HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **A:** Research Reports — ISSN 0783–5396
No. **32**; February 1995 — ISBN 951-22-2443-7

CONSTRUCTING SPHERICAL CODES BY GLOBAL OPTIMIZATION METHODS

Kari J. Nurmela

Digital Systems Laboratory
Department of Computer Science
Helsinki University of Technology
Otaniemi, Finland

Helsinki University of Technology
Department of Computer Science
Digital Systems Laboratory
Otaniemi, Otakaari 1
FIN–02150 ESPOO, FINLAND

HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **A**: Research Reports
No. **32**; February 1995

ISSN 0783–5396
ISBN 951-22-2443-7

Constructing Spherical Codes by Global Optimization Methods

Kari J. Nurmela

Abstract: In this work we search for spherical codes in three to five dimensions using different global optimization methods. In three dimensions the problem of finding optimal spherical codes is the Tammes problem, which asks, how to pack k circular caps on the surface of a sphere so that the diameter of the caps is as large as possible. The optimal codes are known only when k is small. The problem of spherical codes has close connections to sphere packings and kissing numbers.

A variable repulsion-force based approximation of the packing problem is used. This method has been used previously in three dimensions and it is generalized into n dimensions in this work. The potential energy of the point configuration is minimized by using ordinary local optimization algorithms and stochastic global optimization algorithms. New methods that are used employ problem-specific heuristics with simulated annealing and tabu search. It is shown how to reduce the number of optimization variables by restricting the solutions to have certain symmetries.

Three new 3-dimensional codes with 43, 77 and 80 code points are given. Most of the 4- and 5-dimensional codes in this work are better than the previously known codes.

Keywords: Spherical codes, spherical circle-packings, global optimization, stochastic optimization, combinatorial optimization.

Printing: TKK Monistamo; Otaniemi 1995

Helsinki University of Technology
Department of Computer Science
Digital Systems Laboratory
Otaniemi, Otakaari 1
FIN–02150 ESPOO, FINLAND

Phone: $\dfrac{90}{+358\text{–}0}$ 4511
Telex: 125 161 htkk fi
Telefax: +358–0–465 077
E-mail: lab@saturn.hut.fi

Contents

Chapter 1

Introduction

In this chapter we review the background of spherical codes including their relations to some other mathematical problems. We also give a list of applications and discuss the organization of this work.

1.1 Background

Spherical codes or, equivalently, spherical circle-packings are structures that have been studied in connection with geometry, chemistry, biology, engineering and optimization. Spherical codes are configurations of k points on an n-dimensional unit sphere such that the Euclidean distance between any two points is not smaller than a specified constant. It is easy to visualize the configurations in two and three dimensions. In two dimensions the points lie on the periphery of the unit circle, while in three dimensions they are on the surface of the unit sphere.

Finding good spherical codes can be regarded as an optimization problem, because it is often advantageous to have the minimum distance between the points of the code as large as possible, to obtain an optimal spherical code in a given dimension and with a given number of points. The optimal code in two dimensions is obtained trivially by dividing the periphery into k equal parts, see Figure 1.1.

However, in three or higher dimensions the problem of finding a globally optimal spherical code is very difficult. The problem can be stated equivalently as a packing problem: How to pack non-overlapping identical circles (or spherical caps) on the unit sphere such that the size of the circles is as large as possible? The three-dimensional case of this problem and the problem whether the optimal packing is essentially unique were probably first raised by the Dutch botanist Tammes in connection with the distribution of pores on pollen grains

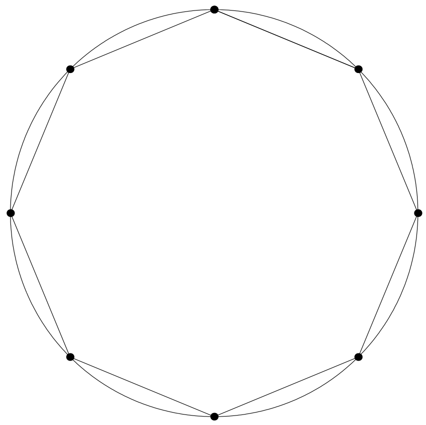

Figure 1.1: The optimal two-dimensional spherical code with eight points.

(Tammes, 1930). Sometimes the problem is stated in yet another equivalent form: Where should k inimical dictators build their palaces on a planet so as to be as far away from one another as possible? In spite of the enormous amount of literature on these problems, the optimal solutions in the three-dimensional case are known only for $k \leq 12$ and $k = 24$ (Croft, Falconer and Guy, 1991). In Figure 1.2 a spherical code is represented as a polyhedron whose vertices are the points of the code (lying on the unit sphere). In this case the polyhedron is an icosahedron, one of the Archimedean regular polyhedra which have the property that each edge has the same length. It is the optimal spherical code in three dimensions with 12 points.

The problem of finding optimal spherical codes is usually regarded as a problem of *discrete geometry*, because it is possible to compose a finite number of point configurations which necessarily contain the optimal packing by considering the so-called *rigid frameworks*, see Section 2. However, in practice this method is computationally infeasible when the number of circles is not very small.

There are some classical open problems in mathematics which are closely related to the problem of optimal spherical codes. One of them is the *kissing number* problem, which in three dimensions dates at least back to Newton: Given an n-dimensional sphere, how many equal spheres can be packed around it so that all of them touch (*kiss*) it? The terminology is borrowed from billiards. This problem has a vast amount of literature and many fascinating results are known, culminating in the maximal kissing number known in 24 dimensions, $\Lambda_{24} = 196560$. In addition to that, the maximal kissing numbers are known for dimensions 1, 2, 3 and 8 (Conway and Sloane, 1988). If optimal spherical codes were known for all dimensions and for all numbers of points, we could determine the maximal kissing numbers by just finding the largest number of points in optimal codes in a given dimension such that the minimum distance between the points is at least one, see Figure 1.3 (assuming that the

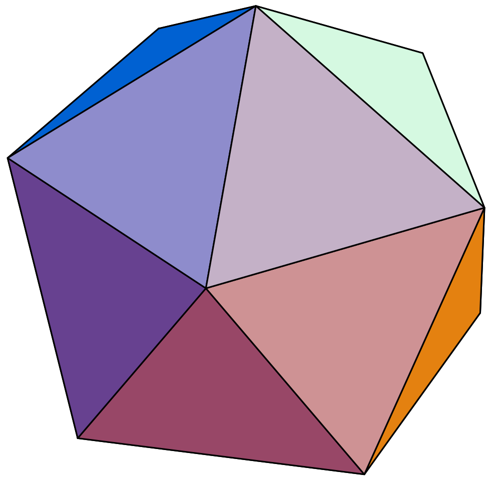

Figure 1.2: The optimal spherical code in three dimensions with 12 points.

spheres have radius 1, as it is assumed throughout this work). This reflects the hardness of the problem of finding good spherical codes.

In this work we search for spherical codes using different cost functions and optimization algorithms. Most of the time no symmetry constraints are imposed in the optimization processes, and many of the codes found possess little or no symmetry at all. In many cases the best spherical codes currently known indeed do not have strong symmetries, which is a drawback in the sense that using this kind of codes in practical applications is difficult: the only possibility is to tabulate the coordinates of all the code vectors in a codebook—while codes with more regular structures could be coded and decoded faster and using less memory.

It is possible to construct spherical codes by using results obtained in another classical problem, the *sphere packing problem*. The sphere packing problem asks, informally, how the n-dimensional space has to be 'filled' with identical n-dimensional hyperspheres such that the density of the packing is maximal. The density is the volume of the hyperspheres divided by the volume of the space and its value is obtained as a limiting value when the space under inspection tends to the whole n-dimensional space. The solution to this problem was known for a long time only in the 1- and 2-dimensional cases; the solution to the latter being the well-known hexagonal packing of circles in the plane (Figure 1.4). Recently, Hsiang (1993) gave a proof that the optimal packing is obtained by packing spheres of radius $1/\sqrt{2}$ layer by layer so that the centers

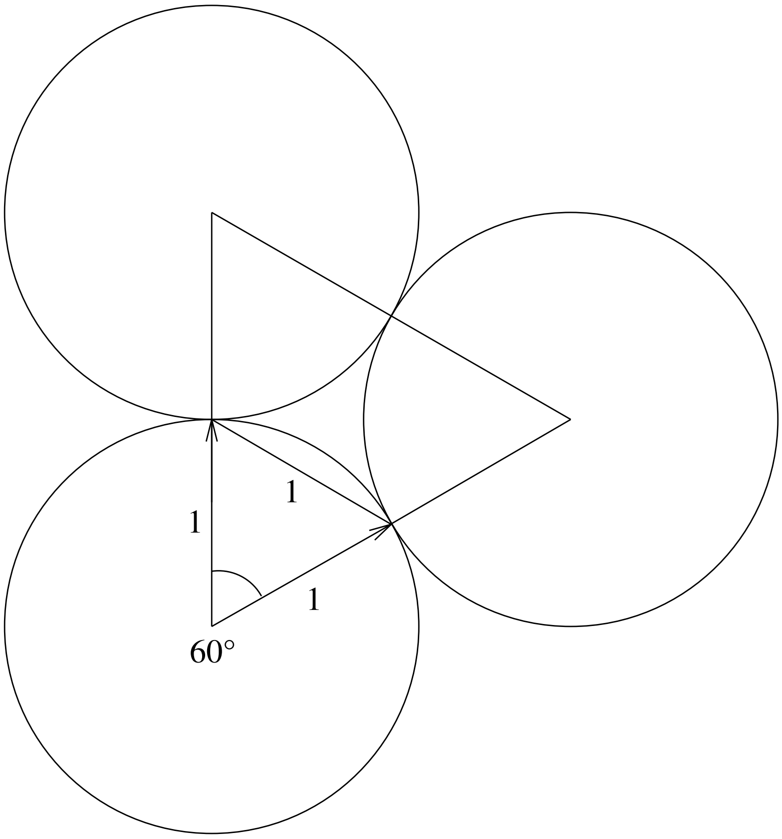

Figure 1.3: The connection between kissing numbers and spherical codes.

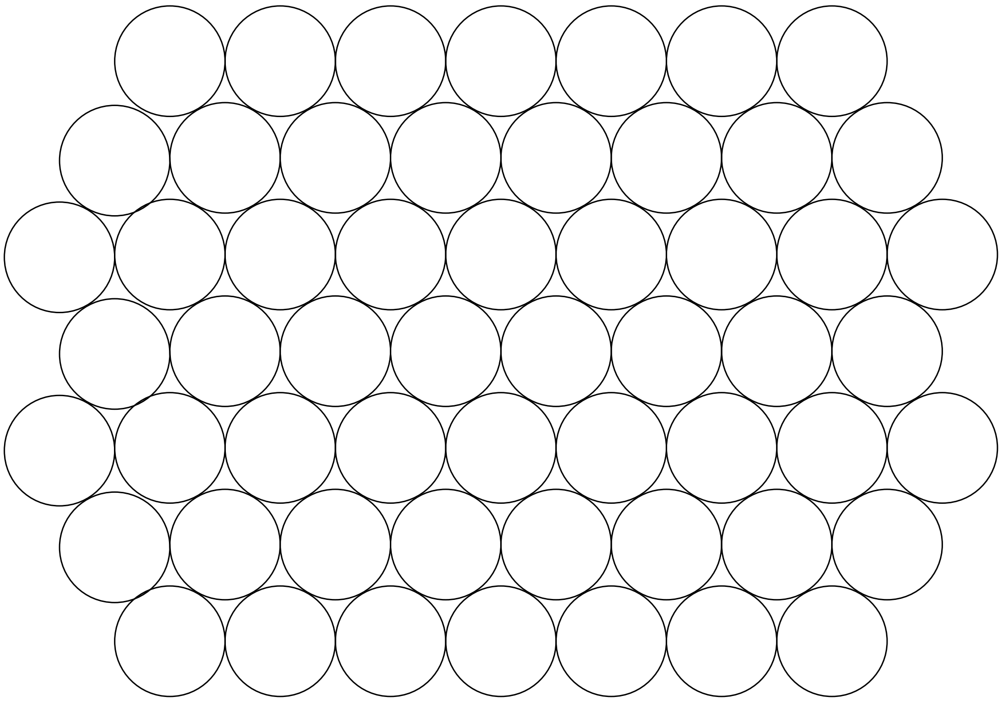

Figure 1.4: Hexagonal packing of circles in the plane.

of the spheres coincide with the integer coordinates (x, y, z) where the sum $x + y + z$ is even (so-called *face-centered cubic lattice*). This was the third part of the 18th problem on the famous list of Hilbert's (Hilbert, 1901). The result is not surprising, because, quoting Rogers (Conway and Sloane, 1988), "Many mathematicians believe and all physicists know" that no packing can be denser than the face-centered cubic lattice. This was conjectured by Kepler as early as in 1611, and during the last few centuries many great mathematicians have tried to prove it without success. Hsiang's proof is a very complex geometric proof and its status is not yet completely clear (Hales, 1992; Hales, 1993; Muder, 1993).

