Project Number: MA-WJM-4801

A Study of Linear Programming Bounds for Spherical Codes

A Major Qualifying Project Report

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

Aaron J. Nahabedian

Nadia S. Zahid

Date: April 30, 2009

Approved:

Professor William J. Martin, Major Advisor

Contents

Chapter 1

Introduction

This Major Qualifying Project deals with linear programming bounds for spherical codes. In this chapter, we discuss the problem informally, drawing connections to a variety of mathematical problems and applications. Sections 1.3, 1.4, and 1.5 discuss equiangular lines, kissing numbers, and a very special 4-dimensional polytope, respectively. In Section 1.2 the linear programming technique developed by Delsarte, Goethals, and Seidel in 1977 is discussed, and a brief overview of the report is given.

1.1 A Natural Geometric Problem

Suppose you needed to set up radio towers around the span of the Earth, and the only way they would be able to function properly is if they are placed 1,000 miles apart from each other, when measured by the length of a secant joining them. We model this by a 3-D coordinate system with the origin at the center of the Earth. It follows that the unit vectors in three-dimensional space, representing the displacement from the origin to the base of two of these towers, form an angle, which we'll view as the angle between those towers. We can also consider the standard dot product between these two vectors. These three measurements, the Euclidean distance between two towers along a secant through the Earth, the angle formed by the unit vectors in three-dimensional space, and the inner product of these two vectors, are essentially the same thing: any one of them is computable from any other. In this report we freely switch from one to the other.

How many radio towers can you place around the globe of the Earth while achieving the given condition? This is an optimization problem. The answer to this question lies within what we call 'bounds for spherical codes'. In order to compute the linear programming bounds, the inner product is the most useful of the three measures described in the above paragraph.

The Euclidean distance discussed above is the distance between any two towers, P and Q on a secant through the Earth containing those two points. The following is the equation that allows for the computation of the distance between two points P and Q in Euclidean

n-space: if

$$P = (p_1, p_2, p_3, ..., p_n)$$
$$Q = (q_1, q_2, q_3, ..., q_n)$$

then the distance between P and Q is:

$$d(P, Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2}$$

Suppose the points P and Q represent two separate radio towers and the point R represents the center of the Earth. The angle θ between these two towers is formed by the vectors RP and RQ. We will denote the vector RP as ξ and RQ as η in our calculations. The inner product can be calculated using

$$\langle \xi, \eta \rangle = |\xi||\eta| \cos \theta$$

so that

$$\theta = \arccos\left(\frac{\langle \xi, \eta \rangle}{|\xi||\eta|}\right).$$

The distance $d'(P, Q)$ from P to Q along the surface of the Earth is

$$d'(P, Q) = \frac{\theta}{2\pi} C \geq d(P, Q)$$

where C is the circumference of the Earth. As discussed earlier these three elements are closely related to each other as evidence by the above equations.

1.2 The Linear Programming Technique

Linear programming is a technique used to optimize a linear objective function when subject to a finite number of linear equality and inequality constraints. Linear programming is a way in which to achieve the best possible outcome in a given mathematical model. A linear program (LP) can be expressed as follows,

$$\begin{array}{rl} \min & c^T x \\ \text{s.t.} & Ax \leq b \end{array}$$

where x represents the vector of variables, c and b are vectors of coefficients, and A is a matrix of coefficients. The expression that is minimized is the objective function and the inequalities are the constraints which specify a polyhedron over which the objective function is to be optimized. Solving a linear programming problem gives a solution vector x and an objective value $c^T x$.

The founders of this subject are Leonid Kantorovich, who developed linear programming problems in 1939, George Dantzig, who published the simplex method in 1947, and John von Neumann, who developed the theory of duality in the same year. Leonid Khachiyan was the first to show a linear programming problem to be solvable in polynomial time in 1979.

Narendra Karmarkar introduced the interior point method for solving linear programming problems in 1984.

Linear programming is useful for many reasons. Practical problems in operations research can often be expressed as linear programming problems. Also a number of algorithms for network flow problems, and multi-commodity flow problems work by solving LP problems. They are also extensively used in microeconomics and production management.

The linear programming technique used in this project is really of a rather different nature: It became famous in the 1970's and was formulated by Delsarte, Goethals, and Seidel [4] in 1977. This followed Philippe Delsarte's LP approach [3] to find the strongest bounds for discrete q-ary codes [3]. He developed this approach which allowed for the very efficient computation of upper bounds on the cardinalities of binary codes, more generally subsets with restrictions in any association scheme. This approach was then adapted to a quite different situation, namely spherical codes, by Delsarte, Goethals, and Seidel [4]. The study of these linear programming bounds for spherical codes is the subject of our report, which includes our understanding the linear programming approach developed by Delsarte, Goethals, and Seidel, exploring spherical codes, developing a program in Mathematica which implements their linear programming approach, and using that program to find upper bounds on the size of spherical codes.

1.3 Equiangular Lines

The problem discussed in this section has a surprising connection to graphs, although it may not be an obvious one at first, and is an important problem in relevance to algebraic graph theory.

We begin by defining a *simplex*. A *simplex* in a metric space S, having distance function d is a subset of S such that $d(x, y)$ between any two distinct points of the simplex is the same. In $\mathbb{R}^d$ a simplex can contain at most $d+1$ elements. Take $S = \{e_i : i = 1, \cdots, n\}$ and observe this configuration lies in the $(n-1)$-dimensional space $\{x \mid \langle 1, x\rangle = 1\}$ where 1 is the all-ones vector; so we really have $d+1$ vectors in $\mathbb{R}^d$ where $d = n-1$. When considering the same problem in real elliptic space, finding the maximum number of points in a simplex is not as simple. When looking at this space, the points are represented by the lines through the origin of $\mathbb{R}^d$, and the distance between two lines is the square of the sine of the angle that separates them. Therefore, a simplex in elliptic space is a set of lines in $\mathbb{R}^d$ such that the angle between any two distinct lines is equal. This is how we define a set of *equiangular lines.*

A line in $\mathbb{R}^d$ can be represented by the span of a unit vector x. Therefore, a set of lines can be represented by a set of unit vectors $\Omega = \{x_1, ..., x_n\}$. Ω is not unique, because $-x$ represents the same line as x does. More generally, we may apply any orthogonal matrix to the vectors in Ω to get an equivalent set. So, given a set Ω of unit vectors representing a set of equiangular lines where the angle separating any two distinct lines is θ, the following is true when $i \neq j$,

$$\langle x_i, x_j\rangle = \pm\cos(\theta)$$

. for some fixed θ.

An example of this is obtained in $\mathbb{R}^8$ by taking the 28 unit vectors,

$$x_i^T = \sqrt{\frac{1}{24}}\,(3, 3, -1, -1, -1, -1, -1, -1)$$

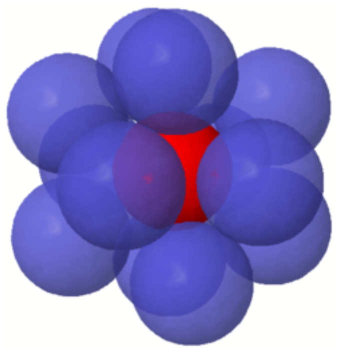
which is formed by any two entries being 3 and the remaining six being -1. For $i \neq j$,

$$\langle x_i, x_j \rangle = \pm\frac{1}{3},$$

where the positive sign is taken if and only if x_i and x_j have the entry of 3 in the same coordinate. So, this forms a set of 28 equiangular lines in $\mathbb{R}^8$. Note that all of these vectors are orthogonal to $\mathbf{1}$, which means that they all lie in the 7-dimensional subspace $\mathbf{1}^\perp$. This means that there are 28 equiangular lines in $\mathbb{R}^7$. The best upper bounds for spherical codes come from linear programming. In projective space, we use the Jacobi polynomials which are essentially the even Gegenbauers $Q_0(x), Q_2(x), Q_4(x), \ldots$ with substitution $z = x^2$. The best known method for obtaining a system of equiangular lines in Euclidean space is described in an article written by Dominique de Caen, which also cites Seidel's work. He was able to find that for each $d = 3 \cdot 2^{2t-1}$, with t being any positive integer, there exists a set of $\frac{2}{9}(d+1)^2$ equiangular lines in Euclidean d-space, where we observe the corresponding upper bound of $\frac{d(d+3)}{2}$ for this number. This subject is further discussed in Section 2.4.

1.4 The Kissing Number Problem in 4 Dimensions

The kissing number problem in dimension d asks, "How many non-overlapping balls can touch a given ball, of the same size, at the same time?". The kissing number τ_d is the maximum number of such balls, excluding the central ball, in dimension d. Although it may seem easy, this problem is surprisingly hard. The answer to this question in dimensions one, two and three are classical and well known. The solution in dimension three is 12, as shown:



In 1979 the answers to this question in dimensions eight and twenty-four, were discovered based on the methods we will discuss in Section 2.5 [7]. However, it was proved that the bounds given by Delsarte's method [3] are not good enough to solve the problem in dimension four. This problem remained a mystery until in 2003 when Oleg Musin was able to find a solution to the problem by making some modifications to Delsarte's method.

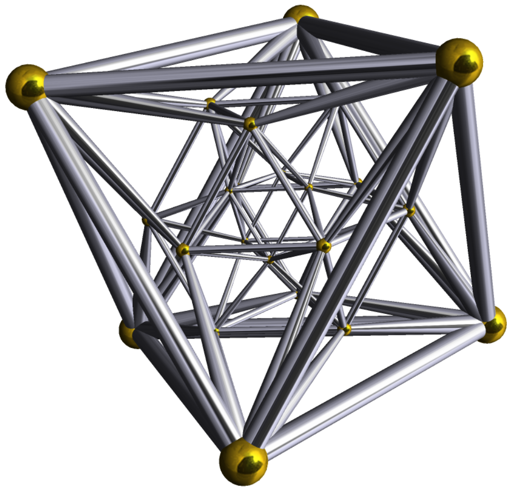
In 1979, A. Odlyzko and N. Sloane established that the kissing number in dimension four is between 24 and 25 [7]. It is possible to produce a packing of 24 spheres around a central sphere; an example of that is the 24-cell (and it is still not known if this configuration is unique). In this case, however, as in the three-dimensional case, there is a lot of space that is still left over. There is actually more space left over than for the case $d = 3$, which makes the situation even less clear.

For $d = 4$, the linear programming method does not work as well as we would like; the method yields that the bound for the kissing number is $\tau_4 \leq 26$, and nothing better than that. Oleg Musin was able to find a method which modifies the linear programming method in a very clever way which yields better bounds. Musin lived in Los Angeles at the time, but received his PhD in Mathematics from Moscow State University. Oleg Musin improved the upper bound from 25 to 24, proving that $\tau_4 = 24$ [7]. He was able to prove this through the use of a subtle trick. This is further discussed in Section 2.5.

The optimal kissing configuration for spheres in four dimensions is described as follows: There are 24 vectors having two zero components and two components equal to ± 1. The notation for this is the following: $(\pm 1, \pm 1, 0, 0)$ or $(\pm 1^2, 0^2)$. Each one of these vectors has length $\sqrt{2}$ and the minimum distance between an two of them is $\sqrt{2}$. When properly rescaled, these 24 points become the centers for a configuration of 24 non-overlapping unit spheres all tangent to the unit sphere centered at the origin. It also implies that the bound for the kissing number is at least 24. The convex hull of these 24 points yields the "24-cell", which is further discussed in the next section.

1.5 The 24-cell

The 24-cell is quite a beautiful convex regular 4-polytope and is also called an octaplex. It is composed of 24 octahedra cells with six meeting at each vertex, and three at each of it's edges. It also consists of 96 triangular faces, 96 edges, and 24 vertices. The figure below is an image of a 24-cell.



We now give another description of the 24-cell. The vertices of our 24-cell are centered at the origin in four-dimensional space and are of two types. First, they are 8 vertices which are obtained by permuting the entries of the vector

$$(\pm 1, 0, 0, 0)$$

and 16 vertices of the following form:

$$\left(\pm\frac{1}{2}, \pm\frac{1}{2}, \pm\frac{1}{2}, \pm\frac{1}{2}\right).$$

These 16 vertices are then split into two separate groups: those having an even number of minus signs and then those having an odd number of minus signs. Any two of these three groups of eight form a set of 16 vectors whose convex hull is isometric to a 4-cube. Since the configuration is known to be unique, there is an orthogonal matrix M such that the mapping $\overline{v} \longmapsto \sqrt{2}M\overline{v}$ sends these 24 vectors to the twenty-four vectors described at the end of the previous section.

Many examples of configurations in dimension three are explored in Appendix B.

Chapter 2

The Theory of Delsarte, Goethals and Seidel

The best bounds in coding theory are usually those which are obtained through linear programming techniques. These ideas were developed by Delsarte [3], and the extension of this theory to spherical codes was developed by Delsarte, Goethals, and Seidel [4] a few years later. In this chapter, we give an overview of the 1977 paper [4] by P. Delsarte, J.-M. Goethals and J. Seidel.

Gegenbauer polynomials form an important family of orthogonal polynomials and are central to the linear programming bound we study. They are discussed in the first section of this chapter. The next section of the chapter introduces spherical codes; in it we give some basic parameters of spherical codes and introduce notation. The linear programming approach developed by Delsarte, Goethals and Seidel is outlined in Section 2.3. The final section of the chapter includes examples where the linear programming approach is applied.

