & 720 & -180 \\ -180 & -180 & 180 & -180 & 855 \end{bmatrix}$$

$(2,1,-1,1,1)$

$$\begin{bmatrix} 1024 & -512 & 512 & -512 & -512 \\ -512 & 1024 & -256 & -1024/15 & -128 \\ 512 & -256 & 1024 & -256 & 128 \\ -512 & -1024/15 & -256 & 783616/675 & 512/15 \\ -512 & -128 & 128 & 512/15 & 1408 \end{bmatrix}$$

$(-1,-1,0,1,1)$

$$\begin{bmatrix} 720 & -360 & 360 & 90 & 270 \\ -360 & 720 & -187 & 225 & 134 \\ 360 & -187 & 720 & -94 & 266 \\ 90 & 225 & -94 & 720 & -135 \\ 270 & 134 & 266 & -135 & 809 \end{bmatrix}$$

$(0,-1,0,1,1)$

$$\begin{bmatrix} 1000 & -500 & 461 & -426 & -74 \\ -500 & 1000 & -432 & 539 & 412 \\ 461 & -432 & 1013 & -454 & 83 \\ -426 & 539 & -454 & 1096 & -222 \\ -74 & 412 & 83 & -222 & 1000 \end{bmatrix}$$

$(-1,0,0,1,1)$

$$\begin{bmatrix} 24 & -12 & 12 & 12 & 12 \\ -12 & 24 & -6 & -6 & -6 \\ 12 & -6 & 24 & 4 & 10 \\ 12 & -6 & 4 & 24 & -3 \\ 12 & -6 & 10 & -3 & 25 \end{bmatrix}$$

$(0,0,0,1,1)$

$$\begin{bmatrix} 1024 & -512 & 512 & 512 & -512 \\ -512 & 1024 & -256 & -256 & 256 \\ 512 & -256 & 1024 & 160 & -64 \\ 512 & -256 & 160 & 1036 & -664 \\ -512 & 256 & -64 & -664 & 1072 \end{bmatrix}$$

$(1,0,0,1,1)$

$$\begin{bmatrix} 1024 & -512 & 512 & -512 & -512 \\ -512 & 1024 & -256 & 256 & 256 \\ 512 & -256 & 1024 & -352 & -64 \\ -512 & 256 & -352 & 1036 & -152 \\ -512 & 256 & -64 & -152 & 1072 \end{bmatrix}$$

$(0,1,0,1,1)$

$$\begin{bmatrix} 1024 & -512 & 512 & 128 & 384 \\ -512 & 1024 & -256 & -448 & -576 \\ 512 & -256 & 1024 & -128 & 384 \\ 128 & -448 & -128 & 1024 & -192 \\ 384 & -576 & 384 & -192 & 1152 \end{bmatrix}$$

Table 6.6: *The 52 certified vectors in X_5^* and their certificates (continued).*

$(1,1,0,1,1)$

$$\begin{bmatrix} 1000, -500, 298, -320, -180 \\ -500, 1000, -500, -169, -285 \\ 298, -500, 1000, -160, 324 \\ -320, -169, -160, 1000, -171 \\ -180, -285, 324, -171, 1000 \end{bmatrix}$$

$(-2,-1,1,1,1)$

$$\begin{bmatrix} 720 & -360 & 360 & 360 & 360 \\ -360 & 720 & -217 & 90 & 90 \\ 360 & -217 & 720 & -69 & -62 \\ 360 & 90 & -69 & 909 & 174 \\ 360 & 90 & -62 & 174 & 911 \end{bmatrix}$$

$(-1,-1,1,1,1)$

$$\begin{bmatrix} 3628800 & -1814400 & 1814400 & 1360800 & -1311192 \\ -1814400 & 3628800 & -907200 & 680400 & 1867520 \\ 1814400 & -907200 & 3628800 & -680400 & -655596 \\ 1360800 & 680400 & -680400 & 4486538 & -1193454 \\ -1311192 & 1867520 & -655596 & -1193454 & 3735040 \end{bmatrix}$$

$(0,-1,1,1,1)$

$$\begin{bmatrix} 720 & -360 & -360 & -180 & 180 \\ -360 & 720 & 180 & 360 & 180 \\ -360 & 180 & 720 & -180 & -90 \\ -180 & 360 & -180 & 834 & -169 \\ 180 & 180 & -90 & -169 & 720 \end{bmatrix}$$