Hsiang (1994) points out that the geometry of spheres is not adequately understood even today, though its development started over two thousand years ago in connection with astronomy. In higher dimensions the optimal packings are not known. It has been conjectured that in 4, 8 and 24 dimensions the optimal packings are given by the packings defined by D_4, E_8 and Leech lattices, respectively (Hsiang, 1993).

Given a sphere packing, we can construct a spherical code by taking all the centers of the spheres of the packing whose distance from origin is equal to a given value (a *shell* of the packing). Then we project these points on the surface of a unit sphere around the origin. See Section 2.2 for a more formal presentation of the method. The drawback with this method is that given a desired number of points in the code, it may be possible that no good such code can be constructed by this method. For example, if a point in a lattice packing is taken, there is always another point symmetrically located on the opposite side of origin, i.e. codes with odd number of points cannot be constructed with this method unless we have a suitable nonlattice packing. Much more is known about lattice packings than about nonlattice packings.

1.2 Applications of Spherical Codes

Maybe the most important applications of spherical codes appear in spread-spectrum communications. Spherical codes have been studied especially in connection with signal transmission in the Gaussian channel (Biglieri and Elia, 1976; Blake, 1971; Blake, 1972; de Buda, 1975; Landau, 1971; Slepian, 1968; Sloane, 1980; van der Waerden, 1961; Wyner, 1969).

The Gaussian white noise channel transmits continuous signals. All the frequencies above a cutoff frequency W cycles per second are attenuated completely; frequencies below W are passed without attenuation. In the course of transmitting the signal, the channel adds white Gaussian noise to it. The noise is independent of the signal.

Shannon's sampling theorem (Shannon, 1948) states that a continuous signal which does not contain frequencies above $W/2$ cycles per second can be perfectly reconstructed from samples taken at frequency W per second. Now the transmitted signal $f(t)$ can be characterized by a point $F = (f_1, \ldots, f_n)$ in n-dimensional space (corresponding to the n samples taken). The Gaussian channel adds a noise vector $Y = (y_1, \ldots, y_n)$ and a vector $F + Y$ is obtained after sampling the received signal. We are interested in defining a set of code vectors such that after receiving a signal we could find the nearest code vector to it and effectively correct any small errors occurred during the transmission.

By increasing the energy of the code vectors, the proportion of noise becomes smaller and the error-correcting easier. In practice, the energy of the code vectors is limited and we are facing the problem of packing small spheres into a larger one. In certain cases, for example when the components of the source vectors are independent and normally distributed, the distribution of the source vectors tends to a hypersphere as the number of dimensions increases. Then it is natural to consider a spherical circle-packing instead of packing small spheres into a larger one, that is, we are searching for *constant-energy codes*, codes in which every signal has the same energy. If the amount of Gaussian noise in the channel is relatively small, then finding the best constant energy code with k code vectors of length n for that channel is approximately the same thing as finding an optimal spherical code in n dimensions with k points (Conway and Sloane, 1988).

Three-dimensional spherical codes, spherical circle-packings, appear surprisingly often in microbiological objects in nature. Tammes presented the problem after finding that certain spherical pollen grains have pores that tend to be dispersed as far from one another as possible (Tammes, 1930). See (Tarnai, 1984) for electron micrographs of the pollen grains of *Psilotrichum gnaphalobryum*, *Psilotrichum elliotti*, *Celosia argentea*, *Volkensinia prostrata* and *Cyathula orthacantha*. In these pollen grains the number of pores is well within the range of number of circles studied in this work.

Various researches have noted that similar structures occur on the surface of some species of radiolarians (Kling, 1978) (with larger number of circles), foraminifers, sponges (Kirkpatrick and Searle, 1975), mould fungus (Fujita, Tokunaga and Inoue, 1971), blood platelets and green algae (Tarnai, 1984). The compound eyes of insects resemble these structures, although their shape is not exactly spherical. Certain icosohedral viruses have structures that resemble spherical circle-packings, but their shape is more accurately described by solutions to a generalization of the Tammes problem (Goldberg, 1967; Molnár, 1975). Tarnai (1984) gives many examples of these with illustrative figures.

Spherical circle-packing has inspired many artists, architects and engineers.

The familiar shape of a golf ball, where the small round dimples were originally added for aerodynamical reasons, is an example of a spherical circle-packing. The number of circles in this packing is usually rather high, in a conventional golf ball there are 336 of them.

Some other examples in engineering include the Laser's geodynamic satellite, a compound camera for Polyvision moving pictures and the projector of Bauersfeld planetarium in München in 1924 (Tarnai, 1984). Sometimes the circles are made nonequal because of technical reasons, for example in the typewriter printing balls, which were quite common a couple of years ago.

Spherical circle-packings have been applied in architecture for hundreds of years (Tarnai, 1984). The skylight openings of the Turkish baths in Budapest built in the 16th century show an early example, although in these arrangements the diameter of the circles drawn around the openings decreases towards the zenith of the sphere. A more recent example can be found in R. B. Fuller's Fly Eye domes.

Spherical circle-packing has occurred in the fine arts, too. H. Bosch's triptych *Garden of Delights* from the beginning of the 16th century shows a close circle-packing in the bottom part of the Fountain of Life. B. Józsa has created statues based on spherical circle-packings.

Some arrangements of atoms around a central atom or cluster of atoms can be modelled as low-energy configurations of equation (3.2) in Section 3.3 when $m \approx 3$ (Clare and Kepert, 1986), whereas spherical circle-packings are low-energy configurations of the same function when m tends to infinity. Another related problem—the problem of distributing equal point charges on the surface of a sphere (Edmundson, 1991; Frickel and Bronk, 1987) can be modelled with $m = 1/2$. There are also some spherical molecules, for example *fullerines*, which have a structure closely resembling a spherical circle-packing.

1.3 Organization of This Work

This work is organized as follows. In Chapter 2 we give the formal definitions of spherical codes and related concepts. We also review some basic results and give the definition and basic properties of hyperspherical coordinates which are used to make the optimization problem unconstrained. Chapter 3 presents the most successful optimization approaches to spherical code construction found in literature. We also present the new methods that are applied to this problem in this work. These include some problem-specific heuristics and combinatorial optimization algorithms, which are used together with continuous stochastic global optimization algorithms. Chapter 4 shows some ways to visualize spherical codes in three dimensions. Chapter 5 lists the new codes found and discusses some implementation issues including data structures and the software tools used. Finally, in Section 6 we conclude this work. The coordinates of the new codes are listed in the Appendix.

Chapter 2

Preliminaries

In this chapter we give formal mathematical definitions of the concepts discussed in the Introduction. We also review some of the basic results in this area. A spherical code construction method by (Sloane, 1981) is presented. We review some basic terminology of optimization of nonlinear continuous problems. The problems of generating random initial solutions and fine-tuning the near-optimal solutions are discussed.

2.1 Definitions and Some Basic Results

We begin by defining the concept of a spherical code. In our definition we do not require that all the code points are distinct, although that is assumed in the rest of this work, because a code with two or more points coinciding is useless in our purposes.

Definition 1 *A* spherical code *is a finite collection of points on the n-dimensional unit sphere Ω_n.*

In this work we assume for simpler notation that the limiting sphere is always a unit sphere centered in the origin. We denote a spherical code S with k points by listing the vectors corresponding to the points of the code, i.e. we write

$$S = (\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k)\,,$$

where

$$\mathbf{s}_i = \begin{pmatrix} s_{i,1} \\ s_{i,2} \\ \vdots \\ s_{i,n} \end{pmatrix}.$$

We use bold lower case letters to denote vectors and bold upper case letters to denote matrices. We write $\mathbf{x} \in S$ to denote the fact that $\mathbf{x} = \mathbf{s}_i$ for some $1 \le i \le k$. The cardinality of the code is $|S| = k$.

For each code vector $\mathbf{x}$,

$$\mathbf{x}^{\mathrm{T}}\mathbf{x} = \sum_{i=1}^{n} x_i^2 = 1,$$

because each point is on the unit sphere.

The most important property of a spherical code S is the minimum distance between any two code vectors

$$\delta(S) = \min\left\{\|\mathbf{s}_i - \mathbf{s}_j\| : 1 \le i < j \le k\right\},$$

where the vector norm $\|\cdot\|$ is the Euclidean norm

$$\|\mathbf{x}\| = \sqrt{\sum_{i=1}^{n} x_i^2}.$$

Equivalently, we could use the minimum angle between any two code vectors, denoted by $\angle(S)$, the maximal dot product between code vectors, or the density of the code, denoted by $D(S)$, which is defined later in this chapter. It is easy to see that if we have an infinite set of points on the surface of the n-dimensional unit sphere, then for any positive constant t we can find a pair of points such that the distance between them is smaller than t. This is why we defined a spherical code to be a *finite* set of points.

The maximal minimum distance among all the spherical codes with k points in n dimensions is denoted by $\delta_n^{\max}(k)$. As mentioned in the Introduction, the determination of the exact values of $\delta_n^{\max}(k)$ is a very hard problem when $n \ge 3$.

Biorthogonal spherical codes (Lachaud and Stern, 1994) are codes with $2n$ code vectors in n dimensions. The points are located at $(\pm 1, 0, 0, \ldots, 0)$, $(0, \pm 1, 0, \ldots, 0)$, $(0, 0, \pm 1, \ldots, 0)$, $\ldots$, $(0, 0, 0, \ldots, \pm 1)$ in cartesian coordinates. The minimum distance is $\sqrt{2}$.

It is sometimes useful to divide the pairs of point indices into equivalency classes $E_1, E_2, \ldots$ by

$$\begin{aligned} E_1(S) &= \left\{\{i,j\} : \|\mathbf{s}_i - \mathbf{s}_j\| = \min\left\{\|\mathbf{s}_t - \mathbf{s}_v\| : 1 \le t < v \le k\right\}\right\}, \\ E_i(S) &= \left\{\{i,j\} : \|\mathbf{s}_i - \mathbf{s}_j\| = \min\left\{\|\mathbf{s}_t - \mathbf{s}_v\| : \begin{array}{c} 1 \le t < v \le k \text{ and} \\ \{t,v\} \notin \bigcup_{l=1}^{i-1} E_l(S) \end{array}\right\}\right\}. \end{aligned}$$

The *set of the shortest edges* is $E_1(S)$, the set of the second shortest edges is $E_2(S)$, and so on. In practice, numerical inaccuracies may be a problem when constructing the set hierarchy.

Now we can define the maximum kissing number by

$$\Lambda_n = \max\{|S| : S \text{ is an } n\text{-dimensional spherical code with } \delta(S) \geq 1\}\,.$$

Note that traditionally the term *kissing number* means the *maximal* kissing number. The explicit use of the word 'maximal' in this work has been adopted from (Conway and Sloane, 1988).

Definition 2 *A* sphere packing Λ *is a countably infinite set of points in* $\mathbb{R}^n$ *such that for all* $\mathbf{x}, \mathbf{y} \in \Lambda$ *it holds that* $\|\mathbf{x} - \mathbf{y}\| \geq 2$.

The density of a sphere packing is obtained as the limit value

$$\Delta(\Lambda) = \lim_{r \to \infty} \frac{|\{\mathbf{x} : \mathbf{x} \in \Lambda, \|\mathbf{x}\| \leq r\}|}{r^3},$$

where we have used the fact that the sphere packings in this work are assumed to consist of unit spheres. It is sometimes convenient to scale the packing so that the spheres have radius different from one (Sloane, 1981). We obtain the density of the packing informally by taking larger and larger spherical origin-centered regions and count the volume occupied by the spheres whose centers fall in that region and divide it by the volume of the region.

While explicit constructions of spherical codes provide lower bounds for the maximal minimum distance, the upper bounds are obtained by some analytical results. For example in three dimensions we can use two bounds by Robinson (1961) which prove that the best known packing of 24 circles (the so-called '*snub cube*', an Archimedean semiregular polyhedron, see Figure 4.2 on page 59) is optimal.

In (Mackay, 1980) the minimum distance of an optimal code in four dimensions is approximated by a simple volume argument. However, their approximation is not a true lower bound neither a true upper bound and there is no estimate of the error of the approximation.

Note that if an n-dimensional code S has minimum angle $\angle(S)$, we can place k non-overlapping circular n-dimensional cones, one for each code point, with conical angle $\angle(S)$ around the code points so that the tips of the cones are located in the origin. We define here the *density* of a code as the proportion of the area of the unit sphere that intersects with these cones. This definition is equivalent to the definition of the circle-packing density on the surface of the three-dimensional sphere and it helps to some extent to compare the goodness of different codes with different numbers of code vectors.