2.1 Gegenbauer Polynomials

Gegenbauer polynomials are one of the basic tools necessary to compute bounds for spherical codes. They are a family of orthogonal polynomials, denoted $\{Q_k(x) : k \in \mathbb{N}\}$ where each $Q_k(x)$ is in one variable x, which is defined for a fixed dimension, $d \geq 2$.

Theorem 2.1.1 ([4]) *For fixed $d \geq 2$ the* Gegenbauer polynomial *$Q_k(x)$ of degree k is defined by*

$$\ell_{k+1}Q_{k+1}(x) = xQ_k(x) - (1 - \ell_{k-1})Q_{k-1}(x),$$

$$\ell_k = \frac{k}{d + 2k - 2}, \qquad Q_0(x) = 1, \qquad Q_1(x) = dx.$$

According to this definition the next few Gegenbauer polynomials are:

$$2Q_2(x) = (d+2)(dx^2-1),$$

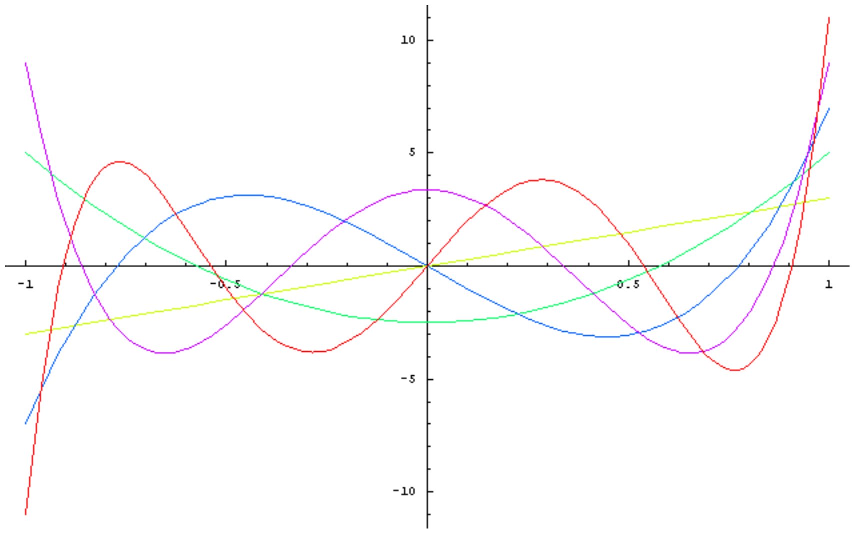
$$6Q_3(x) = d(d+4)((d+2)x^3-3x),$$
$$24Q_4(x) = d(d+6)((d+2)(d+4)x^4-6(d+2)x^2+3),$$
$$120Q_5(x) = d(d+2)(d+8)((d+4)(d+6)x^5-10(d+4)x^3+15x)$$

Example 2.1.2 ([4]) *The following is an example of a few polynomials in dimension 3:*

$$Q_1(x) = 3x,$$

$$Q_2(x) = \frac{1}{2}(3+2)(3x^2-1) = \frac{15}{2}x^2 - \frac{5}{2},$$
$$Q_3(x) = \frac{1}{6}(3)(3+4)((3+2)x^3-3x) = \frac{35}{2}x^3 - \frac{21}{2}x,$$
$$Q_4(x) = \frac{1}{24}(3)(3+6)((3+2)(3+4)x^4-6(3+2)x^2+3) = \frac{315}{8}x^4 - \frac{135}{4}x^2 + \frac{27}{8},$$
$$Q_5(x) = \frac{1}{120}(3)(3+2)(3+8)((3+4)(3+6)x^5-10(3+4)x^3+15x) = \frac{693}{8}x^5 - \frac{377}{4}x^3 + \frac{165}{8}x$$

The following is a graph of these polynomials:



We said that the Gegenbauer polynomials are "orthogonal polynomials". This means that these polynomials are pairwise orthogonal with respect to some prescribed inner product on the vector space of all polynomials. The sort of inner product we consider is built from a weight function as follows.

Definition 2.1.3 ([4]) *Let $[x_1, x_2]$ be an interval in the real line called the interval of orthogonality. Now let $W : [x_1, x_2] \to \mathbb{R}$ be a function on that interval; W must be strictly positive on the interior (x_1, x_2), and it can also be zero or go to infinity at the end points. It is also necessary that W satisfies the requirement that, for any polynomial f, the integral $\int_{x_1}^{x_2} f(x)W(x)dx$ is finite. A function W meeting these requirements is called a* weight function.

Now, given any x_1 and x_2, along with a function W, we will define an inner product on polynomials by

$$\langle f, g\rangle = \int_{x_1}^{x_2} f(x)g(x)W(x)dx.$$

This operation is easily seen to be linear in each of its arguments. The two polynomials, f and g, are orthogonal *when their inner product is zero.*

For our application, we consider polynomials on the interval $[x_1, x_2] = [-1, 1]$ and weight function $W(x) = (1 - x^2)^{\frac{d-3}{2}}$. We find,

$$\int_{-1}^{1} Q_k(x)Q_i(x)(1 - x^2)^{(d-3)/2}\, \mathrm{d}x = a_d Q_k(1)\delta_{k,i}$$

where a_d is some positive constant, and $\delta_{k,i}$ is the Kronecker delta.

Now there turns out to be one $Q_k(x)$ of degree k for each k. So $\{Q_k(x)\}_{k=0}^{\infty}$ is a basis for the vector space of polynomials. Therefore, each $F(x)$ is uniquely expressible as a linear combination of $Q_0, Q_1, Q_2, \ldots$

The *Gegenbauer expansion* associated with the polynomial $F(x)$ is defined as follows:

$$F(x) = \sum_{k=0}^{\infty} f_k Q_k(x),$$

for the defined Gegenbauer coefficients f_k.

2.2 Spherical Codes

In this section we describe some basic parameters of spherical codes. Spherical codes are of course essential to our study and we now define them: a spherical code is simply any non-empty finite subset of the unit sphere in $\mathbb{R}^d$.

A finite non-empty set X of unit vectors in Euclidean space $\mathbb{R}^d$ has several characteristics of interest to us: the dimension $d(X)$ of the space spanned by X, its cardinality $n = |X|$, and degree $s(X)$.

The number of values assumed by the inner product between distinct vectors in X is the *degree* $s(X)$:

$$s(X) = |A(X)|, \qquad A(X) = \{\langle \xi, \eta\rangle; \xi \neq \eta \in X\}.$$

Sets X may have the property that $A(X)$ is contained in a prescribed subset A of the interval $[-1, 1)$. Sets who have this property are what we call *spherical A-codes.* We seek upper bounds on the size $n = |X|$ of an A-code X. The most powerful known upper bounds for the size of spherical A-codes are derived using the concepts of linear programming and Gegenbauer polynomials.

Definition 2.2.1 ([4]) *Let A be a subset of the interval $[-1, 1)$. A spherical A-code, is a non-empty subset X of the unit sphere in $\mathbb{R}^d$ satisfying $\langle \xi, \eta \rangle \in A$, for all $\xi \neq \eta \in X$.*

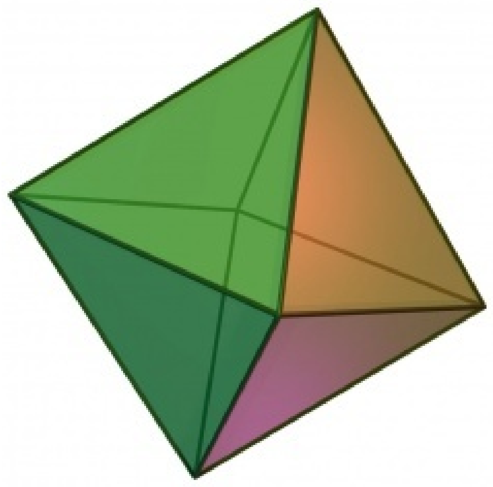
Through this definition, it is interpreted that a spherical A-code can be described as a set of unit vectors who have angles from the set $\{\arccos \xi \mid \xi \in A\}$.

Example 2.2.2 ([4]) *A set X is* antipodal *if for all ξ in X, $-\xi$ is also in X. This yields that $A'(X) = -A'(X)$, for $A' := A \cup \{1\}$. The antipodal codes on Ω_d are in one-to-one correspondence with the sets of lines through the origin of $\mathbb{R}^d$.*

Example 2.2.3 *The following is an example in three dimensions. Let $A = [-1, 0]$, so the angle between any two points will be between π and $\frac{\pi}{2}$. The optimal configuration on this interval is the octahedron:*

$$\{(-1,0,0),(1,0,0),(0,-1,0),(0,1,0),(0,0,-1),(0,0,1)\}$$

Note that the inner product between any two vectors will be either -1 or 0. Both of these values lie within the interval A.



2.3 Linear Programming Approach

Delsarte, Goethals, and Seidel developed the following approach for finding linear programming bounds.

Definition 2.3.1 ([4]) *A polynomial $F(x)$ is said to be* compatible *with the set A when for all $\alpha \in A$, $F(\alpha) \leq 0$.*

Theorem 2.3.2 ([4]) *Let $F(x)$, with Gegenbauer coefficients $f_0 > 0$ and $f_k \geq 0$ for all k, be compatible with the set A. Then the cardinality, n, of any A-code X satisfies $n \leq F(1)/f_0$.*

We omit the proof in order to focus on the application of this theorem.

Suppose your have some function F with non-negative Gegenbauer coefficients and $f_0 = 1$, such that F is non-positive within A. If this is true, then F evaluated at 1 is an upper bound on the size of a spherical A-code. The goal is to minimize $F(1)$ while keeping $F(x)$ compatible with A.

Using Theorem 2.3.2, we can generate a linear programmming problem ("linear program" or simply "LP") as follows:

$$\begin{array}{lrcl}
\text{min} & \frac{F(1)}{f_0} & & \\
\text{s.t.} & F(\alpha) & \leq & 0 \quad \forall \alpha \in A \\
f_0 > 0, & f_k & \geq & 0 \quad \text{for } k = 1, 2, \ldots
\end{array}$$

Notice that there are uncountably many constraints and countably many variables. This LP is further analyzed in Section 3.1.

The following example demonstrates Theorem 2.3.2.

Example 2.3.3 ([4]) *Let A be a subset of the interval $[-1, \beta]$, where β is within the range of $-1 \leq \beta < 0$. The polynomial $F(x) = x - \beta$ is compatible with A, and the first few Gegenbauer coefficients satisfy these conditions: $f_0 = -\beta > 0$, and $f_1 = 1/d > 0$. Recall that Theorem 2.3.2 states $|X| \leq F(1)/f_0$ for any A-code since our $F(x)$ is compatible with A. In this case we obtain the bound*

$$|X| \leq \frac{(1-\beta)}{-\beta} = 1 - \frac{1}{\beta}.$$

For example when $d = 2$ and $\beta = -1/2$, we get $|X| \leq 3$ and the optimal solutions are equilateral triangles. More generally, an A-code of the given dimension $r \leq d$ achieves this bound if and only if it is an r-dimensional regular simplex, with $\beta = -1/r$.

2.4 An In-Depth Look at Equiangular Lines

De Caen describes the construction of the equiangular lines as follows [2]: let G be a graph, having Seidel adjacency matrix S. This means that $S_{xy} = -1$ when vertices x and y are adjacent to each other, $S_{xy} = 1$ when x and y are distinct and non-adjacent, and $S_{xx} = 0$ for all x. If θ is the smallest eigenvalue of S, then it follows that $M := I - \frac{1}{\theta}S$ is positive semidefinite. Also, it has rank $d = n - m$ where n is the number of vertices and m is the eigenvalue multiplicity of θ. Therefore, M can now be represented as a Gram matrix of n unit vectors $x_1, ..., x_n$ in real d-space. The inner product, $\langle x_i, x_j \rangle = \pm\frac{1}{\theta}$ for i and j distinct. The lines, or 1-dimensional subspaces, which are spanned by these x_i's will have constant pairwise angle $\arccos(\frac{1}{\theta})$.

By simply reversing this process we are able to find a large equiangular set of lines in Euclidean space by finding a graph whose Seidel adjacency matrix has the smallest eigenvalue of large multiplicity.

Theorem 2.4.1 ([2]) *For each $d = 3 \cdot 2^{2t-1}$, with t any positive integer, there exists an equiangular set of $\frac{2}{9}(d+1)^2$ lines in Euclidean d-space.*

The graphs used in the construction of equiangular lines refer to the theory of quadratic forms over $GF(2)$, which is discussed in De Caen and Van Dam's 1999 article, "Association schemes related to Kasami codes and Kerdock sets" in the journal Designs, Codes, and Cryptography.

Let V be a vector space over $GF(2)$. If $Q : V \to GF(2)$ is a quadratic form, then its polarization $B(x, y) := Q(x + y) + Q(x) + Q(y)$ is an alternating bilinear form. It is also important to note that B can only be non-singular if V has even dimension. Therefore, assume that $dim(V) = 2t$ for some positive integer t. If Q polarizes to a non-singular B, then Q must be of one of two types $\chi(Q) = \pm 1$, where Q has exactly $2^{2t-1} + \chi(Q)2^{t-1}$ zeroes. Now we let $B_1, B_2, ..., B_r$ be a set of alternating bilinear forms on V. If $B_i + B_j$ is non-singular for all $i \neq j$ then the set is called *non-singular*. A non-singular set has $r \leq 2^{2t-1}$, and when equality holds it is called a *Kerdock set*. These maximal non-singular sets exist for all t.