$(-1,0,1,1,1)$

$$\begin{bmatrix} 3628800 & -1814400 & 1814400 & 453532 & 1360799 \\ -1814400 & 3628800 & -1247400 & 1134033 & -1773990 \\ 1814400 & -1247400 & 3628800 & -1282871 & 817098 \\ 453532 & 1134033 & -1282871 & 3986944 & -1666767 \\ 1360799 & -1773990 & 817098 & -1666767 & 3628800 \end{bmatrix}$$

$(1,-1,1,1,1)$

$$\begin{bmatrix} 720 & -360 & -360 & -360 & -360 \\ -360 & 720 & 143 & 450 & 450 \\ -360 & 143 & 720 & -69 & -62 \\ -360 & 450 & -69 & 909 & 174 \\ -360 & 450 & -62 & 174 & 912 \end{bmatrix}$$

$(0,0,1,1,1)$

$$\begin{bmatrix} 1024 & -512 & -256 & 512 & -256 \\ -512 & 1024 & -256 & 128 & 128 \\ -256 & -256 & 1024 & -704 & -320 \\ 512 & 128 & -704 & 1408 & -320 \\ -256 & 128 & -320 & -320 & 1024 \end{bmatrix}$$

$(1,0,1,1,1)$

$$\begin{bmatrix} 1024 & -512 & -512 & -256 & -256 \\ -512 & 1024 & 64 & 320 & 128 \\ -512 & 64 & 1072 & -304 & -256 \\ -256 & 320 & -304 & 1072 & -128 \\ -256 & 128 & -256 & -128 & 1024 \end{bmatrix}$$

Table 6.7: *The 52 certified vectors in X_5^* and their certificates (continued).*

$(-1,1,1,1,1)$

$$\begin{bmatrix} 10000 & -5000 & 5000 & 5000 & 5000 \\ -5000 & 10000 & -5545 & -3417 & -4687 \\ 5000 & -5545 & 11091 & 421 & 988 \\ 5000 & -3417 & 421 & 10000 & 282 \\ 5000 & -4687 & 988 & 282 & 11492 \end{bmatrix}$$

$(0,1,1,1,1)$

$$\begin{bmatrix} 1024 & -512 & 256 & 256 & 0 \\ -512 & 1024 & -512 & -128 & -384 \\ 256 & -512 & 1024 & -320 & -192 \\ 256 & -128 & -320 & 1024 & -192 \\ 0 & -384 & -192 & -192 & 1152 \end{bmatrix}$$

$(1,1,1,1,1)$

$$\begin{bmatrix} 720 & -360 & -360 & 360 & -360 \\ -360 & 720 & 45 & -180 & -90 \\ -360 & 45 & 735 & -310 & -13 \\ 360 & -180 & -310 & 720 & -368 \\ -360 & -90 & -13 & -368 & 953 \end{bmatrix}$$

$(2,1,1,1,1)$

$$\begin{bmatrix} 24 & -12 & -12 & -12 & -12 \\ -12 & 24 & 1 & 6 & -3 \\ -12 & 1 & 25 & 2 & 0 \\ -12 & 6 & 2 & 24 & 0 \\ -12 & -3 & 0 & 0 & 32 \end{bmatrix}$$

$(-1,2,1,1,1)$

$$\begin{bmatrix} 10000 & -2500 & 5000 & 5000 & 5000 \\ -2500 & 10000 & -5937 & -5937 & -5937 \\ 5000 & -5937 & 11875 & 2494 & 2494 \\ 5000 & -5937 & 2494 & 11886 & 2500 \\ 5000 & -5937 & 2494 & 2500 & 11886 \end{bmatrix}$$

$(0,2,1,1,1)$

$$\begin{bmatrix} 720 & -220 & -315 & 360 & 360 \\ -220 & 720 & -230 & -436 & -436 \\ -315 & -230 & 810 & -177 & -177 \\ 360 & -436 & -177 & 878 & 175 \\ 360 & -436 & -177 & 175 & 878 \end{bmatrix}$$