The density can be calculated as follows. Let the area of an n-dimensional sphere with radius r be $A_n(r)$. The area intersected by one of the cones with

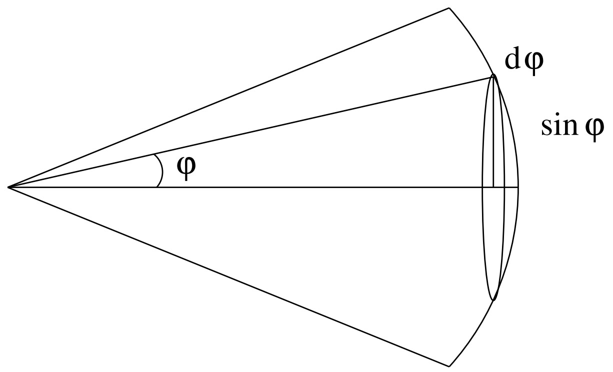

Figure 2.1: The area intersected by a cone

conical angle α can be calculated by

$$A_n^c(\alpha) = \int_0^{\alpha/2} A_{n-1}(\sin\varphi)\, d\varphi.$$

The three-dimensional case is illustrated in Figure 2.1. The ring in the figure has the width $d\varphi$. It is an intersection of two n-dimensional spheres, thus it is an $(n-1)$-dimensional sphere (Sommerville, 1958). Its area is $A_{n-1}(\sin\varphi)\, d\varphi$.

The area of an n-dimensional sphere is obtained by (Sommerville, 1958)

$$A_n(r) = \frac{2\pi^{n/2} r^{n-1}}{\Gamma(n/2)}, \tag{2.1}$$

which can be rewritten when n is an integer as

$$\begin{aligned} A_n(r) &= \frac{2\pi^{n/2} r^{n-1}}{(n/2-1)!}, \quad \text{when } n \text{ is even, and} \\ A_n(r) &= \frac{2\pi^{n-1}(n-1)!(2r)^{2(n-1)}}{(2n-2)!}, \quad \text{when } n \text{ is odd.} \end{aligned}$$

This gives

$$\begin{aligned} A_2(r) &= 2\pi r, \\ A_3(r) &= 4\pi r^2, \\ A_4(r) &= 2\pi^2 r^3, \\ A_5(r) &= \frac{8}{3}\pi^2 r^4, \end{aligned}$$

and thus

$$\begin{aligned} A_3^c(\alpha) &= -2\pi(\cos(\alpha/2) - 1), \\ A_4^c(\alpha) &= \pi(\alpha - \sin(\alpha)), \\ A_5^c(\alpha) &= \frac{2}{9}\pi^2 \left(-9\cos(\alpha/2) + \cos(3\alpha/2) + 8\right). \end{aligned}$$

The cones are non-overlapping, hence the density is calculated by summing the areas corresponding to each cone and by dividing the sum by the area of the unit hypersphere, thus the density is

$$D(S) = \frac{kA_n^c(\alpha)}{A_n(1)}.$$

A simple area argument yields the bound

$$D(S) < 1.$$

If we know bounds for the maximal sphere packing density in $n-1$ dimensions (we denote this by Δ_{n-1}^{max}), we can use the asymptotic approximation

$$D_n^{max} \approx \Delta_{n-1}^{max}$$

for the maximal code density in n dimensions which is accurate when the number of code vectors is large. The claim can be justified informally by approximating the hypersphere in n dimensions by a polyhedron whose facets are hyperplanes, i.e. regions in $n-1$ dimensional space. These spaces can be filled with small spheres roughly with the density Δ_{n-1}^{max}. Note that due to the finiteness of these regions the actual packing density may in principle be even greater than Δ_{n-1}^{max}—if $n \geq 4$, see (Boerdijk, 1952)—so the approximation is not necessarily an upper bound.

In most cases the set of shortest edges in an optimal packing form a *rigid framework*, where no circle can move infinitesimally without overlapping another circle and the only infinitesimal movements that all the circles together can do without introducing overlaps are rotations around the origin. The rigid framework of a code S is denoted by $R(S)$ which is a set of point indices. The index of a point belongs to this set if and only if the circle belongs to the rigid framework. It is obvious that there are only finitely many possible sets of shortest edges. Now if we could construct all the packings corresponding to a set of shortest edges, the search of the optimal packing would, in theory, reduce to examining a finite number of packings and selecting the best of them. In practice the combinatorial explosion makes this approach infeasible with all but the smallest codes. Efficient pruning strategies might make this method useful in moderately small problem instances.

The problem is made more difficult by the fact that in some optimal packings the rigid framework is not unique (Tarnai and Gáspár, 1983). Take for example the optimal three-dimensional packing with five circles. This packing may be visualized as one having a circle on the north pole of the sphere, one on the south pole and the remaining three on the equator, see Figure 2.2. The three circles on the equator do not necessarily touch each other, so they may be moved independently along the equator without causing overlapping.

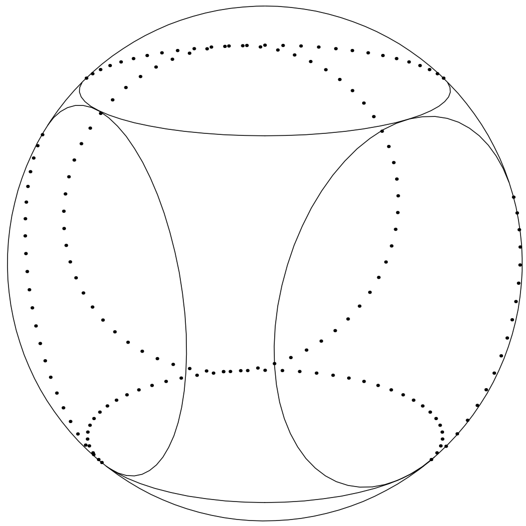

Figure 2.2: Optimal packing of five equal circles on the sphere.

In order to compare the configurations to the ones mentioned in the literature, we can use invariants that do not change in different orientations of the configuration. Additionally, we would like these invariants to remain constant even if a free circle is moved. Kottwitz (1991a) gives two such invariants,

$$A(S) = \frac{1}{z} \sum_{\substack{i,j \in R(S) \\ 1 \le i < j \le k}} \|\mathbf{s}_i - \mathbf{s}_j\|, \tag{2.2}$$

$$B(S) = \frac{1}{z} \sum_{\substack{i,j \in R(S) \\ 1 \le i < j \le k}} \|\mathbf{s}_i - \mathbf{s}_j\|^{-1}, \tag{2.3}$$

where

$$z = |R(S)| \cdot (|R(S)| - 1)/2.$$

These invariants are based on averaging the distances between points in the rigid framework.

2.2 Constructing Spherical Codes from Sphere Packings

Given a sphere packing Λ, let

$$L(\Lambda) = \{\|\mathbf{x}\| : \mathbf{x} \in \Lambda\}$$

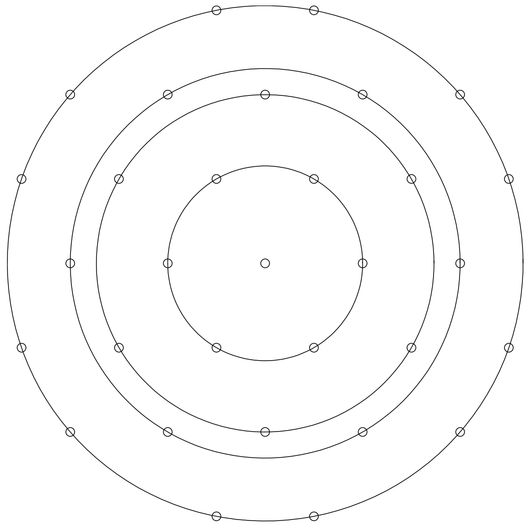

Figure 2.3: The first five shells of the hexagonal circle-packing in the plane.

be the set of the distances of the points from the origin. Clearly, we can sort the set into increasing order $(l_1, l_2, \ldots)$. Now we can divide the points of the sphere packing Λ into equivalence classes $\Lambda^{(1)}, \Lambda^{(2)}, \ldots$ according to

$$\Lambda^{(i)} = \{\mathbf{x} : \|\mathbf{x}\| = l_i\} .$$

Each class $\Lambda^{(i)}$ is called a *shell*. The first five shells of the hexagonal lattice packing in the plane are shown in Figure 2.3. Note that the distance corresponding to the first shell is zero as with all lattice packings, because there is always a sphere centered in the origin.

Now we can construct a spherical code by taking a shell and projecting the points of it onto the surface of the unit sphere. For a two-dimensional example, in Figure 2.3 the shells 2, 3 and 4 produce the optimal spherical code with 6 points in two dimensions after they are projected onto the unit circle around the origin. Shell number 5 contains 12 points, but because they are not uniformly distributed on the periphery of the circle it is not the optimal spherical code. In a similar way we can construct many spherical codes in higher dimensions from known sphere packings (Sloane, 1981).

In this work we construct spherical codes in three, four and five dimensions using the lattice packings D_3 (*face-centered cubic lattice*), $D_3^\perp$ (*body-centered cubic lattice*), $D_4 = D_4^\perp$, D_5, $D_5^\perp$ and a non-lattice packing, namely *hexagonal close-packing*. For most of these packings, the numbers of points in the first fifty or more shells are tabulated in (Sloane, 1981), but they do not give the minimum distances of the corresponding codes, so we have explicitly constructed the first few dozen shells and examined the codes that they produces. The results are discussed in Chapter 5. Generally the codes produced this way are not very dense, at least with these packings, but the construction method is easy even in larger dimensions with large number of points.

Points of a lattice packing can be generated if the *generator matrix* is known. A lattice sphere packing consists of all integer combinations of the rows of its generator matrix. We denote the generator matrix of a sphere packing Λ by $\mathbf{M}(\Lambda)$. We have (Sloane, 1981)

$$\begin{aligned}
\mathbf{M}(D_3) &= \begin{pmatrix} 2 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}, \\
\mathbf{M}(D_3^{\perp}) &= \begin{pmatrix} 1 & 1 & 1 \\ 2 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix}, \\
\mathbf{M}(D_4) &= \begin{pmatrix} 2 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}, \\
\mathbf{M}(D_5) &= \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}, \\
\mathbf{M}(D_5^{\perp}) &= \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \end{pmatrix}.
\end{aligned}$$

The hexagonal close-packing in three dimensions can be generated by first generating the lattice packing L spanned by

$$\mathbf{M}(L) = \begin{pmatrix} \sqrt{3} & 0 & 0 \\ \sqrt{3}/2 & 3/2 & 0 \\ 0 & 0 & 2\sqrt{2} \end{pmatrix}$$

and then adding the points of the coset

$$L' = \begin{pmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} & \sqrt{2} \end{pmatrix} + L.$$

Even though the results were not very encouraging compared to the other methods in this work when the numbers of code points and dimensions were small (see Chapter 5), better results might be obtained by using different lattices. For example we could take the points of an icosahedron and use them as a generating matrix and use the shells of that lattice as spherical codes having icosahedral symmetry.

2.3 Optimization of Continuous Functions

The prime object of this work is to apply different optimization methods for spherical code construction. It is necessary to define a few concepts common to some or all of the optimization methods used. We assume that the optimization problem is a minimization problem. A maximization problem could be solved by negating the cost function.

The aim of an minimization problem is to find values for a set of variables such that the *cost function* has as low value as possible. The *solution space* consists of all permitted values for the variables, and each point in the solution space denotes a solution for the problem.

Definition 3 *A* local optimum *is a point in the solution space such that no infinitesimal movement of one or several variable values results in cost decrease.*

Definition 4 *A* global optimum *is a point in the solution space such that no point has lower cost.*

For simplicity, we assume that the cost values are bounded. There may be one or several global optima.

The *Jacobian* or *gradient* of a scalar valued function $f(x_1, \ldots, x_m)$ is a vector consisting of the first partial derivatives of the function and it is defined as

$$\mathbf{J}(f) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_m} \end{pmatrix}.$$

In a local optimum the Jacobian of the cost function is a zero vector, if the partial derivatives exist in that point. In some cases the Jacobian is zero even if the point is not a local optimum. The point is then a *saddle point.* To determine whether or not a zero point of the Jacobian is a local optimum, we can use the Hessian of the cost function which is a matrix containing the second partial derivatives of the functions and it is defined by

$$\mathbf{H}(f) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_m} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_m)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_m \partial x_1} & \frac{\partial^2 f}{\partial x_m \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_m \partial x_m} \end{pmatrix}.$$

If the Hessian is positively definite ($\mathbf{y}^{\mathrm{T}}\mathbf{H}\mathbf{y} > 0$ for all $\mathbf{y} \neq \mathbf{0}$), then the point is a local optimum. If the Hessian is positively semidefinite ($\mathbf{y}^{\mathrm{T}}\mathbf{H}\mathbf{y} \geq 0$ for all $\mathbf{y} \neq \mathbf{0}$), then the point may or may not be a local optimum. In many local optimization algorithms the Hessian or its approximation is used to estimate the change of the current values of $x_1, \ldots, x_m$ so that the Jacobian goes to zero, because when the function is twice continuously differentiable and the difference $\|\mathbf{x}_2 - \mathbf{x}_1\|$ is small, then

$$\mathbf{J}(\mathbf{x}_2) - \mathbf{J}(\mathbf{x}_1) \approx \mathbf{H}(\mathbf{x}_2 - \mathbf{x}_1),$$

which holds exactly when the cost function is quadratic.