Now we can describe the graphs that occur when constructing equiangular lines. Let K be a Kerdock set of alternating forms on V, where $dim(V) = 2t$. The graph G_t will have as vertex set all pairs (B, Q) where B belongs to K and Q polarizes to B. Two vertices (B, Q) and (B', Q') are declared adjacent precisely when $B \neq B'$ and $\chi(Q + Q') = -1$. The eigenvalues of the Seidel adjacency matrix $S(G_t)$ are as follows:

$$\theta_1 = 2^{3t-1} + 2^{2t} - 1$$

with multiplicity one;

$$\theta_2 = 2^{3t-1} + 2^{t} - 1$$

with multiplicity $2q - 1$ where $q := 2^{2t-1}$;

$$\theta_3 = 2^{2t} + 2^{t} - 1$$

with multiplicity $q - 1$;

$$\theta_4 = -2^{t} - 1$$

with multiplicity $(q-1)(2q-1)$.

Example 2.4.2 *The following is an example of the values of θ_1 up to θ_4 when t is equal to 2:*

$$\theta_1 = 2^{3(2)-1} + 2^{2(2)} - 1 = 47$$

with multiplicity one;

$$\theta_2 = 2^{3(2)-1} + 2^2 - 1 = 35$$

with multiplicity $2q-1$ where $q := 2^{2(2)-1} = 8$;

$$\theta_3 = 2^{2(2)} + 2^2 - 1 = 19$$

with multiplicity $q-1$;

$$\theta_4 = -2^2 - 1 = 3$$

with multiplicity $(q-1)(2q-1)$.

The ordinary adjacency matrix A and the Seidel matrix S are related by the following equation $S = J - I - 2A$.

We can now see that the eigenvalue $\theta = \theta_4$ is the smallest eigenvalue of $S(G_t)$ and it has a very large multiplicity. The rank of $M = I - \frac{1}{\theta}S$ is $d = 3q-1$ and the graph has $2q^2 = \frac{2}{9}(d+1)^2$ vertices. Following the procedure described above, we can now find a set of $\frac{2}{9}(d+1)^2$ equiangular lines in Euclidean d-space, whenever $d = 3q - 1 = 3 \cdot 2^{2t-1} - 1$ for some positive integer t. This completes our summary of the construction of De Caen.

The best constructions come from regular two-graphs where the Seidel adjacency matrix has just two distinct eigenvalues. Thus far there has been nothing better asymptotically than multiplying a constant and $d\sqrt{d}$.

Example 2.4.3 *One way that equiangular lines are formed is when the internal angles of a polyhedron are congruent to each other. Equiangular lines exist in various polyhedra, such as the tetrahedron whose diagonal lines are equiangular. There is an image of the tetrahedron in Appendix B.*

2.5 Musin's Theorem

The kissing number, τ_d, is the maximum number of spheres of radius 1 that can touch the unit sphere simultaneously in d-dimensional Euclidean space.

Musin's improvement to Delsarte's methods [3] allows the function $F(t)$ in Theorem 2.3.2 to creep above the x-axis within the interval $A = [-1, \frac{1}{2}]$, but only "opposite to the given sphere", which means only when close to $t = 1$.

The result is as follows:

Theorem 2.5.1 ([7]) *Fix a parameter t_0 in the range $-1 \leq t_0 < -\frac{1}{2}$. If,*

$$F(t) = \sum_{k \geq 0} f_k Q_k(t)$$

is a non-negative combination of Gegenbauer polynomials ($f_k \geq 0$ for all k, with $f_0 > 0$) and if $F(t) \leq 0$ holds for all $t \in [t_0, \frac{1}{2}]$, while $F' < 0$ for $t \in [-1, t_0]$, then the kissing number for $\mathbb{R}^d$ is bounded by

$$\tau_d \leq \frac{1}{f_0} \max\{h_0, h_1, ..., h_\mu\},$$

where h_μ is the maximum of

$$F(1) + \sum_{j=1}^{m} F(\langle e_1, y_j \rangle)$$

over all configurations of $m \leq \mu$ unit vectors y_j in the spherical cap given by $\langle e_1, y_j \rangle \leq t_0$ whose pairwise scalar products are at most $\frac{1}{2}$. Here μ denotes the maximal number of points that fit into the spherical cap.

The following is a sketch of Musin's proof for $\tau_d < 25$.

Musin produced a polynomial of degree 9 satisfying the assumptions of his Theorem 2.5.1 with $t_0 \approx -0.608$:

$$\begin{aligned} F(t) &= Q_0(t) + 2Q_1(t) + 6.12Q_2(t) + 3.484Q_3(t) + 5.12Q_4(t) + 1.05Q_5(t) \\ &= 53.76t^9 - 107.52t^7 + 70.56t^5 + 16.384t^4 - 9.832t^3 - 4.128t^2 - 0.434t - 0.016. \end{aligned}$$

This was found through discretization and linear programming, and these methods have already been used by A. Odlyzko and N. Sloane.

Now, in order to evaluate h_m, the arrangements of m points $y_1, ..., y_m$ in the spherical cap must be considered, $C_0 := \{y \in S^3 : \langle e_1, y \rangle \leq t_0\}$. The points have a minimum angle of 60°; we have $\pi/3 = \arccos(\frac{1}{2})$ given by $\langle y_i, y_j \rangle \leq \frac{1}{2}$, and this distance is larger than the radius $\arccos(-t_0)$ of the spherical cap. We know that in an optimal arrangement for a given m we cannot move one or more of the points towards the center of the cap while maintaining the "minimum distance" requirement, because of the monotonicity assumption on $F(t)$. Musin derives strong conditions from this on the combinatorics of optimal configurations.

An example is for $m \geq 1$ the center $-e_1$ of the spherical cap is contained in the (spherical) convex hull of the m points y_i. When $m \geq 2$ each point has at least one other point at a distance of exactly $\pi/3$. This yields that:

$$h_0 = F(1) = 18.774$$

$$h_1 = F(1) + F(-1) = 24.48$$

$$h_2 = max_{\phi \leq \pi/3} \left\{ F(1) + F(-\cos(\phi)) + F\left(-\cos\left(\frac{\pi}{3} - \phi\right)\right) \right\}$$

and it follows that

$$h_2 \approx 24.8644.$$

Oleg Musin's breakthrough in this method is a piece of great progress related to the packing of spheres in high-dimensional space.

Chapter 3

A Practical Reformulation of the Linear Program

Now that a background on the theory of Delsarte, Goethals and Seidel has been presented, we can begin to explain how we adapt that theory into a problem that can be solved by a computer. The linear programming problem (LP) that they derive has infinitely many variables and constraints. In order to write a program to find upper bounds on A-codes, we need to develop a way to approximate this LP with only finitely many variables and constraints.

In this chapter we examine the LP derived in Section 2.3. We reformulate this LP so that it can be solved using standard software. Examples are provided to show the limitations inherent in the reformulation, and techniques are developed to avoid them.

3.1 Analysis of the Delsarte, Goethals and Seidel LP

In order to construct an optimization problem that will give upper bounds on the size, n, of a spherical A-code, we use Theorem 2.3.2. Without loss of generality, we may set $f_0 = 1$ so that $n \leq \frac{F(1)}{f_0} = F(1)$. We now get the following LP:

$$\begin{array}{rrcl} \min & F(1) & & \\ \text{s.t.} & F(\alpha) & \leq & 0 \quad \forall \alpha \in A \\ f_0 = 1, & f_k & \geq & 0 \quad \text{for } k = 1, 2, \ldots \end{array}$$

with $F(x) = \sum_{k \geq 0} f_k Q_k(x)$, where $Q_k(x)$ is the k^{th} Gegenbauer polynomial for a given dimension d. The problem with solving this LP directly is that it has countably many variables $(f_0, f_1, f_2, ...)$ and, in a general application, uncountably many constraints (one constraint for each point $\alpha \in A$). In the following sections we show how to restrict these to finite amounts. By doing this we may lose both feasibility and optimality, only the former of which we are able to repair.

3.2 Restriction of Variables

Any optimal polynomial will be of finite degree, but this degree is not known *a priori*. This is why it is necessary to have infinitely many variables in the LP. To get an LP with finitely many variables, we simply restrict the number of variables we put into the LP by setting $f_k = 0$ for k greater than some given m. The LP then becomes:

$$\begin{array}{rrcl}
\text{min} & F(1) & & \\
\text{s.t.} & F(\alpha) & \leq & 0 \quad \forall \alpha \in A \\
f_0 = 1, & f_k & \geq & 0 \quad \text{for } k = 1, 2, \ldots, m \\
 & f_k & = & 0 \quad \text{for } k > m.
\end{array}$$

The LP now has m variables. Unfortunately, with this restricted degree, a feasible polynomial may not even exist. If so, a greater degree must be used. Even if an optimal polynomial can be found, there may exist a better polynomial of a higher degree than m. To see if this is the case, the solution vector of the LP must be inspected. For example, if $m = 5$ and for some A the LP gives a solution vector of $\boldsymbol{f} = (f_0, f_1, f_2, f_3, f_4, f_5) = (1, \frac{1}{2}, 0, 0, \frac{1}{3}, \frac{1}{4})$ then it is worth considering increasing m, since it is known that the optimal polynomial is of degree at least five, but may be of higher degree. If m is increased to 12 and the solution vector becomes $(1, \frac{1}{2}, 0, 0, \frac{1}{3}, \frac{1}{4}, \frac{1}{8}, 0, 0, 0, 0, 0, 0)$ then there is good evidence that $m = 6$ is sufficient, since the last few entries are zero.

3.3 Restriction of Constraints

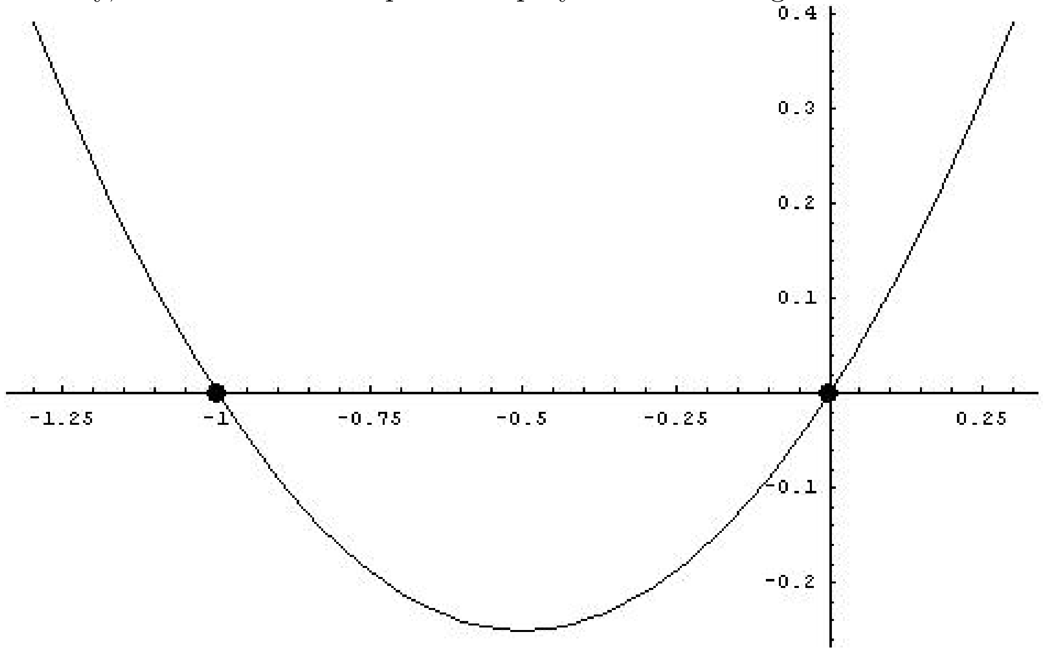
The LP now has m variables, but it still has uncountably many constraints. To solve this, we choose a finite subset P of A. In our LP, we use this finite P in place of the interval A. The LP then becomes:

$$\begin{array}{rrcl}
\text{min} & F(1) & & \\
\text{s.t.} & F(p) & \leq & 0 \quad \forall p \in P \\
f_0 = 1, & f_k & \geq & 0 \quad \text{for } k = 1, 2, \ldots, m \\
 & f_k & = & 0 \quad \text{for } k > m.
\end{array}$$

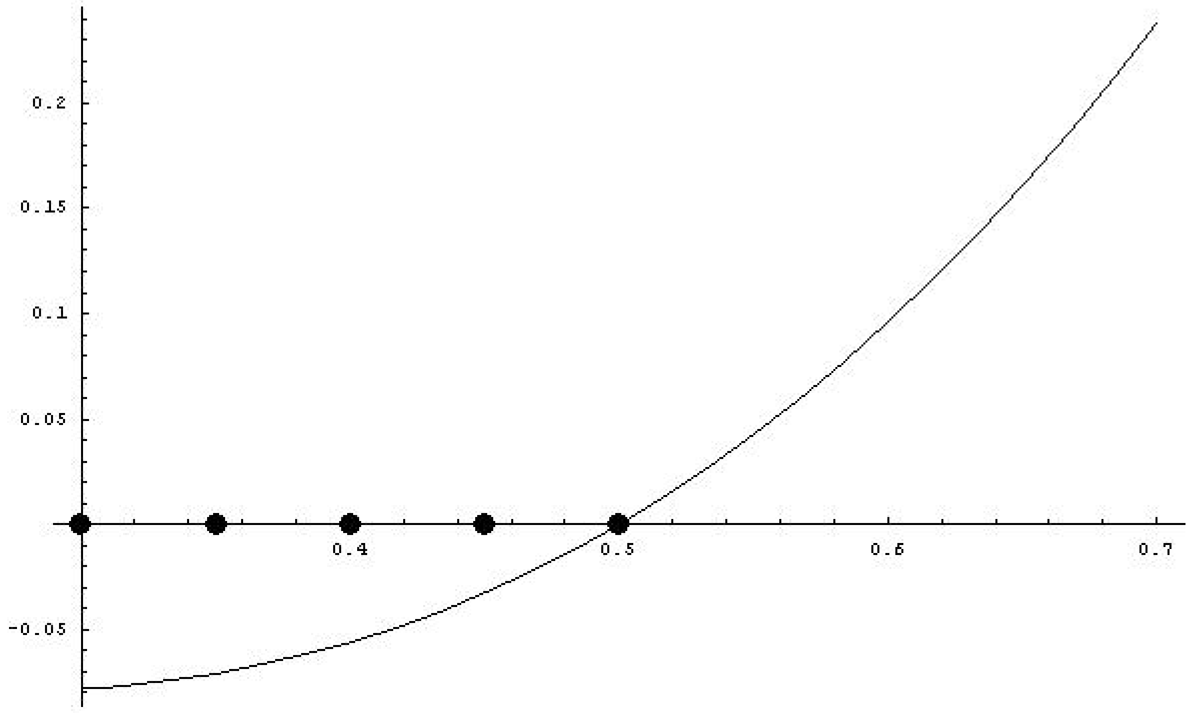
Since P is finite, the LP now has finitely many constraints and variables, and is therefore in a form that can be solved with standard LP software.