$(1,2,1,1,1)$

$$\begin{bmatrix} 100000 & -34598 & -32701 & 1897 & 1897 \\ -34598 & 100000 & -32701 & -44671 & -44671 \\ -32701 & -32701 & 100000 & -7226 & -7226 \\ 1897 & -44671 & -7226 & 100000 & 1313 \\ 1897 & -44671 & -7226 & 1313 & 100000 \end{bmatrix}$$

$(2,2,1,1,1)$

$$\begin{bmatrix} 1000000 & -462748 & -268626 & -312920 & -500000 \\ -462748 & 1000000 & -268625 & -248128 & -161557 \\ -268626 & -268625 & 1000000 & 28142 & 37252 \\ -312920 & -248128 & 28142 & 1065817 & 112047 \\ -500000 & -161557 & 37252 & 112047 & 1248670 \end{bmatrix}$$

Table 6.8: *The 52 certified vectors in X_5^* and their certificates (continued).*

(-2, -2, -1, 1, 1)	(-1, -2, -1, 1, 1)
(-1, -1, 0, -1, 1)	(-1, 0, -1, -1, 1)
(-1, 1, -1, -1, 1)	(2, 1, -1, -1, 1)
(0, -1, 0, -1, 1)	(0, 0, -1, -1, 1)
(0, 1, -1, -1, 1)	(0, 1, 0, -1, 1)
(1, -2, -1, 1, 1)	(1, 0, -1, -1, 1)
(1, 1, -1, -1, 1)	(1, 1, 0, -1, 1)

Table 6.9: *The 14 vectors provably not in X_5^*.*

(-2, -1, -1, -1, 1)
(-1, -1, -1, -1, 1)
(-1, 0, 0, -1, 1)
(0, -1, -1, -1, 1)
(0, 0, 0, -1, 1)
(1, -1, -1, -1, 1)
(1, 0, 0, -1, 1)
(0, -2, -1, 1, 1)

Table 6.10: *The 8 vectors probably not in X_5^* for which semidefinite programming can not decide.*

Bibliography

[KZ73] A. Korkine and G. Zolotareff, Sur les formes quadratiques. *Mathematische Annalen*, vol. 6, pp. 366–389 (1873).

[Nov83] N. V. Novikova, Korkin–Zolotarev reduction domains of positive quadratic forms in $n \leq 8$ variables and a reduction algorithm for these domains. *Soviet Mathematics. Doklady*, vol. 27, no. 3, pp. 557–560 (1983). Translated from *Doklady Akademii Nauk SSSR, Mathematics*.

[PR01] V. Powers and B. Reznick, A new bound for Pólya's Theorem with applications to polynomials positive on polyhedra. *Journal of Pure and Applied Algebra*, vol. 164, pp. 221–229 (2001).

[PZ06] R. A. Pendavingh and S. H. M. van Zwam, New Korkin–Zolotarev Inequalities(2006). In preparation.

[RB79] S. S. Ryshkov and E. P. Baranovskii, Classical methods in the theory of lattice packings. *Russian Math. Surveys*, vol. 34, no. 4, pp. 1–68 (1979). Translated from *Uspekhi Mat. Nauk*.

[Sho] V. Shoup, NTL: A Library for doing Number Theory. Software package. Available from http://www.shoup.net/ntl/.

[Stu99] J. F. Sturm, Using SeDuMi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, vol. 11–12, pp. 625–653 (1999). Software and manual available from http://sedumi.mcmaster.ca/.

[SV] A. Schürmann and F. Vallentin, Rigorous MAXDET: software for finding rational solutions to semidefinite programming problems. Available from http://fma2.math.uni-magdeburg.de/~latgeo/.

[SV06] A. Schürmann and F. Vallentin, Computational approaches to lattice packing and covering problems. *Discrete and Computational Geometry*, vol. 35, no. 1, pp. 73–116 (2006).

[VBW] L. Vandenberghe, S. Boyd, and S.-P. Wu, MAXDET: software for determinant maximization problems. Software package. Available from http://www.stanford.edu/~boyd/MAXDET.html.

[Zwa05] S. H. M. van Zwam, *Properties of Lattices, a Semidefinite Programming Approach*. Master's thesis, Eindhoven University of Technology (2005).