An important measure of the quality of optimization methods is their *convergence rate*. A sequence of cost values $c_1, c_2, \ldots$ converges to value c^*, if

$$\lim_{i \to \infty} |c_i - c^*| = 0.$$

The convergence is *linear* if there is a constant $0 \leq c < 1$ and an integer $N \geq 1$ such that

$$\text{for all } i \geq N: \quad |c_{i+1} - c^*| \leq c\,|c_i - c^*|.$$

If we have a sequence $c_1, c_2, \ldots$ which converges towards zero and it holds that

$$\text{for all } i \geq N: \quad |c_{i+1} - c^*| \leq c_i\,|c_i - c^*|,$$

then we say that the convergence is *superlinear*. The convergence is *quadratic* if there is a real-valued constant $c \geq 0$ and an integer $N \geq 0$ such that

$$\text{for all } i \geq N: \quad |c_{i+1} - c^*| \leq c\,|c_i - c^*|^2.$$

Cubic convergence is defined by replacing the exponent 2 by 3 and so on.

2.4 Removing Constraints by Using Hyperspherical Coordinates

An optimization problem with real-valued variables may be either constrained or unconstrained. In an unconstrained minimization problem any real values for the variables are acceptable and the goal is to find such values that the cost function is as small as possible.

In constrained optimization there are *constraints* that restrict the permitted values for the variables. Equality constraints are of form

$$f_i(x_1, x_2, \ldots, x_j) = 0,$$

while inequality constraints can be expressed in the form

$$f_i(x_1, x_2, \ldots, x_j) \le 0.$$

If the corresponding function f_i is linear, the constraint is called a *linear* constraint.

There are various methods for constrained optimization, but most of them work well only when the constraints are linear or they can be locally approximated by linear constraints with reasonable accuracy. Constrained optimization can be considerably more difficult than unconstrained optimization and the convergence may be significantly slower.

If we express the positions of the points of a spherical code in the usual cartesian coordinates, we have a constrained optimization problem

$$\begin{array}{rrcll} \text{variables:} & S & = & (s_{ij}); & 1 \le i \le k,\ 1 \le j \le n \\ \text{cost:} & c(S) & = & -\delta(S) & \\ \text{constraints:} & s_{i,1}^2 + s_{i,2}^2 + \ldots + s_{i,n}^2 & = & 1; & 1 \le i \le k. \end{array}$$

Now the cost is easy to calculate, but the nonlinear constraints make the optimization process slower. See Section 3.2 for discussion of the cost function $c(S) = -\delta(S)$.

It is possible to transform the problem into an unconstrained problem by using *spherical* (or hyperspherical) coordinates. In the rest of this work we use notation $\widetilde{\mathbf{x}} = (\widetilde{x}_1, \widetilde{x}_2, \ldots, \widetilde{x}_{n-1})$ to denote a representation (in hyperspherical coordinates) of a point $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ on the n-dimensional unit sphere. The mapping between these two coordinate systems is not bijective, because the trigonometric functions are periodic. There are infinitely many points in hyperspherical coordinates corresponding to a single point in cartesian coordinates.

Note that in spherical coordinates one coordinate less is needed than in cartesian coordinates, because we assume that the sphere radius is one. For example, in the simple two-dimensional case only one spherical coordinate is needed; we denote this by $\widetilde{x}_1$. Now for a point $\mathbf{x} = (x_1, x_2)$ we have the equations

$$\begin{aligned} x_1 &= \cos(\widetilde{x}_1), \\ x_2 &= \sin(\widetilde{x}_1), \end{aligned}$$

which determine a transformation from the spherical coordinates to the cartesian coordinates. It is trivial to check that if x_1 and x_2 satisfy these equations for some $\widetilde{x}_1$, then $x_1^2 + x_2^2 = 1$. It is also easy to find an $\widetilde{x}_1$ for any point that lies on the 2-dimensional unit sphere.

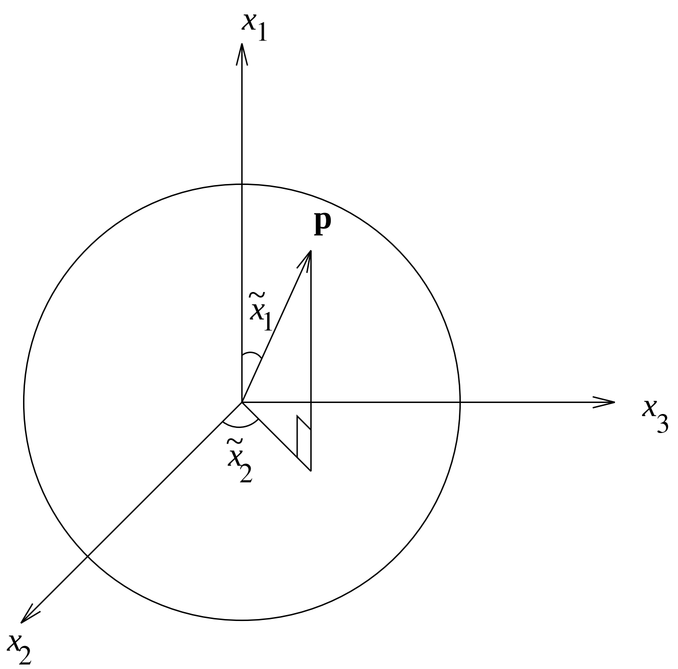

Figure 2.4: Spherical coordinates in three dimensions.

In the same way we can define the spherical coordinates in three dimensions by

$$
\begin{aligned}
x_1 &= \cos(\tilde{x}_1), \\
x_2 &= \sin(\tilde{x}_1)\cos(\tilde{x}_2), \\
x_3 &= \sin(\tilde{x}_1)\sin(\tilde{x}_2),
\end{aligned}
$$

see Figure 2.4. Generalizing this we obtain one way to define hyperspherical coordinates in n dimensions,

$$
\begin{aligned}
x_1 &= \cos(\tilde{x}_1) = T_1(x), \\
x_2 &= \sin(\tilde{x}_1)\cos(\tilde{x}_2) = T_2(x), \\
x_3 &= \sin(\tilde{x}_1)\sin(\tilde{x}_2)\cos(\tilde{x}_3) = T_3(x), \\
&\vdots \\
x_{n-1} &= \sin(\tilde{x}_1)\sin(\tilde{x}_2)\cdots\sin(\tilde{x}_{n-2})\cos(\tilde{x}_{n-1}) = T_{n-1}(x), \\
x_n &= \sin(\tilde{x}_1)\sin(\tilde{x}_2)\cdots\sin(\tilde{x}_{n-2})\sin(\tilde{x}_{n-1}) = \widehat{T}_{n-1}(x), \qquad (2.4)
\end{aligned}
$$

where the notations $\widehat{T}_t(x)$ and $T_t(x)$ are defined as

$$
\begin{aligned}
\widehat{T}_t(x) &= \prod_{i=1}^{t} \sin(\tilde{x}_i), \\
T_t(x) &= \cos(\tilde{x}_t)\prod_{i=1}^{t-1} \sin(\tilde{x}_i).
\end{aligned}
$$

It is easy to see that $\sum_{i=1}^{n} x_i^2 = 1$ by noting that

$$\begin{aligned} T_i(x)^2 + \widehat{T}_i(x)^2 &= \sin^2(\widetilde{x}_1)\sin^2(\widetilde{x}_2)\cdots\sin^2(\widetilde{x}_{i-1})\cos^2(\widetilde{x}_i) + \\ &\quad \sin^2(\widetilde{x}_1)\sin^2(\widetilde{x}_2)\cdots\sin^2(\widetilde{x}_{i-1})\sin^2(\widetilde{x}_i) \\ &= \sin^2(\widetilde{x}_1)\sin^2(\widetilde{x}_2)\cdots\sin^2(\widetilde{x}_{i-1})\underbrace{[\cos^2(\widetilde{x}_i) + \sin^2(\widetilde{x}_i)]}_{=1} \\ &= \widehat{T}_{i-1}(x)^2, \end{aligned}$$

and by adding the squares of x_i in the order $x_n^2 + x_{n-1}^2 + \ldots + x_1^2$.

In the general case the transformation from the cartesian coordinates into hyperspherical coordinates is slightly more complicated than in 2- or 3-dimensional case. One way of doing this is to take

$$\begin{aligned} \widetilde{x}_1 &= \arccos(x_1), \\ \widetilde{x}_2 &= \arccos(x_2/\sin(\widetilde{x}_1)), \\ \widetilde{x}_3 &= \arccos(x_3/\sin(\widetilde{x}_1)\sin(\widetilde{x}_2)), \\ &\vdots \\ \widetilde{x}_{n-2} &= \arccos(x_{n-2}/\sin(\widetilde{x}_1)\sin(\widetilde{x}_2)\cdots\sin(\widetilde{x}_{n-3})), \\ \widetilde{x}_{n-1} &= \operatorname{arctan2}(x_n, x_{n-1}), \end{aligned}$$

where $\operatorname{arctan2}(a, b)$ is the four quadrant inverse tangent which gives the direction angle of the vector from origin to the point (b, a) in the 2-dimensional cartesian coordinate system.

Later we shall introduce several cost functions that require calculation of distances between the code points. We have decided to use the Euclidean distance between the points, although the angles between the code vectors could also be used. No significant distinctions between these two alternatives can be found when used in the cost functions in this work. The Euclidean distance between points in hyperspherical coordinates can be expressed in two dimensions by

$$\|\mathbf{x} - \mathbf{y}\|^2 = 2 - 2\cos(\widetilde{x}_1 - \widetilde{y}_1),$$

which is obtained by noting that

$$\frac{1}{2}\|\mathbf{x} - \mathbf{y}\| = \sin\left(\frac{1}{2}\cdot\angle(\mathbf{x}, \mathbf{y})\right),$$

where $\angle(\mathbf{x}, \mathbf{y})$ is the angle between the two vectors $\mathbf{x}$ and $\mathbf{y}$. Now using the sine rule for half angles we obtain

$$\|\mathbf{x} - \mathbf{y}\|^2 = 2 - 2\cos(\mathbf{x}, \mathbf{y}).$$

The cosine of the angle between the two vectors is obtained by dot product in the cartesian coordinates

$$\cos(\mathbf{x}, \mathbf{y}) = x_1 y_1 + x_2 y_2 = \cos(\widetilde{x}_1)\cos(\widetilde{y}_1) + \sin(\widetilde{x}_1)\sin(\widetilde{y}_1) = \cos(\widetilde{x}_1 - \widetilde{y}_1).$$

In three dimensions the corresponding formula can be obtained analogously and it is

$$\|\mathbf{x}-\mathbf{y}\|^2 = 2 - 2\cos(\widetilde{x}_1)\cos(\widetilde{y}_1) - 2\sin(\widetilde{x}_1)\sin(\widetilde{y}_1)\cos(\widetilde{x}_2 - \widetilde{y}_2). \qquad (2.5)$$

Still further, in four dimensions the distance can be calculated according to (Mackay, 1980)

$$\begin{aligned}&\|\mathbf{x}-\mathbf{y}\|^2 = \\ &\quad 2 - 2\cos(\widetilde{x}_1)\cos(\widetilde{y}_1) - \\ &\quad 2\sin(\widetilde{x}_1)\sin(\widetilde{y}_1)\left[\cos(\widetilde{x}_2)\cos(\widetilde{y}_2) + \sin(\widetilde{x}_2)\sin(\widetilde{y}_2)\cos(\widetilde{x}_3 - \widetilde{y}_3)\right]. \quad (2.6)\end{aligned}$$

The general formula is

$$\|\mathbf{x}-\mathbf{y}\|^2 = 2 - 2\cos(\mathbf{x},\mathbf{y}) = 2 - 2\sum_{i=1}^{n} x_i y_i, \qquad (2.7)$$

and by substituting the system of equations (2.4) we obtain the general expression in hyperspherical coordinates. Equations (2.5) and (2.6) are then obtained by simplifying the expression.