We refer to P as the set of "pinpoints". This set has an intuitive meaning. We need $F(\alpha) \leq 0$ for all $\alpha \in A$, but this restriction will require infinitely many constraints. We therefore designate finitely many of the points in A to have constraints. These points are the "pinpoints". These points, which we choose to be spread out somehow evenly along A, should be sufficient to "pin down" $F(x)$ in A. Since $F(x)$ is a polynomial, there should not be any "wild" behavior of the function in-between pinpoints. This restriction of A to P means that the solution of this LP will give un upper bound on the size of a P-code (the inner product of any two points must be in P). We will use this bound of the P-code to approximate the bound for the A-code. This will be shown in Section 3.5.

It is not enough that we require the pinpoints to be spread evenly along A. We also require that the endpoints of every maximal interval in A be included in P. For example, if $A = [-1, -\frac{1}{2}] \cup [-\frac{1}{3}, 0] \cup [\frac{1}{4}, \frac{1}{2}]$ then $\{-1, -\frac{1}{2}, -\frac{1}{3}, 0, \frac{1}{4}, \frac{1}{2}\}$ is required to be a subset of P. To see why, consider a convex quadratic polynomial crossing the x-axis at -1 and 0.



This polynomial is certainly negative between these two points. For a higher degree polynomial, this requirement ensures that the polynomial will not go above (and stay above) the axis near the endpoints of the interval. For example, if $A = [-1, .55]$, and we did not place a pinpoint at $x = .55$, then the polynomial may look something like this:



The polynomial is positive from $x = .5$ to $x = .55$.

3.4 Some Illustrative Examples

Example 3.4.1 *Take $A = [-1, \frac{1}{2}]$ in dimension three. Set the degree of $F(x)$ to be three (that is, set $m = 3$). Suppose we are restricted to four pinpoints. A natural choice of these would be $P = (-1, -\frac{1}{2}, 0, \frac{1}{2})$. The LP for this problem is*

$$\begin{array}{rrl} \min & F(1) & \\ s.t. & F(-1) & \leq 0 \\ & F\left(-\frac{1}{2}\right) & \leq 0 \\ & F(0) & \leq 0 \\ & F\left(\frac{1}{2}\right) & \leq 0 \\ & f_0 & = 1 \quad f_1, f_2, f_3 \geq 0 \end{array}$$

Recall that, in dimension three, the first four Gegenbauer polynomials are $Q_0(x) = 1$, $Q_1(x) = 3x$, $Q_2(x) = -\frac{5}{2} + \frac{15}{2}x^2$, and $Q_3(x) = -\frac{21}{2}x + \frac{35}{2}x^3$. Substituting these into F we get

$$\begin{array}{rccccccccl} \min & f_0 & + & 3f_1 & + & 5f_2 & + & 7f_3 & \\ s.t. & f_0 & - & 3f_1 & + & 5f_2 & - & 7f_3 & \leq 0 \\ & f_0 & - & \frac{3}{2}f_1 & - & \frac{5}{8}f_2 & + & \frac{49}{16}f_3 & \leq 0 \\ & f_0 & - & & & \frac{5}{2}f_2 & & & \leq 0 \\ & f_0 & + & \frac{3}{2}f_1 & - & \frac{5}{8}f_2 & - & \frac{49}{16}f_3 & \leq 0 \\ & f_0 = 1, & & f_1, & & f_2, & & f_3 & \geq 0 \end{array}$$

Solving this LP gives us a solution vector of

$$\boldsymbol{f} = \left(1, \frac{7}{5}, \frac{8}{5}, \frac{24}{35}\right),$$

with the optimal polynomial

$$\begin{aligned} G(x) &= f_0 q_0(x) + \ldots + f_3 q_3(x) \\ &= -3 - 3x + 12x^2 + 12x^3. \end{aligned}$$

The objective value of our LP is $G(1) = 18$*. This is an upper bound on the size of a spherical* P*-code for* A*. We cannot yet say that it is a bound for the* A*-code, since we do not yet know if* $G(x) \leq 0$ *for* $x \in A$ *(we will deal with this problem later).*

We know that the optimal configuration on the interval $A = [-1, \frac{1}{2}]$ *is the icosahedron, which has* 12 *points [8] (see Appendix B), so either the method we have described does not work very well, or we need to do something else to find a better bound. If we increase the degree of the polynomial to four (i.e. set* $m = 4$*) and do the same as above, we get a polynomial of*

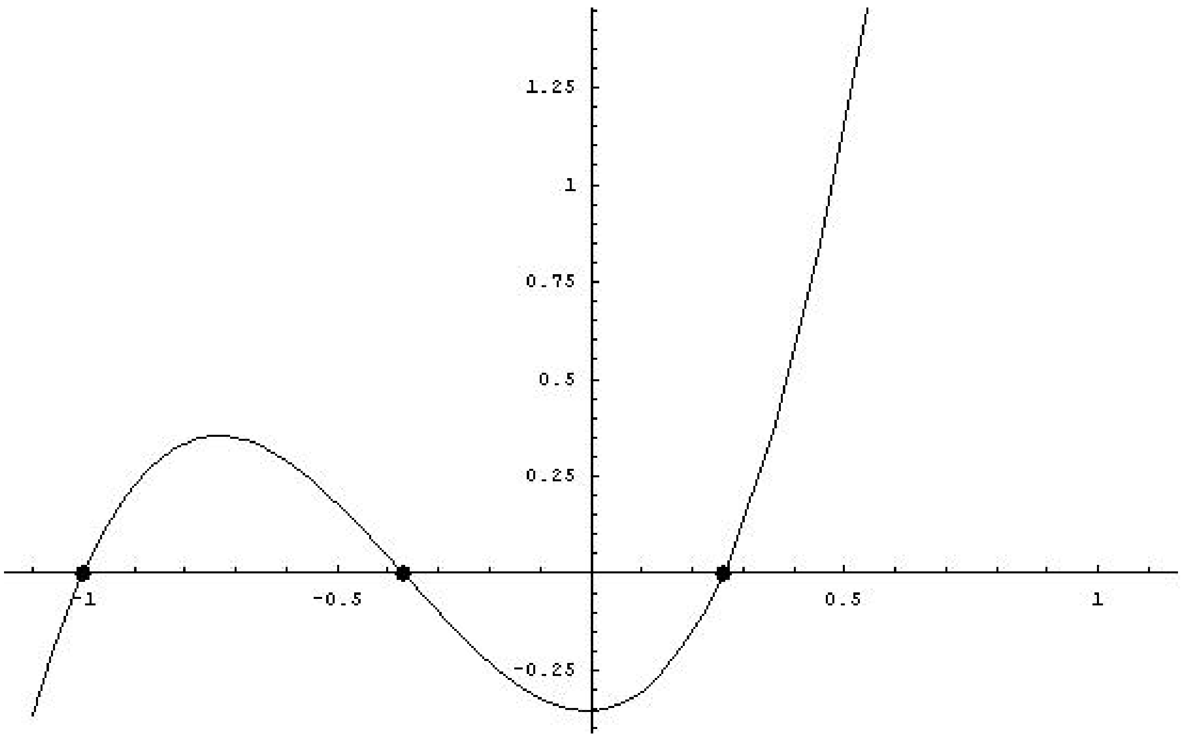
$$G(x) = -\frac{15}{7}x - \frac{15}{7}x^2 + \frac{60}{7}x^3 + \frac{60}{7}x^4$$

with an objective value of $G(1) = \frac{90}{7} \approx 12.8571$*. Increasing the degree incrementally, it can be seen that there is no change to the objective value until we reach* $m = 10$*, when we get a value of* $\frac{1207470}{99071} \approx 12.1879$*. So increasing the number of variables in the LP by just one can sometimes make a large difference on the bound we get for the* P*-code.*

Example 3.4.2 *Consider the interval* $A = \left[-1, \frac{\sqrt{2}}{4+\sqrt{2}}\right] \approx [-1, .261204]$*. The optimal configuration on this interval is the square antiprism, which has eight points (see Appendix B and Section 5.2.1). We will use a polynomial of degree four on three constraints. The pinpoints will be placed evenly along* A *at* $P = \{-1, -0.368398, 0.261204\}$*. Solving this LP gives a solution vector of* $\boldsymbol{f} = (1, 0.74714, 0.541421, 0.209384, 0)$*. The optimal polynomial is*

$$G(x) = -0.353553 + 0.0428932x + 4.06066x^2 + 3.66421x^3$$

and the objective value is $G(1) = 7.41421$*. We now run into a different problem than the one we encountered in Example 3.4.1. We know that the optimal configuration for* $A = \left[-1, \frac{\sqrt{2}}{4+\sqrt{2}}\right]$ *has eight points. Is the LP telling us that this configuration does not exist? Certainly something is wrong here. A plot of* $G(x)$ *with the pinpoints highlighted,*



shows us that $G(x) > 0$ at some points in A. The constraints of the finite LP have been satisfied, but not those of the original LP. To remedy this we need to develop another tool.

3.5 Finding A Compatible Polynomial

Recall Definition 2.3.1: $F(x)$ is compatible with A if $F(\alpha) \leq 0 \ \forall \alpha \in A$. As we have seen in the previous example, when we restrict the LP to P instead of A we cannot always ensure that $G(x)$ will be compatible with A. We need a way to make $G(x)$ compatible after we find it through the LP.

We will define ε to be the maximum value that $G(x)$ achieves in A. If $\varepsilon \leq 0$ then all of the constraints of the original LP are satisfied, so we can confidently take $G(x)$ to be an optimal polynomial for the given degree. If $\varepsilon > 0$ then $G(x)$ is not compatible with A. Now we consider the polynomial

$$G(x) - \varepsilon = (1 - \varepsilon) + f_1 Q_1(x) + \ldots + f_m Q_m(x).$$

This polynomial is certainly compatible with A. The problem now is that we do not have the zeroth term $f_0 = 1$, which was required in the original LP. We now scale the entire polynomial to achieve this constraint by taking

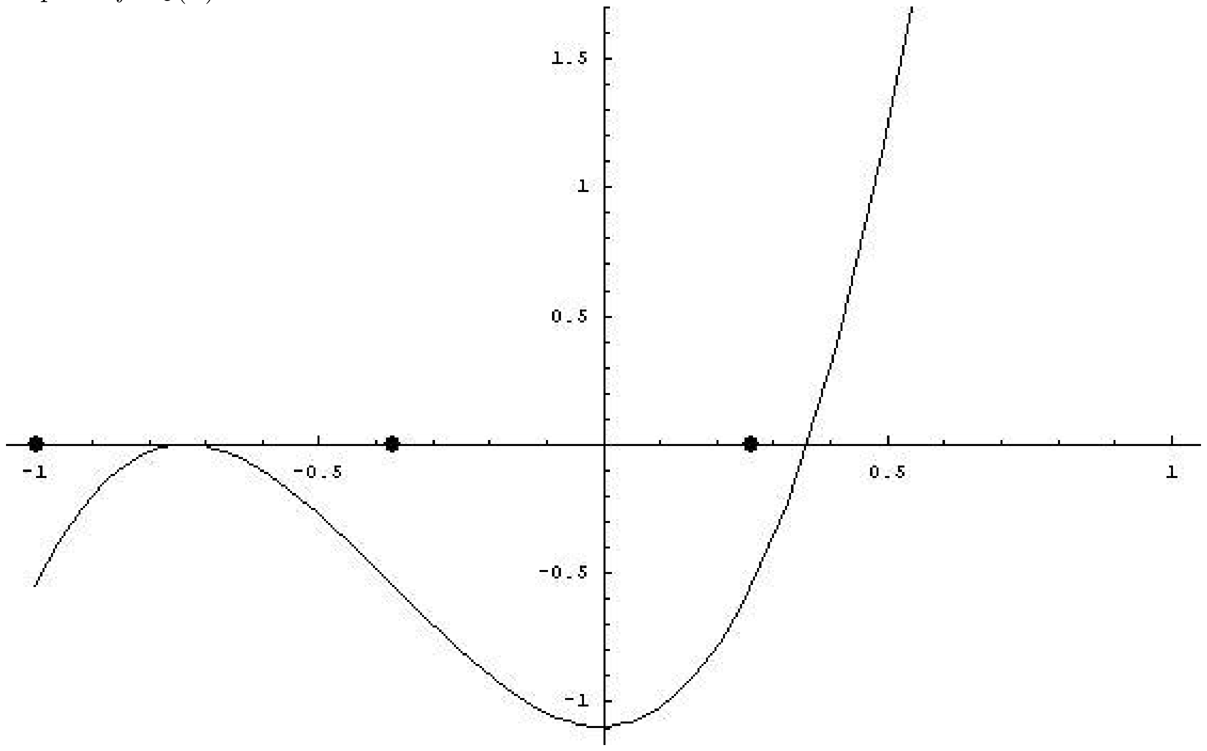
$$G_\varepsilon(x) = \frac{G(x) - \varepsilon}{1 - \varepsilon}.$$

This transformation has the effect of shifting $G(x)$ to below zero, and then rescaling to get $f_0 = 1$. It must be noted that in practice ε is usually very small, so $1 - \varepsilon$ will be very close to one. The shifting and scaling of the polynomial will not be dramatic, but it is still necessary to achieve a true upper bound.