2.5 Generating Random Initial Solutions

In all the optimization algorithms in this work we first start with a random initial solution. We would like that any point on the surface of the sphere is equally likely to occur in the solution. To be more precise, we would like the probability that a point in the solution falls in a region on the surface of the sphere to be dependent only on the area of the region, not on its shape or orientation. That is, we would like the probability density function to be constant on the surface of the unit sphere. The density function is

$$p(\mathbf{x}) = \begin{cases} \frac{1}{A_n(1)} & \text{when } \|\mathbf{x}\| = 1 \\ 0 & \text{otherwise} \end{cases},$$

where $A_n(1)$ is the area of the surface of an n-dimensional unit sphere (2.1).

In two dimensions we could pick a random point (x_1, x_2) by taking

$$\widetilde{x}_1 = 2\pi(\mathtt{rand} - 1/2),$$

where `rand` is a uniform random number in the interval $[0, 1)$. In three dimensions we could take

$$\begin{aligned}\widetilde{x}_1 &= \arcsin(1 - 2\mathtt{rand}), \\ \widetilde{x}_2 &= 2\pi(\mathtt{rand} - 1/2).\end{aligned}$$

Unfortunately, this approach gets more and more complicated as the number of dimensions is increased. A better method is to use a suitable n-dimensional normal distribution. The density function of an n-dimensional normal distribution is of form (Therrien, 1989)

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}\left|\mathbf{K_x}\right|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m_x})^{\mathrm{T}}\mathbf{K_x}^{-1}(\mathbf{x}-\mathbf{m_x})\right),$$

where $\mathbf{K_x}$ is the covariance matrix of the distribution, $|\mathbf{K_x}|$ is its determinant, and the expected value of the distribution is denoted by $\mathbf{m_x}$. Now if we take the components to be independent, zero mean and set the variance of each of them to 1, the density becomes

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}} \exp\left(-\frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{x}\right) = \frac{1}{(2\pi)^{n/2}} \exp\left(-\frac{1}{2}\left\|\mathbf{x}\right\|^2\right). \tag{2.8}$$

The density of the distribution depends only on the length of the vector, not on its direction. If we have vectors that have this kind of n-dimensional normal distribution, we can generate points distributed uniformly on the unit sphere simply by scaling the vector lengths to 1.

Because the vector components are independent ($\mathbf{K_x} = \mathbf{I}$), we have

$$\begin{aligned}
p(x_1) &= p(x_1 \mid x_2 = a_1, x_3 = a_2, \ldots, x_n = a_{n-1}) \\
&= \frac{1}{c(2\pi)^{n/2}} \exp\left(-\frac{1}{2}x_1^2\right) \exp\left(-\frac{1}{2}(a_1^2 + a_2^2 + \ldots + a_{n-1}^2)\right) \\
&= C \cdot \exp\left(-\frac{1}{2}x_1^2\right),
\end{aligned}$$

where c and C are suitable constants so that this is a valid density function. But now x_1 has a (1-dimensional) normal distribution with zero mean and variance 1. That is, we can generate vectors that obey the density (2.8) by taking the components of the vector independently normally distributed with zero mean and variance 1, which is the standard 1-dimensional normal distribution.

Since the system in which we implemented part of the algorithms did not provide us with a normally distributed random number generator, we used the method by Box and Müller (1958) which generates two normally distributed random values (n_1, n_2) by using two uniformly distributed ones (r_1, r_2) according to

$$\begin{aligned}
n_1 &= \cos(2\pi r_1)\sqrt{-2\ln r_2}, \\
n_2 &= \sin(2\pi r_1)\sqrt{-2\ln r_2}.
\end{aligned}$$

1. Let $\mathbf{x}_0$ be the initial solution and

$$\mathbf{e}_0 = \mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_l(\mathbf{x}) \end{pmatrix}$$

   the initial error.

2. For $k = 0, 1, 2, \ldots$ until the error $\mathbf{e}_k$ is small enough:

   (a) Solve $\mathbf{d}_k$ in
   $$\mathbf{J}(\mathbf{x}_k)\mathbf{d}_k = -\mathbf{e}_k,$$
   where $\mathbf{J}(\mathbf{x})$ is the Jacobian of $\mathbf{f}$ at $\mathbf{x}$.

   (b) Let $\mathbf{x}_{k+1} := \mathbf{x}_k + \mathbf{d}_k$ and $\mathbf{e}_{k+1} := \mathbf{f}(\mathbf{x}_{k+1})$.

Figure 2.5: Newton-Raphson's method for solving a system of nonlinear equations.

2.6 Fine-Tuning the Solutions

After the optimization phase the set of shortest edges $E_1(S)$ is usually clearly visible. The final cost can be even further diminished (or the precision increased) by solving numerically the group of equations where the lengths of the edges in the set of shortest edges are set equal.

This is done conveniently in cartesian coordinates by requiring that the lengths of all the edges in the set of shortest edges are equal and that all the corresponding points lie on the surface of the unit sphere. We obtain the system of equations

$$\begin{array}{lrcl} \text{for all } \{i,j\} \in E_1(S): & (x_{i,1} - x_{j,1})^2 + \ldots + (x_{i,n} - x_{j,n})^2 - d^2 & = & 0 \\ \text{for all } i \in \bigcup\limits_{e \in E_1(S)} e: & x_{i,1}^2 + \ldots + x_{i,n}^2 - 1 & = & 0. \end{array}$$

It can be solved numerically by using Newton-Raphson's method for solving systems of nonlinear equations, see Figure 2.5. Newton-Raphson's method is based on the linearization of the equations. If the initial solution is near a solution of the system, the convergence is fast. The first partial derivatives needed in the algorithm are obtained very easily by

$$\frac{\partial((x_{i,1} - x_{j,1})^2 + \ldots (x_{i,n} - x_{j,n})^2 - d^2)}{\partial x_{i,k}} = 2x_{i,k} - 2x_{j,k},$$

$$
\begin{aligned}
\frac{\partial((x_{i,1}-x_{j,1})^2+\ldots(x_{i,n}-y_{j,n})^2-d^2)}{\partial d} &= -2d, \\
\frac{\partial(x_{i,1}^2+\ldots+x_{i,n}^2-1)}{\partial x_{i,k}} &= 2x_{i,k},
\end{aligned}
$$

and the Jacobian of the system can be calculated. There may be more equations than variables (overdetermined case) or fewer equations than variables (underdetermined case). Instead of using the usual matrix inverse in the Newton-Raphson's method, we use the pseudoinverse. It guarantees that in the overdetermined case the system is solved so that the error is minimized in the least-squares sense. In the underdetermined case the values of the variables are changed as little as possible. Note that

$$
\mathbf{A}^{\dagger} = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}} = \mathbf{A}^{\mathrm{T}}(\mathbf{A}\mathbf{A}^{\mathrm{T}})^{-1},
$$

if $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ or $\mathbf{A}\mathbf{A}^{\mathrm{T}}$ is nonsingular, that is, usually we do not have to calculate the value of the pseudoinverse, but instead it suffices to solve a linear system of equations, although this is not always numerically so stable.

The initial value for variable $x_{i,j}$ is $s_{i,j}$. Usually the initial solution is very near a local optimum and the solution to the system of the nonlinear equations is found (within the machine precision) after only about three to five iterations.

2.7 Restricting the Solution Space

It is possible to reduce the number of variables by assuming a specified structure for the code. For example, if the number of code vectors is even, we can require that for each vector $\mathbf{s}$ in the code, $-\mathbf{s}$ exists also in the code, i.e. the code vectors are located pairwise symmetrically around the origin. Essentially this reduces the number of variables to half. This method was tested in some cases with relatively large even number of code points, see Section 3.4.1 and Chapter 5.

Even larger reduction in the number of variables can be achieved by assuming larger degree of symmetry, for example icosahedral symmetry. There is a trade-off: the cases where the method is applicable become more and more rare when larger (and possibly more efficient) restrictions are applied. To date, all known spherical circle-packings with high packing density and large number of circles have a very regular structure, a couple of examples can be found in (Gáspár, 1989).

Chapter 3

Optimization Methods of This Work

There have been many different kinds of approaches to find good spherical codes, especially in the three-dimensional case. In this chapter we review some of them and present some new approaches. We concentrate on methods where the problem is viewed as an optimization problem. The methods can be classified in two groups—in the first group no *a priori* information of the minimum distance of the code is required and in the second group a goal value for the minimum distance is needed.

3.1 Characterization of the Problem

The problem of finding spherical codes with as large minimum distance as possible between points is an example of a *multimodal* optimization problem (Törn and Žilinskas, 1989). There exist many local optima and, according to (Mackay, 1980), these optima may differ only slightly in cost but in order to transform one local optima into another a complete rearrangement of the points may be needed. This is why we are interested in using *global* optimization algorithms. In this section we will show how to use two global optimization algorithms for continuous problems in this problem. We will also show how to apply simulated annealing and tabu search, which are more often used in combinatorial optimization problems. One method which can be classified as a problem-specific heuristic is presented, too.

It should be noted that in all the optimization methods it is possible, in principle, that two points coincide (at least within the computation precision). This is a contradiction with our definition of a spherical code, because the points do not form a set of k distinct points. In addition, some of the cost functions we use have infinite cost value in such situations. In practise this possibility is not a problem. The cost increases so rapidly when approaching such a configuration that it is highly unlikely that two points would coincide by chance during the optimization process.

In this work, we are interested in finding spherical codes where the minimum angle between any two code vectors is as large as possible, that is, the objective stated by the dot product of vectors is to

> find a spherical code $S = (\mathbf{s}_1, \ldots \mathbf{s}_k)$ such that the maximum dot product $\max\{\mathbf{s}_i \cdot \mathbf{s}_j : 1 \leq i < j \leq k\}$ between the code vectors is as small as possible.

Note that the dot product of any two vectors is smaller when the angle between them is larger, this is why we try to minimize the maximal dot product. The objective can be equivalently stated by

> find a spherical code $S = (\mathbf{s}_1, \ldots \mathbf{s}_k)$ such that the minimum distance $\min\{\|\mathbf{s}_i - \mathbf{s}_j\| : 1 \leq i < j \leq k\}$ between the code vectors is as large as possible.

In the following we will formulate the different optimization approaches with the help of different cost functions. The fundamental cost function that we try to minimize is

$$C(s) = -\min\{\|\mathbf{s}_i - \mathbf{s}_j\| : 1 \leq i < j \leq k\}, \tag{3.1}$$

although in most approaches we approximate the cost C by a more suitable function.

3.2 A Naive Approach

We can try to optimize directly the cost function (3.1) by, say, a simple gradient descent algorithm. However, there is a serious drawback that only the set of the shortest edges between points affects the cost, so the gradient is zero for all $s_{i,l}$ such that

$$\|\mathbf{s}_i - \mathbf{s}_j\| > \delta(S) \text{ whenever } j \neq i.$$

Since we are calculating with finite precision, the number of edges in the set of shortest edges is small. Even if we combine a line search strategy and the gradient descent algorithm (Figure 3.1), so that after each optimization step at least one new edge is added to the set of shortest edges, there are

1. Get a random initial solution $\mathbf{x}$.
2. Repeat until the gradient $\nabla f(\mathbf{x})$ is small enough.
   (a) Let the search direction be $\mathbf{p} = -\nabla f(\mathbf{x})$. Find $s > 0$ such that $f(\mathbf{x} + s\mathbf{p})$ is at its minimum (line search).

Figure 3.1: General gradient descent algorithm with line search.

usually only a few edges in that set. The convergence near a local optimum is extremely slow when the number of circles is not very small.

Furthermore, the first derivatives of the cost function (3.1) do not exist everywhere, thus many more efficient optimization algorithms, for example Newton's method (see Figure 3.2) are not applicable. It should be noted that in cost function (3.1) the first partial derivatives do not exist if the corresponding circle is part of the rigid framework and the edge length of the framework is equal to the minimum distance of the solution. Many difficulties are encountered if the derivatives cease to exist when a local optimum is approached.

3.3 A Continuously Differentiable Approximation

One of the most successful approaches in the literature (Clare and Kepert, 1986; Kottwitz, 1991a) is the *variable repulsion-force method* based on an approximation (in a very loose sense) of the cost function (3.1) by

$$C_{\mathrm{cont}}(S) = \sum_{1 \leq i < j \leq k} \left(\frac{\lambda}{\|\mathbf{s}_i - \mathbf{s}_j\|^2} \right)^m, \tag{3.2}$$

where λ is a suitable scaling factor and m is a positive integer. It is immaterial whether the denominator is $\|\mathbf{s}_i - \mathbf{s}_j\|$ or $\|\mathbf{s}_i - \mathbf{s}_j\|^2$, because we can always adjust m and λ to add or remove the exponent. The partial derivatives of (3.2) exist everywhere except if two or more points of the solution are the same. This is not a serious problem, because in the optimization process we always move to better and better solutions; it is thus in practice impossible to move to a solution where two points are the same (infinite cost). The cost function (3.2) approximates (3.1) in the sense that as m is increased, (3.2) ranks different spherical codes more and more accurately in the same order as (3.1). Note that if we set $m = 1/2$, the interaction between the points is Coulombic and globally optimal solutions are solutions to the problem of packing equal point charges on a sphere (Edmundson, 1991; Frickel and Bronk, 1987).