Example 3.5.1 *Returning to Example 3.4.2, we find that the local maximum of* $G(x)$ *occurs at the point* $x = -0.733476$*. We take* $\varepsilon = G(-0.733476) = 0.353667$ *and we find that*

$$G_\varepsilon(x) = \frac{G(x) - 0.353667}{1 - 0.353667} = -1.0942 + 0.066364x + 6.28261x^2 + 5.66923x^3$$

A plot of $G_\varepsilon(x)$ *shows that the curve does indeed lie below the x-axis in A.*



Our revised bound is $G_\varepsilon(1) = 10.924$*. This bound is now consistent with the antiprism. It must be noted that in general practice, when many more pinpoints are used,* ε *is very small, and so the scaling factor of* $\frac{1}{1-\varepsilon}$ *is close to one. Indeed, if we were to use the same interval A but on 50 pinpoints, we find that* $\varepsilon = 0.000632252$ *and that* $G_\varepsilon(1) = 8.48819$*, proving optimality of the antiprism.*

Chapter 4

Software Implementation

With an LP now in a form that can be solved using software, we wish to do just that. We want to write a program that, when given a few user specified-inputs, will give information on an A-code, such as an optimal polynomial, a solution vector and an upper bound.

In this chapter we discuss the concept behind the package; what we want it to accomplish and why we chose to write it in Mathematica. We then give an idea of how the package handles a typical problem; specifically, how pinpoints are chosen and how ε is found. Throughout the chapter we provide a running example to help describe the functionality of the package.

4.1 Theory Behind the Program

Mathematica is a computational software program. It was initially conceived by Stephen Wolfram and was developed and released by Wolfram Research in 1988. We created our package using the WPI liscense of Version 5.2 of this software. We wrote a Mathematica package that will take the following user input: dimension d, polynomial degree m, number of constraints $|P|$ and interval A. There are functions included in the package that will give a solution vector $\boldsymbol{f}$, an optimal polynomial $G_\varepsilon(x)$, an upper bound $G_\varepsilon(1)$, as well as many other desired outputs. A user is able to download the package and an accompanying tutorial worksheet from the Mathematica website at http://library.wolfram.com/.

We chose to write our package using Mathematica because of its ability to do a large amount of exact rational arithmetic. If A is input with exact rational endpoints, then $\boldsymbol{f}$ will be rational. Unfortunately, since it may be impossible to find ε exactly, if $G(x)$ is of a high enough degree, it is not always possible to find an exact rational expression for $G_\varepsilon(x)$.

4.2 Description of the Code

In our package, we define the Gegenbauer polynomials recursively, as described in Theorem 2.1.1:

$$\ell_{k+1}Q_{k+1}(x) = xQ_k(x) - (1-\ell_{k-1})Q_{k-1}(x),$$

$$\ell_k = \frac{k}{d+2k-2}, \qquad Q_0(x) = 1, \qquad Q_1(x) = dx.$$

From this we construct a list $\{Q_0(x), \dots, Q_m(x)\}$ which, in our package, will be a function of x that serves the purpose of $F(x)$. Note that in Mathematica all lists are enclosed in curly brackets, and this notation will be used throughout this chapter.

4.2.1 Specifying Pinpoints

The LinearProgramming function of Mathematica takes in lists of coefficient vectors as the objective function and as the constraint matrix. This is why $\{Q_0(x), \dots, Q_m(x)\}$ serves as $F(x)$. To generate the constraint matrix, we first need a way to place the pinpoints. We use the following algorithm to do this:

```
Input: a collection of intervals $A$ and a desired size of $P$.
Set $P_0 = \varnothing$.
if $A$ is finite then
  return $A$.
else
  Let $\{s_1, \dots, s_i\}$ denote the subintervals of $A$, and let $\{|s_1|, \dots, |s_i|\}$ denote their lengths.
  Add the endpoints of $s_j$ to $P_0$, for $j = 1, \dots, i$.
  Calculate $\left\{\frac{|s_1|}{|A|}, \dots, \frac{|s_i|}{|A|}\right\}$.
  Multiply these by $|P|$ and round up: $\left\{\left\lceil\frac{|P||s_1|}{|A|}\right\rceil, \dots, \left\lceil\frac{|P||s_i|}{|A|}\right\rceil\right\}$
  Distribute that number of points evenly along each corresponding subinterval.
  Add the placed points to $P_0$.
  return $P_0$.
end if
```

The P_0 obtained from the previous algorithm will be the set of pinpoints P used in the LP. If A is finite, then we do not need to find P; A can be used for the constraints. This algorithm begins by making sure P includes the endpoints of every subinterval in A. The subintervals are determined by the way in which A is defined. For instance, if A is input as $[-1, 0]$ then the algorithm will begin by adding $\{-1, 0\}$ to P_0. On the other hand, if A is input as $[-1, -.4] \cup [-.6, 0]$ then the algorithm will begin by adding $\{-1, .6, .4, 0\}$ to P_0. An interval can be input in this manner if some specific points are desired to be used as constraints. This will also cause a greater concentration of pinpoints to be placed within [-.6,-.4], as this region appears in multiple subintervals of A. After these endpoints are added to P_0, the algorithm then distributes the remaining points evenly along A, and then adds these to P_0.

A set A is entered as a list of intervals and single points. As the code is explained, an example interval of

$$A = [-\frac{9}{10}, -\frac{3}{4}] \cup [-\frac{1}{2}] \cup [-\frac{2}{5}, 0] \cup [\frac{1}{10}, \frac{2}{5}]$$

will be used, with $m = 8$. This interval is specified in Mathematica as $\{\{-\frac{9}{10}, -\frac{3}{4}\}, \{-\frac{1}{2}\}, \{-\frac{2}{5}, 0\}, \{\frac{1}{10}, \frac{2}{5}\}\}$. Running the previous algorithm on this interval, with a desired P of size 15, we find that the lengths the subintervals are $\{\frac{3}{20}, 0, \frac{2}{5}, \frac{3}{10}\}$ and that they get $\{3, 0, 8, 6\}$ pinpoints each. Distributing these out we get

$$P_0 = \left\{-\frac{9}{10}, -\frac{33}{40}, -\frac{3}{4}, -\frac{1}{2}, -\frac{2}{5}, -\frac{12}{35}, -\frac{2}{7}, -\frac{8}{35}, -\frac{6}{35}, -\frac{4}{35}, -\frac{2}{35}, 0, \frac{1}{10}, \frac{4}{25}, \frac{11}{50}, \frac{7}{25}, \frac{17}{50}, \frac{2}{5}\right\}$$

Notice that $|P_0| = 18$. This is because of the rounding up that is done in choosing how many pinpoints to place. It is also because of the subinterval of length zero at $-\frac{1}{2}$. Since this subinterval does not contribute to the length of A, it is not taken into consideration when choosing how many points to place in each subinterval and an extra point is added on top of the desired 15.

4.2.2 Finding $G_\varepsilon(x)$

The next step in our package is to set up and solve the LP, which is done easily now that we have P_0. After setting up the constraints, we use the Mathematica LinearProgramming function to get a solution vector $\boldsymbol{f}$. In our example interval, the solution vector is

$$\boldsymbol{f} = \{1, \frac{22021553820}{26047002169}, \frac{6282568590714}{10028095835065}, \frac{8495863800}{26047002169}, \frac{291299502560}{2370277197379}, 0, 0, 0, \frac{983449600000}{63320262272839}\}.$$

The LinearProgramming function only outputs the solution vector. Since $F(x)$ is represented now by the coefficient vector of $F(x) = \{Q_0(x), \ldots, Q_m(x)\}$, to find the optimal polynomial for the LP we take

$$\boldsymbol{f} \cdot F(x) = f_0 Q_0(x) + \ldots + f_m Q_m(x) = G(x).$$

The LP has an objective value of

$$\boldsymbol{f} \cdot F(1) = f_0 Q_0(1) + \ldots + f_m Q_m(1) = G(1).$$

In dimension three, for a degree eight polynomial, $F(1) = \{1, 3, 5, 7, 9, 11, 13, 15, 17\}$. For our example we get the polynomial

$$G(x) = \frac{345744000000}{26047002169}x^8 - \frac{645388800000}{26047002169}x^6 + \frac{498382746300}{26047002169}x^4 + \frac{148677616500}{26047002169}x^3 - \frac{2807754183}{1370894851}x^2 - \frac{23141908440}{26047002169}x - \frac{108668628}{1370894851}$$

with

$$G(1) = \frac{14150611629}{1370894851} \approx 10.322$$

The next step in our package is to find $G_\varepsilon(x)$. This is done using the following algorithm:

```
**Input:** $A$ and $G(x)$
Calculate $G'(x)$, the derivative of $G(x)$.
Find all real $x$ such that $G'(x) = 0$ and $x \in A$. Call these $x_1, x_2, \ldots, x_i$.
Take $\varepsilon = \max\{G(x_1), G(x_2), \ldots, G(x_i)\}$.
if $\varepsilon > 0$ then
  return $G_\varepsilon(x) = \frac{G(x)-\varepsilon}{1-\varepsilon}$.
else
  return $G_\varepsilon(x) = G(x)$
end if
```

Since $G(x)$ is a polynomial, finding the derivative is simple. Recall Section 3.3: since all endpoints of A are pinpoints, ε will always be a local maximum in the interior of A (i.e. $G(x)$ will never be positive on an endpoint of A).

This is the step at which we may lose rationality. If $G(x)$ is of degree greater than or equal to six, then $G'(x)$ is of degree greater than or equal to five. Because of this, it will not always be possible to find the roots of $G'(x)$ exactly. Mathematica must employ numerical methods.

Returning to our example interval A, using our package we find that the local extrema of $G(x)$ occur at $x_1 = -0.845167$, $x_2 = -0.379143$, $x_3 = -0.184345$, and $x_4 = 0.235995$. We find that

$$\begin{aligned}\varepsilon &= \max\{G(x_1), G(x_2), G(x_3), G(x_4)\}\\ &= \max\{-0.0494678, -0.0204761, 0.000298193, -0.272789\}\\ &= 0.000298193.\end{aligned}$$

Using this ε we are able to find

$$G_\varepsilon(x) = -0.0795903 - 0.888732x - 2.04873x^2 + 5.70975x^3 + 19.1397x^4 - 24.7852x^6 + 13.2778x^8$$

and

$$G_\varepsilon(1) = 10.325.$$

While this is only a small change from $G(1) = 10.322$, in some cases the transition from $G(x)$ to $G_\varepsilon(x)$ is very important, as we saw in Example 3.5.1.

Chapter 5

Low-Dimensional Codes

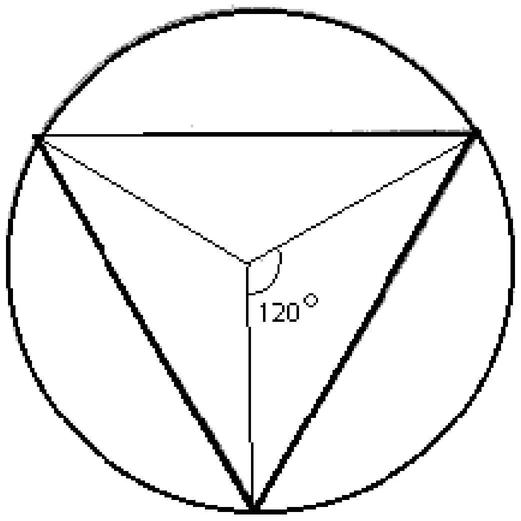
With a working Mathematica package, we wish to see how the outputs from the package relate to known examples in dimensions two and three. Instead of looking at upper bounds on single intervals, we wish to see how an upper bound will change in relation to a change in the size of an interval.

In this chapter we study known configurations in dimensions two and three. Looking at the output of the program developed in Mathematica reveals some intriguing results, and we investigate these. We conclude the chapter by proving optimality of a configuration on eight points in dimension three.

5.1 A Study of Dimension Two

A configuration of points is simply a placement of those points on the unit sphere. An optimal configuration is one such that a different placement of the same number of points cannot result in larger minimum angle between any two points. Equivalently, this means that a different configuration cannot result in a smaller interval A of inner products.