1. Get a random solution $\mathbf{x}$.
2. Repeat until the gradient $\nabla f(\mathbf{x})$ is small enough.
   (a) If the Hessian $\mathbf{H}(\mathbf{x}) = \nabla^2 f(\mathbf{x})$ is not positively definite, then let the search direction be $\mathbf{p} = -\nabla f(\mathbf{x})$. Find $s > 0$ such that $f(\mathbf{x} + s\mathbf{p})$ is at its minimum (gradient line search).
   (b) Otherwise, if $\mathbf{H}(\mathbf{x})$ is positively definite, solve the search direction $\mathbf{p}$ in
   $$\mathbf{H}(\mathbf{x})\mathbf{p} = -\nabla f(\mathbf{x})$$
   and do a line search in that direction or calculate the new position $\mathbf{x}'$ by $\mathbf{x}' = \mathbf{x} + \mathbf{p}$ (Newton step).

Figure 3.2: Compound gradient and Newton's method.

Clare and Kepert (1986) and later Kottwitz (1991a) used this method in the three-dimensional case. The expression for the Euclidean distance (2.5) was substituted in (3.2) and the first and second derivatives of the cost function were calculated. Kottwitz (1991a) used first a simple gradient search which proceeded until the gradient components were moderately small. Then a compound procedure using gradient and Newton's method was used, something like in Figure 3.2. The convergence of Newton's method is quadratic near the optimum. If the cost function is quadratic, the optimum is found (with precise arithmetic) after one iteration. Since this optimization algorithm is only a local optimization method, repeated application of the process with different random initial solutions are used. This method, *multistart* local optimization is one of the simplest global optimization algorithms (Törn and Žilinskas, 1989). As the number of runs approaches infinity, the probability of finding a global optimum approaches one. In a problem like this, where finding even one local optimum is time-consuming, this method may give better solutions in the CPU-time available than more complex global optimization algorithms, because usually we have CPU-time enough for only a few dozen local optimization runs.

This method works quite well in the three-dimensional case. All the best codes previously known can be found relatively easily by repeated application of the search, when the number of code vectors is not too large (Kottwitz (1991a) searched for three-dimensional codes with up to 90 code vectors). Three of these packing were improved during this work. Among other things, it is shown in (Kottwitz, 1991a) that $\delta_3^{\max}(23)$, $\delta_3^{\max}(47)$ and $\delta_3^{\max}(59)$ are all larger than the distances corresponding to the highly symmetric arrangements with one more point, three questions that were still open in (Croft et al., 1991).

3.4 Generalization of the Approximation in n Dimensions

Since the variable repulsion-force method of (Kottwitz, 1991a) and (Clare and Kepert, 1986) works so well in three dimensions we generalize it to n dimensions. We use here the hyperspherical coordinates defined in Section 2.4. By substituting (2.7) into (3.2) we obtain

$$C^n_{\text{cont}}(S) = \sum_{1\le i<j\le k} \left(\frac{\lambda}{2 - 2\sum_{l=1}^n s_{i,l}s_{j,l}} \right)^m . \tag{3.3}$$

In this work we use the BFGS secant method to optimize this function (see Figure 3.3). To apply that method we need to calculate the first partial derivatives of the function. To simplify the notation we denote one of the code vectors by $\mathbf{z} = \mathbf{s}_i$ and write $\sum_{\mathbf{x}\in S,\mathbf{x}\neq\mathbf{z}}$ to denote the sum over all the other code vectors except $\mathbf{z} = \mathbf{s}_i$. Let $\tilde{z}_k = \tilde{s}_{i,k}$, $1 \le k \le n$ be one of the hyperspherical coordinates of that vector. We calculate the partial derivative of (3.3) with respect to $\tilde{z}_k$. It is immediate that

$$\frac{\partial C^n_{\text{cont}}(S)}{\partial \tilde{z}_k} = \sum_{\mathbf{x}\in S,\mathbf{x}\neq\mathbf{z}} \frac{\partial}{\partial \tilde{z}_k} \left(\frac{\lambda}{2 - 2\sum_{i=1}^n x_i z_i} \right)^m , \tag{3.4}$$

because only those terms of (3.3) which contain at least one of the (hyperspherical) components of $\mathbf{z}$ can be nonzero in the derivative. Routine calculation shows that

$$\frac{\partial}{\partial \tilde{z}_k} \left(\frac{\lambda}{2 \mp 2\sum_{i=1}^n x_i z_i} \right)^m = -m \left(\frac{\lambda}{2}\right)^m \left(1 \mp \sum_{i=1}^n x_i z_i\right)^{-m-1} \frac{\partial}{\partial \tilde{z}_k} \left(1 \mp \sum_{i=1}^n x_i z_i\right).$$

For purposes of Section 3.4.1 we calculate simultaneously the derivative where the minus sign under the line is changed into a plus. Note that in order to avoid numerical overflows and underflows with large values of m, the coefficient in front of the partial derivative should be calculated as

$$-m \left(\frac{\lambda}{2}\right)^m \left(1 \mp \sum_{i=1}^n x_i z_i\right)^{-m-1} = -m\frac{2}{\lambda} \left(\frac{\lambda}{2 \mp 2\sum_{i=1}^n x_i z_i} \right)^{m+1} .$$

The scaling factor λ must be adjusted during the optimization. We did this by setting λ equal to

$$\min \left\{ 2 - 2\sum_{i=1}^n x_i y_i \,\middle|\, \mathbf{x},\mathbf{y} \in S; \mathbf{x} \neq \mathbf{y} \right\}$$

whenever the cost became smaller than a small constant. To complete the calculation started in (3.4) we substitute (2.4) and obtain

$$
\begin{aligned}
&\frac{\partial}{\partial \widetilde{z}_k}\left(1 \mp \sum_{i=1}^{n} x_i z_i\right) \\
&= \mp \frac{\partial}{\partial \widetilde{z}_k}\left(\sum_{i=1}^{n-1} T_i(x) T_i(z) + \widehat{T}_{n-1}(x)\widehat{T}_{n-1}(z)\right) \\
&= \mp \frac{\partial}{\partial \widetilde{z}_k} T_k(x) T_k(z) \mp \frac{\partial}{\partial \widetilde{z}_k} \sum_{i=k+1}^{n-1} T_i(x) T_i(z) \mp \frac{\partial}{\partial \widetilde{z}_k} \widehat{T}_{n-1}(x)\widehat{T}_{n-1}(z) \\
&= \pm \cos(\widetilde{x}_k) \prod_{j=1}^{k-1} \sin(\widetilde{x}_j) \cdot \prod_{j=1}^{k} \sin(\widetilde{z}_j) \\
&\quad \mp \sum_{i=k+1}^{n-1} \left[\cos(\widetilde{x}_i)\cos(\widetilde{z}_i)\cos(\widetilde{z}_k) \prod_{j=1}^{i-1} \sin(\widetilde{x}_j) \cdot \prod_{j=1}^{k-1} \sin(\widetilde{z}_j) \cdot \prod_{j=k+1}^{i-1} \sin(\widetilde{z}_j)\right] \\
&\quad \mp \cos(\widetilde{z}_k) \prod_{j=1}^{n-1} \sin(\widetilde{x}_j) \cdot \prod_{j=1}^{k-1} \sin(\widetilde{z}_j) \cdot \prod_{j=k+1}^{n-1} \sin(\widetilde{z}_j).
\end{aligned}
$$

Again we calculate simultaneously the derivative where the minus sign is changed to a plus. That formula will be used later in Section 3.4.1.

As an optimization algorithm we use the BFGS secant method (Figure 3.3). To quote Cichocki and Unbehauen (1993), "To date, one of the most powerful and sophisticated quasi-Newton methods is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm…". BFGS has the advantage over Newton's method that the second partial derivatives (which may be difficult to derive and/or heavy to compute) are not needed, because the Hessian is estimated by iteratively updating the approximation. On the other hand, the convergence rate near an optimum may be slower than that of Newton's method. In the beginning of the optimization run some time can be saved if a simple gradient search is used until the gradient components are suitably small. However, preliminary test runs showed that in this problem the cost function and gradient evaluations are relatively costly when the dimension and number of circles are increased, so the time needed to solve the linear system of equations in the BFGS algorithm is only a fraction of the total CPU-time needed; thus we decided to use BFGS right from the beginning of the optimization runs. Note that when the approximation of the Hessian is done as in Figure 3.3 the approximation remains positively definite, thus we do no more need to check the state of the matrix at every iteration.

The convergence of BFGS is linear or superlinear near the optimum. If the cost function is quadratic, the optimum is found in as many steps as there are variables. In practice, numerical inaccuracies may affect the convergence rate.

1. Take a random initial solution $\mathbf{x}_0$ and initial estimate for the Hessian $\mathbf{B}_0$, for example $\mathbf{B}_0 = \mathbf{I}$.
2. For $k = 0, 1, \ldots$ until the gradient is small enough:
   (a) Solve the search direction $\mathbf{p}^k$ in
   $$B_k \mathbf{p}^k = -\nabla f(\mathbf{x}^k).$$
   (b) Do a line search in direction $\mathbf{p}$. Let $\mathbf{x}^{k+1}$ be the result of the search.
   (c) Let
   $$\begin{aligned} \mathbf{s}^k &= \mathbf{x}^{k+1} - \mathbf{x}^k, \\ \mathbf{y}^k &= \nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^k), \\ B_{k+1} &= B_k + \frac{\mathbf{y}^k \mathbf{y}^{k\mathrm{T}}}{\mathbf{y}^k \cdot \mathbf{s}^k} - \frac{B_k \mathbf{s}^k \mathbf{s}^{k\mathrm{T}} B_k}{\mathbf{s}^k \cdot (B_k \mathbf{s}^k)}. \end{aligned}$$

Figure 3.3: BFGS secant method

As a line search algorithm we use mixed quadratic and cubic polynomial interpolation line search. The line search could also be done by for example the golden ratio method (Scales, 1985), but interpolation is faster in this case. We tried to use the line search provided with MATLAB, but it did not work too well when m in (3.3) was large, thus we implemented a simple quadratic/cubic interpolation algorithm. It works by first finding a point in the direction of search where the cost is larger than in the current solution. Then the derivative of the cost in the search direction is calculated and the second degree polynomial corresponding to these two points and one derivative is constructed. The polynomial has a minimum between the two endpoints, and its position is calculated (quadratic interpolation). This is continued until the minimum is smaller than the startpoint of the search.

Now there are three points on the direction line, the middle one having the lowest cost. The cubic polynomial corresponding to these three points and the derivative in the first point is constructed and its minimum between the endpoints is calculated (cubic interpolation). Now we discard one of the endpoints so that the middle point of the three remaining points has smaller cost than the other two. Another cubic interpolation is performed. Note that sometimes we have to compute the derivative in the new first point, this can be done conveniently by calculating the cost after a small displacement of the point in the search direction.

Experiments show that the variable repulsion-force method with BFGS is a rather efficient method when the number of variables is not too large. The primary problem when the number of variables is increased is that the size of the matrix approximating the Hessian becomes large. For example with 200 variables the size of the Hessian is $200 \times 200 = 40000$ elements, that is why in larger problems than in this work it may be advisable to use e.g. the conjugate gradient method (Hestenes, 1980), or limited memory versions of BFGS (Coleman, 1984; Nocedal, 1992).

3.4.1 Codes With Pairwise Symmetry

In some cases when the number of code vectors is even, the code vectors of the best known spherical code are located pairwise symmetrically around the origin. We can use this feature to reduce the number of optimization variables when searching for codes with large even number of code vectors.

Now the code has $2k$ vectors and the optimization variables are $\mathbf{s}_1, \ldots, \mathbf{s}_k$ or in scalar form $s_{1,1}, s_{1,2}, \ldots, s_{1,n}, s_{2,1}, \ldots, s_{k,n}$. The code vectors are $\mathbf{s}_1, \ldots, \mathbf{s}_k, -\mathbf{s}_1, -\mathbf{s}_2, \ldots, -\mathbf{s}_k$, thus the cost (3.3) value for this code is

$$
\begin{aligned}
& \sum_{1\le i<j\le k}\left[\left(\frac{\lambda}{\|\mathbf{s}_i-\mathbf{s}_j\|^2}\right)^m+\left(\frac{\lambda}{\|\mathbf{s}_i+\mathbf{s}_j\|^2}\right)^m\right]+ \\
& \sum_{1\le i<j\le k}\left[\left(\frac{\lambda}{\|-\mathbf{s}_i-\mathbf{s}_j)\|^2}\right)^m+\left(\frac{\lambda}{\|-\mathbf{s}_i+\mathbf{s}_j\|^2}\right)^m\right]+ \\
& \sum_{1\le i\le k}\left(\frac{\lambda}{\|\mathbf{s}_i+\mathbf{s}_i\|^2}\right)^m \\
= & \sum_{1\le i<j\le k}\left[2\left(\frac{\lambda}{\|\mathbf{s}_i-\mathbf{s}_j\|^2}\right)^m+2\left(\frac{\lambda}{\|\mathbf{s}_i+\mathbf{s}_j\|^2}\right)^m\right]+\sum_{1\le i\le k}\left(\frac{\lambda}{\|\mathbf{s}_i+\mathbf{s}_i\|^2}\right)^m .
\end{aligned}
$$

But now the last sum term is a constant $k(\lambda/4)^m$, so we can leave it out of the optimization process. We have the pairwise cost function

$$
C^n_{\mathrm{cont},2}(S)=2\sum_{1\le i<j\le k}\left[\left(\frac{\lambda}{\|\mathbf{s}_i-\mathbf{s}_j\|^2}\right)^m+\left(\frac{\lambda}{\|\mathbf{s}_i+\mathbf{s}_j\|^2}\right)^m\right]. \tag{3.5}
$$

Its first partial derivatives are obtained by

$$
\frac{\partial C^n_{\mathrm{cont},2}(S)}{\partial \tilde{z}_k}=2\sum_{\mathbf{x}\in S,\mathbf{x}\neq\mathbf{z}}\left[\frac{\partial}{\partial \tilde{z}_k}\left(\frac{\lambda}{2-2\sum_{i=1}^n x_i z_i}\right)^m+\frac{\partial}{\partial \tilde{z}_k}\left(\frac{\lambda}{2+2\sum_{i=1}^n x_i z_i}\right)^m\right],
$$

where the partial derivatives on the right hand side were calculated in Section 3.4. We have used the identity for code vectors $\mathbf{x}$ and $\mathbf{y}$

$$
\|\mathbf{x}+\mathbf{y}\|^2=\|\mathbf{x}-(-\mathbf{y})\|^2=2-2\cos(\mathbf{x},-\mathbf{y})=2-2\sum_{i=1}^n x_i(-y_i)=2+2\sum_{i=1}^n x_i y_i.
$$

The scaling factor λ is now adapted by setting it to

$$\min \left\{ 2 \pm 2 \sum_{i=1}^{n} x_i y_i \,\middle|\, \mathbf{x}, \mathbf{y} \in S; \mathbf{x} \neq \mathbf{y} \right\}$$

whenever the cost becomes small.

The pairwise symmetry can be used in any dimensions. Similar methods can be used to reduce the number of variables even more. We can use (3.3) for example with reflection planes and symmetry axes so that each point together with the orientation of the planes and axes induces a number of other points in the code. The stronger the symmetry constraints are, the fewer variables are needed. There is a trade-off when imposing symmetry constraints on the optimization procedure. Even if the optimization process could be done more efficiently, there may be better codes that do not fulfill the imposed constraints.

3.5 Stochastic Gradient Algorithms

Instead of performing a local optimization run again and again (the multistart algorithm) with different initial solutions, we can try to make the optimization process itself find better local optima or, hopefully, a global optimum. There are many different global optimization algorithms and many different ways to classify them, see (Törn and Žilinskas, 1989). The problem at hand has some characteristics that exclude some of the methods. The solution space is relatively large, thus there is no hope to cover it completely. This makes grid search methods infeasible. Also there are so many local optima that trajectory or tunneling methods probably do not work as well as with many other problems. Note that even by just taking all the permutations of the code vectors we obtain $k!$ different values of variables which all represent essentially the same solution. In addition, different orientations of the same code make it difficult to enumerate all or even most of the local optima. We would like to find the local optima with good accuracy, because a global optimum may differ only very slightly from some of the local optima in cost. This (together with large solution space) makes random sampling or clustering methods difficult to apply successfully.

Because repeated local optimization worked so well, we tested in this work a global optimization method which can be regarded as an extension to gradient descent local optimization, namely, stochastic gradient search based on Langevin equations, see e.g. (Cichocki and Unbehauen, 1993). Here the next solution is calculated according to

$$\mathbf{x}_{k+1} = \mathbf{x}_k - a\nabla f(\mathbf{x}_k) + \sqrt{2aT_0/\ln(2+ak)}\,\mathbf{n}_k,$$

where $\mathbf{n}_k$ is a random sample of n-dimensional normal distribution with identity covariance matrix, a is a finite time increment and T_0 is the initial temperature.

A more general version of the algorithm updates the solution according to

$$\mathbf{x}_{k+1} = \mathbf{x}_k - a_k \nabla f(\mathbf{x}_k) + b_k \mathbf{n}_k. \tag{3.6}$$

The coefficients a_k and b_k are typically calculated according to

$$a_k = \frac{A}{k}, \quad b_k^2 = \frac{B}{k \ln \ln k}, \tag{3.7}$$

where A and B are positive constants. Sometimes the noisy or imprecise measurements of the gradient are modelled by adding a random vector to the gradient. In this work, the gradient can be calculated with good precision, so we do not use that option.

We implemented two different adaptations of the general version of the method. In the first we apply the method as such with the first value of m and use the standard BFGS later. In the second we use (3.6) with the first few values of m but do a line search in the direction obtained from the equation. Random line search methods can be seen as an extreme of the second approach, where the noise is so high that the search direction is selected completely randomly. When the amount of noise is decreased, the search approaches ordinary steepest descent algorithm with line search. In (Nurmela and Östergård, 1994) it has been found that this kind of an approach (from random line search to steepest descent line search) works quite well when packing circles in a square, although the amount of noise is decreased faster than in the equations above.

3.6 Trying to Remove the Overlaps

As mentioned in the beginning of Section 3 it is possible to take a minimum distance value d_t and conjecture that there exists a code with equal or greater minimum distance. In (Mackay, 1980) this approach was used in search for four-dimensional spherical codes, which is equivalent to packing spheres on the surface of four-dimensional hypersphere. This method is more clearly a problem-specific heuristic than the optimization methods mentioned in the previous sections.

Let the target minimum distance be d_t. Now we can define local cost functions

$$c_i(\widetilde{S}) = \sum_{j=1,\ldots,i-1,i+1,\ldots,k} \max(0, d_t - \|\mathbf{s}_i - \mathbf{s}_j\|), \tag{3.8}$$

where $1 \leq i \leq k$ and the transformation from hyperspherical coordinates to cartesian coordinates is done according to (2.4). The local cost corresponding to a code vector is nonzero whenever the vector is too near another vector (the spheres are overlapping).

The method in (Mackay, 1980) takes first those vectors whose local cost is greater than zero. A vector valued function is formed where the components of the value are the nonzero local costs (3.8) of the solution, that is

$$c(\widetilde{S}) = \begin{pmatrix} c_{k_1} \\ c_{k_2} \\ \vdots \\ c_{k_l} \end{pmatrix}, \tag{3.9}$$

where $k_1, k_2, \ldots, k_l$ are exactly the indices of those vectors that have nonzero local cost. The Jacobian $\mathbf{J}$ of this function is calculated (in hyperspherical coordinates $\widetilde{s}_{k_1,1}$, $\widetilde{s}_{k_1,2}, \ldots$, $\widetilde{s}_{k_1,n-1}$, $\widetilde{s}_{k_2,1}, \ldots$, $\widetilde{s}_{k_l,n-1}$). Now we can make a linear approximation of the function at the current solution and find out where the approximation is closest to zero (in the least squares sense) by solving the equation

$$-\mathbf{J}\mathbf{x} = c(\widetilde{S}).$$

However, the Jacobian is only occasionally a square matrix, so the value of $\mathbf{x}$ is calculated using the pseudoinverse of $\mathbf{J}$ by

$$\mathbf{x} = -\mathbf{J}^{\dagger} c(\widetilde{S}).$$

If the system of linear equations determined by $\mathbf{J}$ is underdetermined, the pseudoinverse gives the least $\mathbf{x}$ (according to the Euclidean norm) satisfying the equation. On the other hand, if the system is overdetermined, the pseudoinverse minimizes the error in the least squares sense. If by repeatedly applying this procedure a solution is found, where no spheres overlap, the target distance is increased and the process is continued until no solution satisfying the target distance is found.

This approach was not implemented because the method described in Section 3.4 gave results superior to those in (Mackay, 1980) in almost every case where the optimal code is not known.

In some cases it is possible that this algorithm starts to oscillate between two or more solutions without making almost any progress towards an optimum. For a very simple example, if we use the angular distance between the points instead of the Euclidean distance and try to find a two-dimensional code with four points such that the minimum angular distance is 100°, which of course is impossible, and we start with the solution depicted on the left in Figure 3.4, then one application of the algorithm results in the solution on the right. Another application gives again the solution on the left and so on.

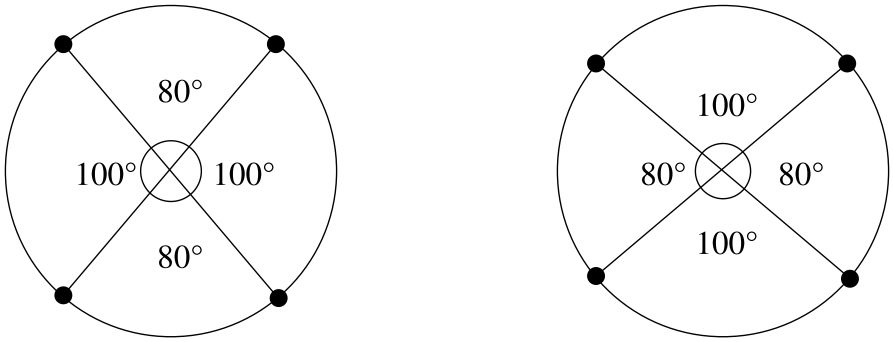

Figure 3.4: Oscillation between two solutions.

3.7 Combinatorial Optimization Methods

During the last ten years many new combinatorial optimization methods have been developed which use (pseudo)random numbers. These stochastic algorithms, including simulated annealing (Kirkpatrick, Gelatt Jr. and Vecchi, 1983), tabu search (Glover, 1989), genetic algorithms (Goldberg, 1989), evolutionary strategies (Fogel, 1988), threshold accepting (Dueck and Scheuer, 1990), record-to-record travel (Dueck, 1993) and great deluge algorithm (Dueck, 1993), are relatively simple to implement but in many cases very powerful.

These algorithms have been applied mainly to combinatorial, or discrete, optimization problems, e.g. traveling salesman problem, graph-coloring problems, code designs and so on. The combinatorial nature of such problems makes it extremely difficult to devise algorithms that can be guaranteed to give the optimal solution (even small integer problems with a few dozen variables can be totally impossible to solve with traditional methods, such as integer programming), some problems are even NP-complete, hence it is very unlikely that an efficient algorithm exists. However, stochastic optimization algorithms can be applied to get relatively good solutions in a reasonable time.

Since the dawn of these algorithms they have been applied to continuous problems, too, with varying success. In (Vanderbilt and Louie, 1984) it was noted that in general continuous function optimizing simulated annealing seems to perform well only when the number of variables is not very large.

Many of these algorithms rely on the assumption that the current solution can be perturbed only in finitely many ways (the *moves* applicable to the current solution) to obtain other, slightly different solutions, which compose the *neighborhood* of the current solution. In continuous problems some problem-specific heuristics are often needed to form a suitable neighborhood structure. Anyway, stochastic combinatorial optimization algorithms are attractive in the sense that many of them are known to converge asymptotically to the set of global optima.

In Sections 3.8 and 3.9 two local search based stochastic combinatorial optimization algorithms—simulated annealing and tabu search—are described and their adaptation to the problem of finding good spherical codes is discussed. We believe that some of the approaches described in those sections are applicable to many other continuous optimization problems as well. Furthermore, e.g. simulated annealing can be used when the derivatives of the cost function are not available, or even when the cost function is discontinuous.

It is often the case that some problem-specific heuristics are needed in order to use this kind of algorithms successfully. We assume that the required minimum distance for the code is known beforehand, and this information is used when implementing the cost function and the neighborhood structure. Usually the required minimum distance is set a little bit larger than the previously known best value for that code. We hope that this extra information will guide the search to find good optima more efficiently.

Even spherical codes have been sought with simulated annealing before, see (El Gamal, Hemachandra, Shperling and Wei, 1987). Point charge configurations were sought in (Edmundson, 1991). Comparisons to the results in (El Gamal et al., 1987) are made in Chapter 5.

In the literature there exist some modifications of combinatorial optimization algorithms that are specially tailored for continuous problems, for example *tree annealing* (Bilbro and Snyder, 1991), which, according to the authors, can be used to optimize up to 30 variables.