In two dimensions, the optimal configurations are the regular polygons. For instance, if we take $A = \left[-1, -\frac{1}{2}\right]$, all points must be at least $\arccos\left(-\frac{1}{2}\right) = 120^\circ = \frac{2\pi}{3}$ radians apart. This restriction is clearly optimized on the vertices of an inscribed equilateral triangle, as shown.



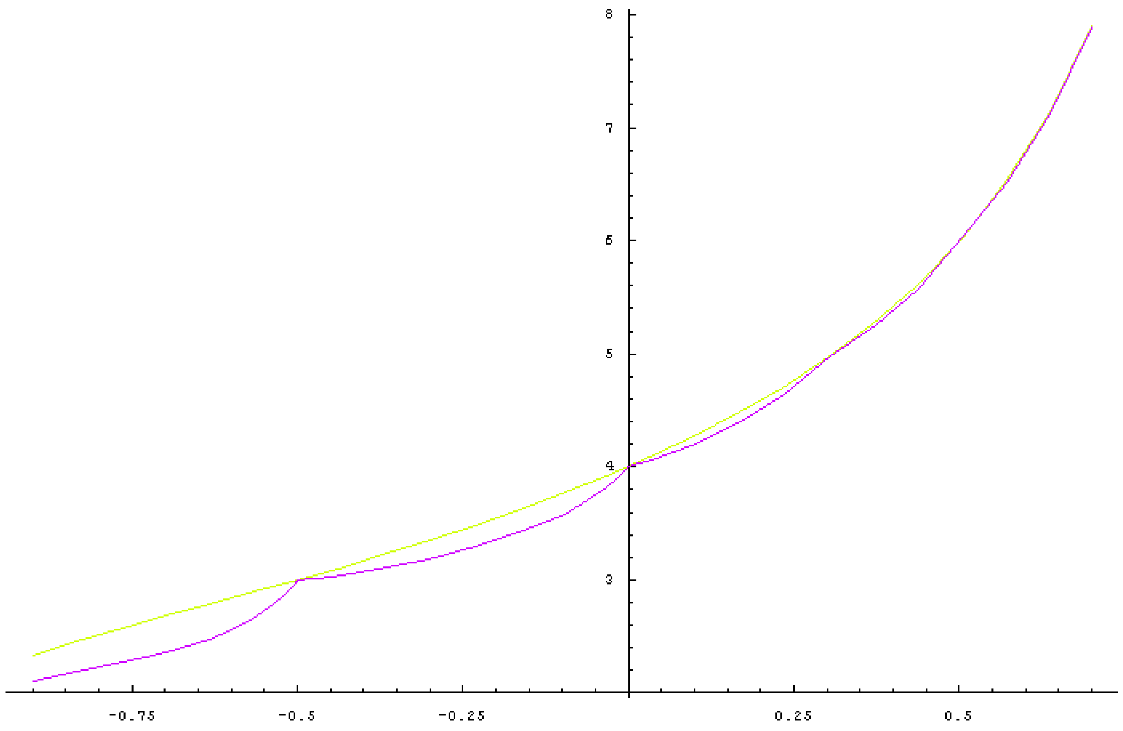
A different configuration of three points cannot have an interval with a lower right endpoint than $-\frac{1}{2}$. Increasing the angle between any two points will decrease the angles between some other pair, and a larger interval will result.

As the length of the interval is increased, another point cannot be added until $A = [-1, 0]$ is reached. Now, every point must be at least $90° = \frac{\pi}{2}$ radians apart. This restriction is optimized on the vertices of an inscribed square. This pattern will continue, with every regular polygon being an optimal configuration. From this we can observe that a valid upper bound for $n = |X|$, where X is a spherical A-code for the interval $A = [-1, x]$, in two dimensions is $k(x) = \frac{2\pi}{\arccos(x)}$.

The question then becomes how well the linear programming bound compares to this. To answer this we define

$$h(x) = G_\varepsilon(1) \text{ for } A = [-1, x].$$

This function will show us how the upper bound changes as A changes. Plotting $h(x)$ (using a polynomial of degree ten) in pink against $k(x) = \frac{2\pi}{\arccos(x)}$ in yellow, we get the following graph:



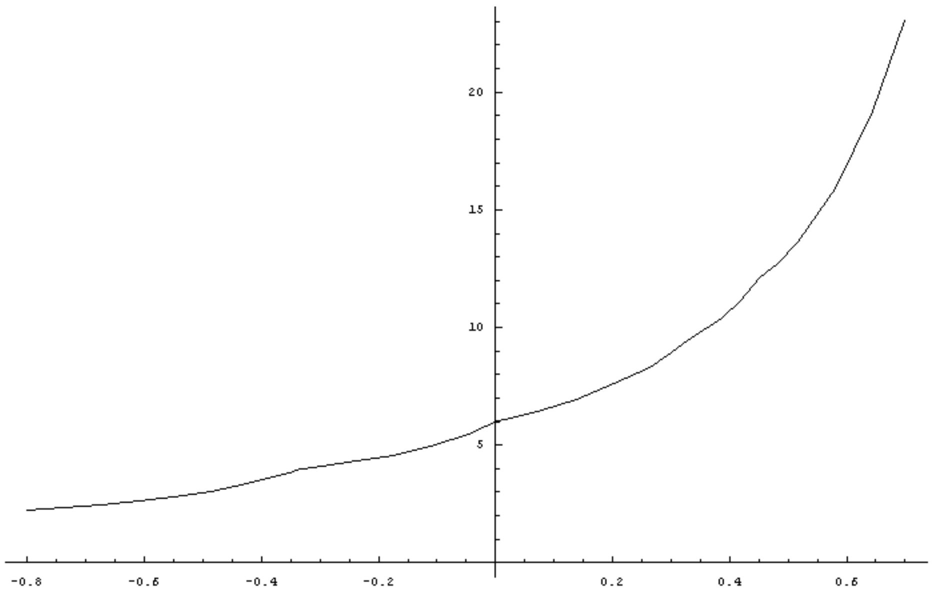
From this we observe that the LP bound is indeed better than the $k(x)$ bound. We also see that the curve of $h(x)$ is not smooth. There are points at which the second derivative of $h(x)$ increases quickly, and it "bumps" up to meet $k(x)$. Upon observation we notice that these bumps correspond exactly to the optimal configurations of the regular polygons. There is a bump at $x = -\frac{1}{2}$, $x = 0$ and so on. We can also see that $h\left(-\frac{1}{2}\right) = 3$ (triangle) and $h(0) = 4$ (square), etc.

At a certain point, $h(x)$ will cross $k(x)$ and will stay above it. The point where this cross occurs is related to the degree of the polynomial used to solve the LP. A higher degree polynomial will have better results for larger values of x. We believe that the location of this crossing will increase as the degree increases. These bumps also eventually stop being visible for larger values of x. Our experiment show that a higher degree polynomial will yield better results, with more bumps being visible.

5.2 A Study of the Dimension Three

Moving from two dimensions to three dimensions, we expect things to get more complicated. Indeed, we do not have a simple elementary function with which to bound $h(x)$. There are also cases where a configuration on k points will have the same inner product interval as a configuration on $k+1$ points. For example, the interval $A = [-1, 0]$ allows the optimal configuration of the vertices of the octahedron. If we take any configuration on five points, we can not do any better than this same A. The optimal configuration on five points is simply the same as on 6 points with an arbitrary point removed. A library of configurations (not all optimal) in three dimensions is given in Appendix B.

If we graph $h(x)$ in three dimensions we get the following:



From this graph we can see that there is not a bump at every single point. We can only see bumps where $h(x)$ equals 4 (the tetrahedron), 6 (the octahedron) and 12 (the icosahedron). These configurations are three of the platonic solids. The dodecahedron has 20 points, but the curve is too steep at that point to observe a visible bump, even upon zooming in on that point. We believe that there should be a bump there, though, since the dodecahedron is known to be optimal on 20 points [8]. The remaining platonic solid, the cube, which has eight points, also does not have a bump. The reason for this is shown in the following subsection.

5.2.1 The Square Antiprism

While most of the platonic solids form optimal configurations in three dimensions, oddly enough the cube does not. The cube, with vertices at

$$\left(\pm\frac{\sqrt{3}}{3}, \pm\frac{\sqrt{3}}{3}, \pm\frac{\sqrt{3}}{3}\right)$$

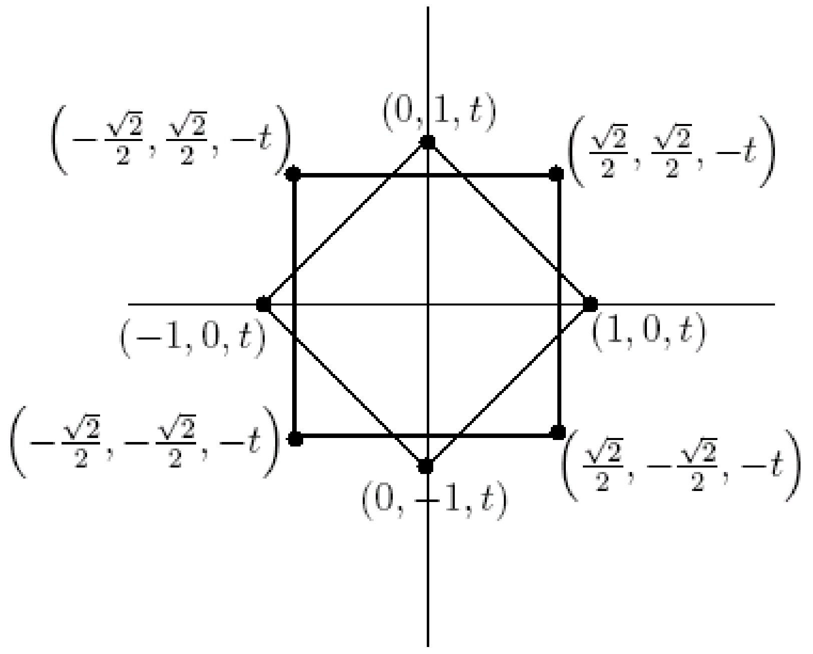
has minimum angle

$$\arccos\left[\left(\frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}\right) \cdot \left(\frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}, -\frac{\sqrt{3}}{3}\right)\right] = \arccos\left(\frac{1}{3}\right)$$

which corresponds to the interval $A = \left[-1, \frac{1}{3}\right]$. Interestingly, we can find another configuration on eight points with a smaller interval. This configuration is the square antiprism.

Imagine looking down on a cube and rotating the bottom face by $\frac{\pi}{4}$. The points on the top face now have a greater angle between adjacent points on the bottom face. Now we begin to move these two faces together (remember that these points are still inscribed in a unit sphere). As these faces grow closer together, the angles between adjacent point on each face grow larger, while the angles between points adjacent on opposite faces grow smaller. As some point these values will be the same. If we determine when this happens, we will have found a configuration of eight points that has a larger minimum angle than the points of the cube, and thus a smaller corresponding interval A.

We will call this value t. To find it, consider the following drawing of the vertices:


$(0,1,t)$
$\left(-\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2},-t\right)$
$\left(\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2},-t\right)$
$(-1,0,t)$
$(1,0,t)$
$\left(-\frac{\sqrt{2}}{2},-\frac{\sqrt{2}}{2},-t\right)$
$\left(\frac{\sqrt{2}}{2},-\frac{\sqrt{2}}{2},-t\right)$
$(0,-1,t)$

This figure is an image of this rotated cube, looking down through the z-axis from a vantage point of $(0,0,10)$. The top face of this figure is on the plane $z = t$. The bottom face is on $z = -t$. The bottom face is rotated by $\frac{\pi}{4}$. For ease of calculations we will find t on these points and scale to a unit sphere at the end. We want the distance between adjacent points on the top face to be equal to the distance between points adjacent on opposite faces. We set up this equation and solve for t:

$$\begin{aligned}
d\left[(1,0,t),(0,1,t)\right] &= d\left[(1,0,t),\left(\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2},-t\right)\right] \\
\sqrt{(1-0)^2+(0-1)^2+(t-t)^2} &= \sqrt{\left(1-\frac{\sqrt{2}}{2}\right)^2+\left(0-\frac{\sqrt{2}}{2}\right)^2+(t-(-t))^2} \\
\sqrt{2} &= \sqrt{2-\sqrt{2}+4t^2} \\
\sqrt{2} &= 4t^2 \\
\frac{\sqrt[4]{2}}{2} &= t
\end{aligned}$$

Now that we have t, we need to scale all of the points so that they lie on the unit sphere. Since all of the points are the same distance from the origin, the scaling factor will be the norm of one of the points:

$$\left|\left(1,0,\frac{\sqrt[4]{2}}{2}\right)\right| = \sqrt{1+\left(\frac{\sqrt[4]{2}}{2}\right)^2} = \sqrt{1+\frac{\sqrt{2}}{4}} = k$$

Finally, to find the maximum dot product between two points on this configuration:

$$\left(\frac{1}{k},0,\frac{t}{k}\right)\cdot\left(0,\frac{1}{k},\frac{t}{k}\right) = \frac{t^2}{k^2} = \frac{\frac{\sqrt{2}}{4}}{\frac{4+\sqrt{2}}{4}} = \frac{\sqrt{2}}{4+\sqrt{2}} \approx .261204.$$

We have therefore found a configuration of eight points on the interval $A = [-1, .261204]$, which is better than the configuration of the cube, on $A = \left[-1, \frac{1}{3}\right]$. Refer to Examples 3.4.2 and 3.5.1 for more information on this configuration.

The question now becomes: why is there not a bump in $h(x)$ at $x = .261204$? We have an optimal configuration there on eight points. For that matter, why is there not a bump at every integer value of $h(x)$? There are bumps in dimension two at every integer value, corresponding to the regular polygons. There exist optimal configurations on $7,\ 8,\ 9,\ldots$ points in three dimensions, but these configurations do not produce bumps. Somehow, information about certain configurations lie within the Gegenbauer polynomials. The question of why the bumps appear, and exactly where they appear in higher dimensions, remains an open problem.

Chapter 6

Conclusions and Future Work

Our main goal in this project was to write a Mathematica package to find the upper bounds on the size of spherical codes. In order to complete this task it was necessary to first study and understand the 1977 paper by Delsarte, Goethals, and Seidel [4] which describes their linear programming technique. This technique allowed the generation of a linear program with countably many variables and uncountably many constraints. We then were able to manipulate and reformulate this technique to compute our linear programming bounds. We also looked into several applications of this technique, including: the kissing number, equiangular lines and the 24-cell. Ultimately, our Mathematica package was developed to allow a user to enter any specified collection of intervals, a dimension, the degree of the polynomial, and the number of constraints they desire. Given this input, the program outputs for the user an optimal solution vector, an optimal polynomial, and the upper bound, as well as a few other things.

In carrying out this project we came across several relevant areas of study that branch off of the main subject, and could possibly be studied in the future. Several open problems we came across, which are directly relevant to this project, are discussed in the sections that follow.

6.1 A Study of the 'Bumps'

The discovery of the bumps, which are discussed in Chapter 5, opened many questions unanswered in this project. The general question that is left unanswered is, why do these bumps exist and what do they mean? The more intriguing result of this discovery is the fact that these bumps correspond to the optimal configurations of polygons in two-dimensions and polyhedra in three-dimensions. The graph of the objective function in dimension two seems to rise up and intersect the graph of the arccosine function, which is also the upper bound on the objective function, at several points. These points correspond to the optimal configurations of the polygons. In three dimensions the graph of the objective function, once again, rises at several points forming bump-like shapes. The points at which this occurs are the optimal configurations of certain Platonic solids. It is important to point out that $h(x)$

is determined entirely by Gegenbauers and therefore the Gegenbauers somehow "encode" the polygons. By looking further into the occurrence of these bumps it may be possible to find an answer to these questions.

6.2 The Dual Linear Program

The Delsarte, Goethals, and Seidel "LP" is not really an LP and does not necessarily admit a strong duality theorem (although there is a weak form of duality). By making the number of variables and number of constraints finite, we have approximated their optimization problem by one which enjoys strong duality. If a follow-up project tries to understand these duals and takes their limit as $m \to \infty$ and $|P| \to \infty$, then one may gain insight not only into spherical codes but also into Lagrangian Duality.

6.3 Improving the Mathematica Package

In the future, it is possible for a more robust Mathematica package to be developed. The program that is developed in this project is the first of its kind that we know of available to the public. Clearly, it is possible to make it more computationally efficient and have more options for the user in the future. A study of the numerical issues can also be carried out in the future. The dual linear program, discussed in the previous Section 6.2, could also be implemented into the software. Taking care of these details within the Mathematica package that we developed in this project would allow for it to be more powerful, and these developments are left open for the future.

Bibliography

[1] J. H. Conway and N. J. A. Sloane. Sphere Packings, Lattices and Groups (3rd ed.). Springer-Verlag, New York (1999).

[2] D. de Caen. Large equiangular sets of lines in Euclidean space. The Electronic Journal of Combinatorics. 7 (2000).

[3] P. Delsarte. An algebraic approach to the association schemes of coding theory. *Philips Res. Reports Suppl.* **10** (1973).

[4] P. Delsarte, J.-M. Goethals, J. J. Seidel. Spherical codes and designs. *Geom. Ded.* **6** (1977), 363-388.

[5] C. D. Godsil. Algebraic Combinatorics. Chapman and Hall, New York (1993).

[6] C. D. Godsil and G. Royle. Algebraic Graph Theory. Springer-Verlag, New York (2001).

[7] F. Pfender and G. M. Ziegler. Kissing Numbers, Sphere Packings, and Some Unexpected Proofs. Notice of the AMS. 51 (2004).

[8] N. J. A. Sloane. "Spherical Codes". http://www.research.att.com/ ~njas.

Appendix A

User Manual

Load the package using the command

<< NameOfPackage‘

using the file name that you saved the package as. For the problem that you wish to solve, you must first enter in the dimension, the degree of the polynomial, and the number of constraints as follows:

d = ;
polydegree = ;
pinpoints = ;

A collection of intervals is specified using curly brackets. For example, the set $A = [-1,-.5] \cup [0,.5]$ must be defined by $\{\{-1,-.5\},\{0,.5\}\}$. Even if a single interval is desired, a double set of curly brackets is still required. For example, the set $A = [-1,0]$ must be defined by $\{\{-1,0\}\}$. The collection of intervals specified need not be disjoint. The previous interval could also be defined as $\{\{-1,.4\},\{.8,0\},\{.5\}\}$. Notice that single points can also be specified. This will ensure that a pinpoint is placed at that point.

The functions in the package take only the interval A as input. Dimension, polynomial degree and number of pinpoints must be specified beforehand, and these values will hold for all functions until they are changed.

Some important functions in the package are:

Allowable[A] gives the set of pinpoints that will be used in the LP. If a specific point is desired, that single point should be added to A.

Fvector[A] gives the solution vector of the LP.

Fpoly[A] gives the optimal polynomial to the LP, $G(x)$.

Fobjective[A] gives $G(1)$.

FepsMax[A] gives the ε value of $G(x)$.

Fepsilon[A] gives $G_\varepsilon(x)$, the polynomial compatible with A.

EpsilonPlot[A] shows a graph of $G_\varepsilon(x)$

UpBound[A] gives $G_\varepsilon(1)$, the upper bound on the size of the A-code.

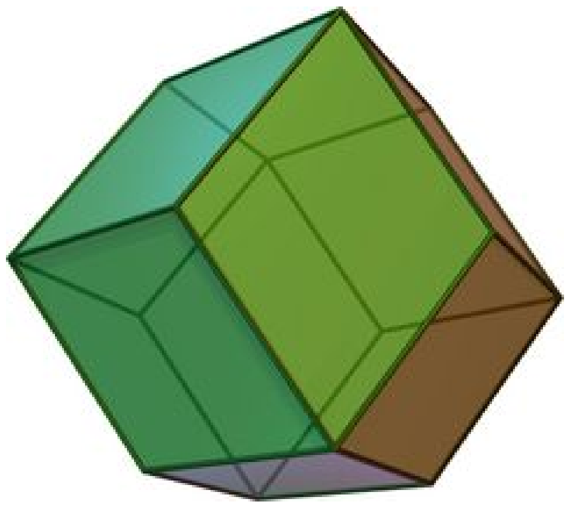
A tutorial worksheet is also included with the package. When loaded it will prompt you to specify the dimension, polynomial degree and number of pinpoints. It will then take you through all of the functions included in the package.

Appendix B

Library of Examples

Rhombic Dodecahedron (the Voronoi cell for fcc lattice), center at origin.

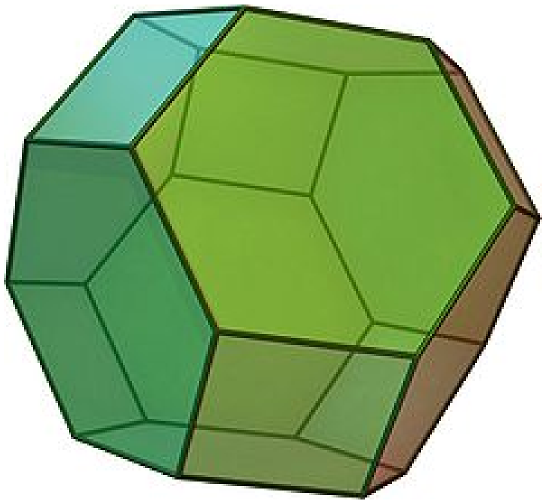
6 vertices of the form[1] $(\pm 1, 0, 0)$ and 8 vertices of the form $(\pm 1/2, \pm 1/2, \pm 1/2)$



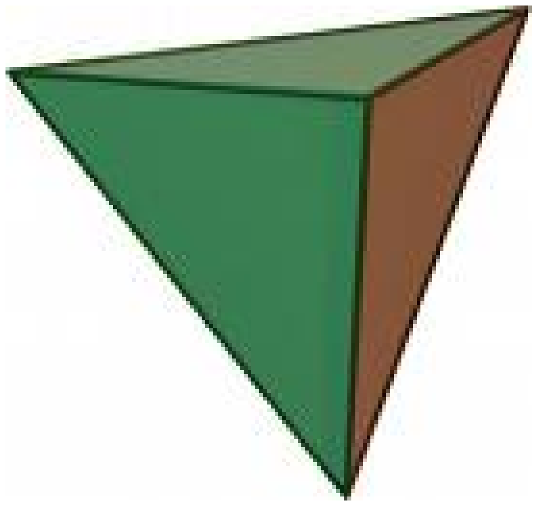
[1]Following N. Sloane's convention, we use the phrase "of the form" to indicate that one should include the entire orbit of this vector under the action of the symmetric permuting coordinates

Truncated Octahedron, (the Voronoi cell for bcc lattice).

24 vertices of the form $(\pm 1, \pm 1/2, 0)$



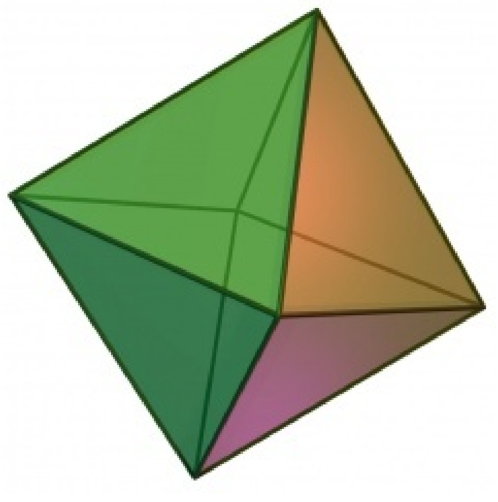
Tetrahedron 3 Dimensions, 4 points (vertices)



$$\begin{array}{rrr}(\ \sigma, & \sigma, & \sigma), \\ (-\sigma, & -\sigma, & \sigma), \\ (-\sigma, & \sigma, & -\sigma), \\ (\ \sigma, & -\sigma, & -\sigma)\end{array}$$

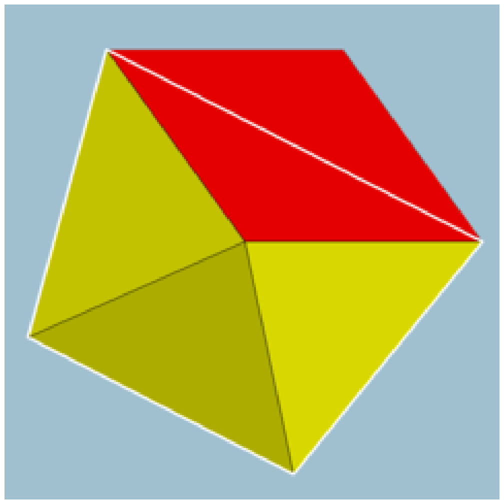
where $\sigma = \frac{1}{\sqrt{3}}$.

Octahedron 3 Dimensions, 6 points (vertices)



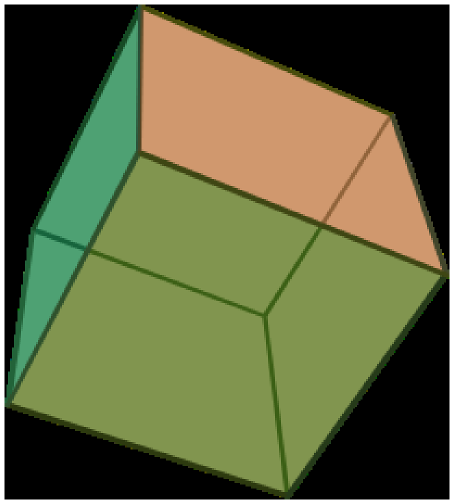
$$\begin{array}{rrr}
(\ 1, & 0, & 0), \\
(-1, & 0, & 0), \\
(\ 0, & 1, & 0), \\
(\ 0, & -1, & 0), \\
(\ 0, & 0, & 1), \\
(\ 0, & 0, & -1)
\end{array}$$

Square Antiprism 3 Dimensions, 8 points
Twisted cube or square antiprism
The edge length is $= \sqrt{2}$.



$$
\begin{array}{rrr}
(\ 1, & 0, & \frac{1}{2^{\frac{3}{4}}}), \\
(-1, & 0, & \frac{1}{2^{\frac{3}{4}}}), \\
(\ 0, & 1, & \frac{1}{2^{\frac{3}{4}}}), \\
(\ 0, & -1, & \frac{1}{2^{\frac{3}{4}}}), \\
(\ \frac{1}{2^{\frac{1}{2}}}, & \frac{1}{2^{\frac{1}{2}}}, & -\frac{1}{2^{\frac{3}{4}}}), \\
(\ \frac{1}{2^{\frac{1}{2}}}, & -\frac{1}{2^{\frac{1}{2}}}, & -\frac{1}{2^{\frac{3}{4}}}), \\
(-\frac{1}{2^{\frac{1}{2}}}, & \frac{1}{2^{\frac{1}{2}}}, & -\frac{1}{2^{\frac{3}{4}}}), \\
(-\frac{1}{2^{\frac{1}{2}}}, & -\frac{1}{2^{\frac{1}{2}}}, & -\frac{1}{2^{\frac{3}{4}}})
\end{array}
$$

Cube 3 Dimension, 8 points

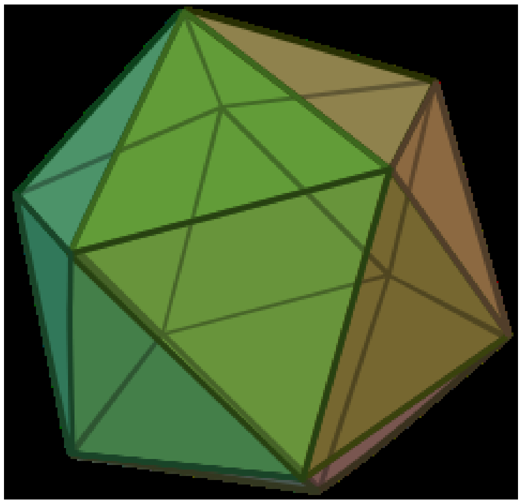


$$
\begin{array}{rrr}
(\ 1, & 1, & 1), \\
(-1, & 1, & 1), \\
(\ 1, & -1, & 1), \\
(-1, & -1, & 1), \\
(\ 1, & 1, & -1), \\
(-1, & 1, & -1), \\
(\ 1, & -1, & -1), \\
(-1, & -1, & -1)
\end{array}
$$

Icosahedron 3 Dimension, 12 points
Have edge length 2, forming five sets of 3 mutually centered vertices.

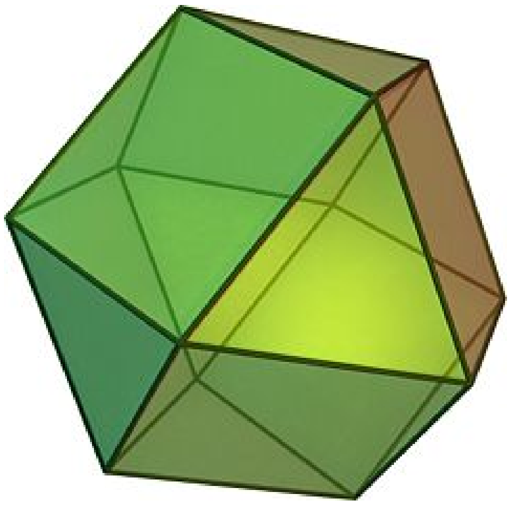
$$\begin{array}{rrr}
(\ 0, & \pm 1, & \pm\frac{1+\sqrt{5}}{2}), \\
(\pm 1, & \pm\frac{1+\sqrt{5}}{2}, & 0), \\
(\pm\frac{1+\sqrt{5}}{2}, & 0, & \pm 1)
\end{array}$$

These vectors are normalized so that they have length one, yielding the following:



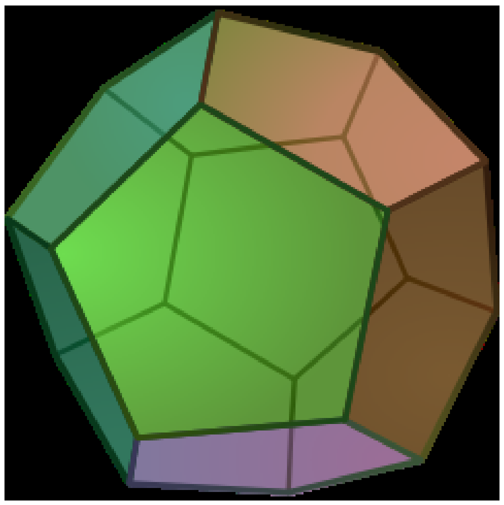
$$\begin{array}{rrr}
(\ 0.85065, & 0, & -0.52573), \\
(\ 0.52573, & -0.85065, & 0), \\
(\ 0, & -0.52573, & 0.85065), \\
(\ 0.85065, & 0, & 0.52573), \\
(-0.52573, & -0.85065, & 0), \\
(\ 0, & 0.52573, & -0.85065), \\
(-0.85065, & 0, & -0.52573), \\
(-0.52573, & 0.85065, & 0), \\
(\ 0, & 0.52573, & 0.85065), \\
(-0.85065, & 0, & 0.52573), \\
(\ 0.52573, & 0.85065, & 0), \\
(\ 0, & -0.52573, & -0.85065)
\end{array}$$

Cuboctahedron 3 Dimensions, 12 points



$$
\begin{array}{rrrl}
(0, & 1, & 1), \\
(0, & 1, & -1), \\
(0, & -1, & 1), \\
(0, & -1, & -1), \\
(1, & 0, & 1), \\
(1, & 0, & -1), \\
(-1, & 0, & 1), \\
(-1, & 0, & -1), \\
(1, & 1, & 0), \\
(1, & -1, & 0), \\
(-1, & 1, & 0), \\
(-1, & -1, & 0)
\end{array}
$$

Dodecahedron 3 Dimensions, 20 points

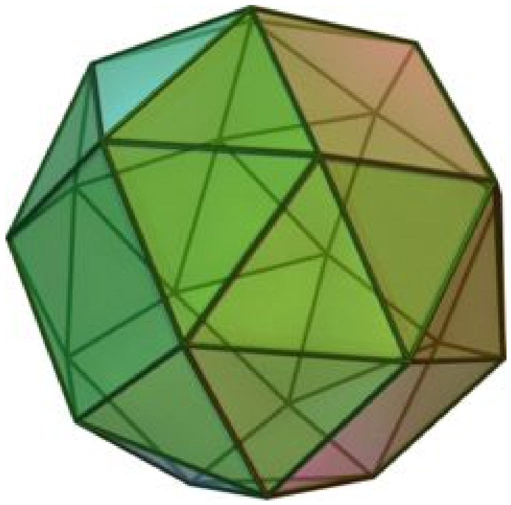


$$
\begin{array}{lrrr}
(& 0, & \frac{1}{\sqrt{3}\frac{1+\sqrt{5}}{2}}, & \frac{\frac{1+\sqrt{5}}{2}}{\sqrt{3}}), \\
(& 0, & -\frac{1}{\sqrt{3}\frac{1+\sqrt{5}}{2}}, & -\frac{\frac{1+\sqrt{5}}{2}}{\sqrt{3}}), \\
(& 0, & \frac{1}{\sqrt{3}\frac{1+\sqrt{5}}{2}}, & -\frac{\frac{1+\sqrt{5}}{2}}{\sqrt{3}}), \\
(& 0, & -\frac{1}{\sqrt{3}\frac{1+\sqrt{5}}{2}}, & \frac{\frac{1+\sqrt{5}}{2}}{\sqrt{3}}), \\
(& \frac{\frac{1+\sqrt{5}}{2}}{\sqrt{3}}, & 0, & \frac{1}{\sqrt{3}\frac{1+\sqrt{5}}{2}}), \\
(& -\frac{\frac{1+\sqrt{5}}{2}}{\sqrt{3}}, & 0, & -\frac{1}{\sqrt{3}\frac{1+\sqrt{5}}{2}}), \\
(& -\frac{\frac{1+\sqrt{5}}{2}}{\sqrt{3}}, & 0, & \frac{1}{\sqrt{3}\frac{1+\sqrt{5}}{2}}), \\
(& \frac{\frac{1+\sqrt{5}}{2}}{\sqrt{3}}, & 0, & -\frac{1}{\sqrt{3}\frac{1+\sqrt{5}}{2}}), \\
(& \frac{1}{\sqrt{3}\frac{1+\sqrt{5}}{2}}, & \frac{\frac{1+\sqrt{5}}{2}}{\sqrt{3}}, & 0), \\
(& -\frac{1}{\sqrt{3}\frac{1+\sqrt{5}}{2}}, & -\frac{\frac{1+\sqrt{5}}{2}}{\sqrt{3}}, & 0), \\
(& \frac{1}{\sqrt{3}\frac{1+\sqrt{5}}{2}}, & -\frac{\frac{1+\sqrt{5}}{2}}{\sqrt{3}}, & 0), \\
(& -\frac{1}{\sqrt{3}\frac{1+\sqrt{5}}{2}}, & \frac{\frac{1+\sqrt{5}}{2}}{\sqrt{3}}, & 0), \\
(& \frac{1}{\sqrt{3}}, & \frac{1}{\sqrt{3}}, & \frac{1}{\sqrt{3}}), \\
(& -\frac{1}{\sqrt{3}}, & -\frac{1}{\sqrt{3}}, & -\frac{1}{\sqrt{3}}), \\
(& \frac{1}{\sqrt{3}}, & \frac{1}{\sqrt{3}}, & -\frac{1}{\sqrt{3}}), \\
(& -\frac{1}{\sqrt{3}}, & -\frac{1}{\sqrt{3}}, & \frac{1}{\sqrt{3}}), \\
(& \frac{1}{\sqrt{3}}, & -\frac{1}{\sqrt{3}}, & \frac{1}{\sqrt{3}}), \\
(& -\frac{1}{\sqrt{3}}, & \frac{1}{\sqrt{3}}, & -\frac{1}{\sqrt{3}}), \\
(& \frac{1}{\sqrt{3}}, & -\frac{1}{\sqrt{3}}, & -\frac{1}{\sqrt{3}}), \\
(& -\frac{1}{\sqrt{3}}, & \frac{1}{\sqrt{3}}, & \frac{1}{\sqrt{3}})
\end{array}
$$

Regular Snub Cube 3 Dimension, 24 points
Coordinates are all the even permutations of the following vertices: $(\pm 1, \pm\xi, \pm\frac{1}{\xi})$ where,

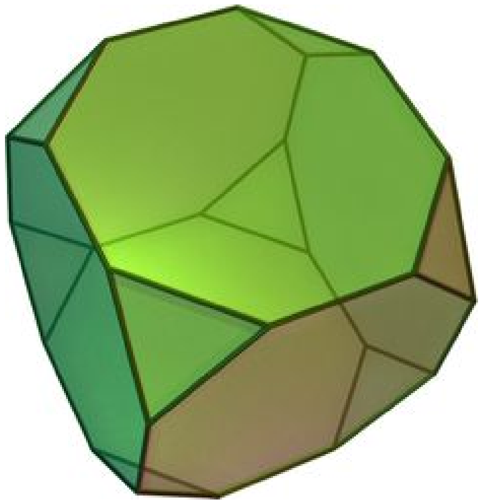
$$\xi = \frac{1}{3}(\sqrt[3]{17+3\sqrt{33}} - \sqrt[3]{-17+3\sqrt{33}} - 1$$

These vectors are normalized so that they have length one.



$$
\begin{array}{lrrr}
(& .85034, & .46232, & .25135), \\
(& .85034, & -.46232, & -.25135), \\
(& .85034, & .25135, & -.46232), \\
(& .85034, & -.25135, & .46232), \\
(& -.85034, & .46232, & -.25135), \\
(& -.85034, & -.46232, & .25135), \\
(& -.85034, & .25135, & .46232), \\
(& -.85034, & -.25135, & -.46232), \\
(& .25135, & .85034, & .46232), \\
(& -.25135, & .85034, & -.46232), \\
(& -.46232, & .85034, & .25135), \\
(& .46232, & .85034, & -.25135), \\
(& -.25135, & -.85034, & .46232), \\
(& .25135, & -.85034, & -.46232), \\
(& .46232, & -.85034, & .25135), \\
(& -.46232, & -.85034, & -.25135), \\
(& .46232, & .25135, & .85034), \\
(& -.46232, & -.25135, & .85034), \\
(& .25135, & -.46232, & .85034), \\
(& -.25135, & .46232, & .85034), \\
(& .46232, & -.25135, & -.85034), \\
(& -.46232, & .25135, & -.85034), \\
(& .25135, & .46232, & -.85034), \\
(& -.25135, & -.46232, & -.85034)
\end{array}
$$

Truncated Cube 3 dimensions, 24 points
Archimedian Solid with edge length $\sqrt{2}-1$.



$$\begin{array}{lrrr}
(& (\sqrt{2}-1), & 1, & 1),\\
(& (-\sqrt{2}-1), & 1, & 1),\\
(& (\sqrt{2}-1), & -1, & 1),\\
 & (-\sqrt{2}-1, & -1, & 1),\\
(& (\sqrt{2}-1), & 1, & -1),\\
(& (-\sqrt{2}-1), & 1, & -1),\\
(& (\sqrt{2}-1), & -1, & -1),\\
(& (-\sqrt{2}-1), & -1, & -1),\\
(& 1, & (\sqrt{2}-1), & 1),\\
(& -1, & (\sqrt{2}-1), & 1),\\
(& 1, & (-\sqrt{2}-1), & 1),\\
(& -1, & (-\sqrt{2}-1), & 1),\\
(& 1, & (\sqrt{2}-1), & -1),\\
(& -1, & (\sqrt{2}-1), & -1),\\
(& 1, & (-\sqrt{2}-1), & -1),\\
(& -1, & (-\sqrt{2}-1), & -1),\\
(& 1, & 1, & (\sqrt{2}-1)),\\
(& -1, & 1, & (\sqrt{2}-1)),\\
(& 1, & -1, & (\sqrt{2}-1)),\\
(& -1, & -1, & (\sqrt{2}-1)),\\
(& 1, & 1, & (-\sqrt{2}-1)),\\
(& -1, & 1, & (-\sqrt{2}-1)),\\
(& 1, & -1, & (-\sqrt{2}-1)),\\
(& -1, & -1, & (-\sqrt{2}-1))
\end{array}$$