One nice feature of these stochastic combinatorial optimization algorithms is that if we have implemented the neighborhood structure and cost calculation for example for simulated annealing, we can immediately use the same implementation for example with record-to-record travel. In this work no other combinatorial optimization algorithms in addition to simulated annealing and tabu search were used, although some performance gain might be obtained in some problem instances by using e.g. the great deluge algorithm, cf. (Nurmela, 1993).

3.8 Simulated Annealing

In simulated annealing (Černy, 1985; Kirkpatrick et al., 1983), a random solution is chosen first. Then new solutions belonging to the neighborhood of the current solution are repeatedly tested and possibly accepted as the new current solution. The algorithm has its origins in statistical mechanics (Metropolis, Rosenbluth, Rosenbluth, Teller and Teller, 1953).

If the proposed new solution is equal to or better than the current solution, it is accepted. If the proposed new solution is worse than the current solution,

1. Get an initial solution S and an initial temperature T.
2. While *stop criterion* not satisfied do the following:
   (a) While *inner loop criterion* not satisfied do the following:
      i. Select a solution S' such that $S' \in N(S)$.
      ii. Let $\Delta = C(S') - C(S)$.
      iii. If $\Delta \leq 0$ (downhill move), set $S := S'$.
      iv. If $\Delta > 0$ (uphill move), set $S := S'$ with probability $e^{-\Delta/T}$.
   (b) Reduce temperature T.
3. Return the best solution encountered.

Figure 3.5: Simulated annealing.

it is even then accepted with probability

$$P_{\text{accept}} = \exp\left(\frac{-\Delta}{T}\right) = \exp\left(-\frac{c(s') - c(s)}{T}\right), \tag{3.10}$$

where $c(s)$ denotes the cost function assigned to the current solution, $c(s')$ denotes the cost function assigned to the proposed next solution, Δ is the cost change, and the temperature T is simply a control parameter in the same units as the cost function. Ideally, simulated annealing can 'climb' out of a poor local optimum. In the beginning the value of T is relatively large so that many cost-increasing moves are accepted in addition to cost-decreasing moves. During the optimization process the temperature is decreased gradually so that fewer and fewer cost-increasing moves are accepted. We obtain the algorithm in Figure 3.5. For a comprehensive treatment of the algorithm, see (Aarts and Korst, 1989; Dekkers and Aarts, 1991; Johnson, Aragon, McGeoch and Schevon, 1989; Johnson, Aragon, McGeoch and Schevon, 1991).

There are many different variants of the algorithm, in particular there are a great variety of different cooling schedules. In (Nurmela and Östergård, 1993) a simple implementation of the algorithm described for example in (Johnson et al., 1989) was used in search for covering designs. In that implementation L new proposed solutions are tested in each temperature and the temperature is reduced according to $T^{(i+1)} = rT^{(i)}$, where r is typically in the range of $0.99 \ldots 0.9999$ and L is a constant whose order of magnitude is about the same as that of the size of the neighborhood.

The iteration is stopped when the process becomes 'frozen'. There are many different criteria for detecting this, one of the simplest is to monitor the percentage of accepted cost-increasing moves. When this percentage goes sufficiently low, the process is terminated.

In the following we will present two modifications of the algorithm that are useful in the problem in this work. The first makes it easier to specify the annealing parameters and the second makes the annealing more adaptive. These modifications cannot be used simultaneously. Both were tested during this work.

Time-Driven Cooling The problem with simulated annealing is that the basic algorithm has so many parameters. Thus the tuning of the algorithm is very difficult and time consuming. That is why we make here a couple of approximations in order to finally have only one parameter to tune.

In the end of the optimization run the temperature should be so low that practically no cost increasing moves are accepted. That temperature value can be obtained by running the basic algorithm and recording the last temperature where a cost increasing move was accepted; denote this temperature by T_f.

The cooling schedule where the next temperature T_{i+1} is obtained from the current one (T_i) according to

$$T_{i+1} = rT_i \tag{3.11}$$

can be made continuous in time by setting

$$T = r_c^t T_0,$$

where r_c is the continuous cooling factor corresponding to r in (3.11), t is time and T_0 is the initial temperature.

Now we require that the, say, last 5% of the time available for the optimization run is used to ensure that the run ends in a local optimum. We get the equation

$$T_f = r_c^{t_A \cdot 0.95} T_0,$$

where t_A is the total time available for the run. It is trivial to solve the continuous cooling factor r_c from the equation. We get

$$r_c = \exp\left(\frac{\ln\left(\frac{T_f}{T_0}\right)}{t_A \cdot 0.95}\right). \tag{3.12}$$

If we let t be the amount of CPU-time used by the program, the *only* parameter to be tuned, given the CPU-time available and the required final temperature, is the initial temperature.

There is one drawback in this strategy. In the beginning of the optimization run many proposed moves are accepted, while in the end of the run almost no moves are accepted. If accepting a move is a time consuming operation like in the problem in this work, it follows that in the beginning of the optimization run the temperature is lowered more rapidly than according to the cooling schedule in (3.11), because each accepted move consumes more CPU-time than a rejected one. The test runs show, however, that the difference in this problem is quite small and we believe that it does not deteriorate the performance of the algorithm significantly, if at all.

Cost-Driven Cooling There are many different *adaptive annealing schedules* used in the literature. These schedules try to identify the 'critical temperatures' and use more time in them. We present here an approach that can be used when the desired final cost is known beforehand.

The idea behind this new adaptive cooling schedule is that the cost should decrease rather slowly and smoothly during the optimization run. If the cost is decreased rapidly, the risk of trapping in a poor local minimum is increased.

The idea of 'crucial temperature regime' (Vanderbilt and Louie, 1984) is that in some temperatures the algorithm chooses between local minima, that is, it is unlikely that the algorithm can 'climb out' over a cost barrier surrounding the chosen region of solution space after the temperature has been lowered below a crucial temperature. This kind of selection of regions surrounded by cost barriers can be seen in the time-cost plot of the optimization process as a sudden drop in the cost. It has been suggested from the very beginning of the development of the simulated annealing algorithm (Kirkpatrick et al., 1983) that those crucial temperatures should be recognized and the temperature should be lowered more slowly near such a temperature. Kirkpatrick et al. (1983) used *specific heat* defined by

$$C(T) = \frac{1}{T^2}\left[\langle E^2\rangle - \langle E\rangle^2\right],$$

where T is the temperature, E is the energy (cost) of the current solution and the brackets indicate an average over steps of the walk. A peak in the specific heat indicates that the freezing has begun and slow annealing is required.

Another heuristic that we present here can be used when the desired final cost is known beforehand. If the sudden drops in the time-cost plot indicate a crucial temperature regime, those drops should be smoothed out in a good cooling schedule. Given a desired initial and final costs (c_i and c_f, respectively) and an initial guess $T^{(0)}$ for a suitable initial temperature, the cooling can be done adaptively for example by

$$T^{(i+1)} = \exp\left(\ln\left(T^{(i)}\right) + \mu\left(c_d(t) - \left[\overline{c}^{(i)} - c_f\right]^{1/q}\right)\right), \tag{3.13}$$

where $T^{(i+1)}$ is the next temperature, μ is the adaptation factor, $\overline{c}^{(i)}$ is the average cost of the accepted solutions in the current temperature and $c_d(t)$ is the desired cost at time t defined by

$$c_d(t) = \left[c_i + (c_f - c_i)\frac{t}{t_{\max}} - c_f \right]^{1/q}, \tag{3.14}$$

where $t_{\max}$ is the time available for the optimization run. The parameter q controls the shape of the desired cost curve. Larger values try to lower the cost faster in the beginning of the optimization run and slower in the end.

We claim that without any *a priori* knowledge of the solution space it is a very reasonable to strive for this kind of smooth decrease in cost.

The test runs indicate that a relatively high value of μ in (3.13) can be used, so the guess of a suitable initial temperature needs not to be very accurate, the temperature adapts itself quickly to a suitable level.

When using the cost function (3.15), the desired final cost is always zero. Given the amount of time that is available, only the desired initial cost is left as a parameter to be tuned. The initial cost could be given directly to the algorithm, but instead we decided to construct a few random initial solutions first and calculate the average value of their costs. Then the algorithm is given this average multiplied by a suitable constant $0 < F_{c_i} < 1$. Values in the range $0.3 \ldots 0.8$ were used in the actual optimization runs.

3.8.1 Cost Function and Implementation of the Neighborhood

The cost function selection is crucial in local search algorithms. The cost function together with the neighborhood structure determines the structure of the solution space. As mentioned in Section 3.7, we have a target minimum distance d_t which we try to obtain by simulated annealing. Maybe the most natural cost function is

$$C_{\mathrm{SA}}(S) = \sum_{1 \le i < j \le k} \max(0, d_t - \|\mathbf{s}_i - \mathbf{s}_j\|)^p, \tag{3.15}$$

where $p \geq 1$. When $p = 1$, (3.15) is the same as the sum of the vector components in (3.9) divided by two. In three dimensions this cost function just sums the overlaps between the circles raised to a suitable exponent. It is clear that the cost is zero if and only if all the edges in the configuration are greater than or equal to the target minimum distance. Larger values of p penalize more heavily large overlaps. We used value $p = 2$ in most experiments; it works better than $p = 1$ and no noticeable increase in performance was recorded when p was increased.

The perturbations of the current solution are implemented by moving one of the points in a random direction. The amount of the movement is randomly

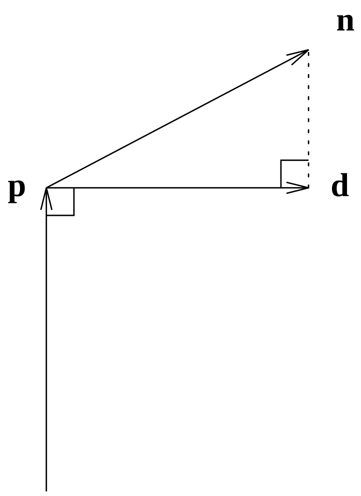

Figure 3.6: Calculation of the direction vector.

distributed and its distribution is adapted dynamically during the run, see Section 3.8.2. In the beginning the moves are generally larger than in the low temperature regime in the end of the annealing. Once the amount of movement is selected, the direction remains to be selected. We want every direction on the surface of the sphere to be equally probable and that can be achieved in the following way.

Let the point to be moved be represented by the vector $\mathbf{p}$. Now we take a randomly directed vector $\mathbf{n}$ by using an n-dimensional normal distribution as in Section 2.5. We can project it on the plane normal to $\mathbf{p}$ by setting the projection to $\mathbf{d} = \mathbf{n} - r\mathbf{p}$, see Figure 3.6. We can solve the coefficient r by noting that

$$0 = \mathbf{p} \cdot \mathbf{d} = \mathbf{p} \cdot \mathbf{n} - r\mathbf{p} \cdot \mathbf{p} = \mathbf{p} \cdot \mathbf{n} - r.$$

Now all that remains is to calculate the new position of the point when the direction on the surface of the sphere and the length of the movement are known. The vectors $\mathbf{p}$ and $\mathbf{d}/\left\|\mathbf{d}\right\|$ compose an orthonormal basis for the plane in which the moved point lies, see Figure 3.7. Given the angle γ corresponding to the amount of movement we obtain the new point $\mathbf{p}_1$ by

$$\mathbf{p}_1 = \frac{\mathbf{d}}{\left\|\mathbf{d}\right\|} \sin(\gamma) + \mathbf{p} \cos(\gamma). \tag{3.16}$$

In this simulated annealing neighborhood only one code point is moved at a time, thus only the distances between the moved point and the other points have to be recomputed. Most of the other points are so far away from the moved point that they do not affect the cost. It would be advantageous to use *volume bins* to make the cost change calculation more efficient. The idea is to divide the space around the origin into small cells and keep track of the cells where each code point belongs. By choosing the cell size suitably we could

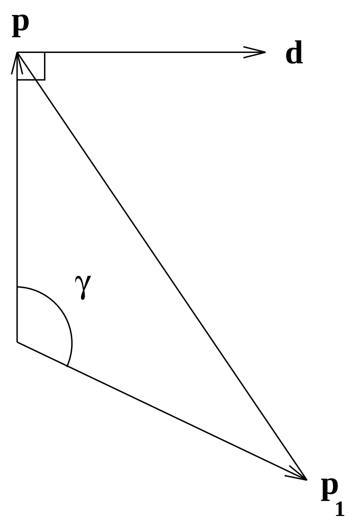

Figure 3.7: Calculation of the new point position.

reduce significantly the number of distance calculations between the points by eliminating the distance calculations between points that belong to such cells that they are necessarily too far away from each other to affect the cost. However, this was not implemented, because the efficiency increase with the number of code points in this work would probably be marginal.

3.8.2 Adaptation of The Move Distance Density

A solution is perturbed in order to achieve a new, slightly different solutions by moving a circle a small random amount in a random direction. The direction distribution is made uniform, as explained in Section 3.8.1. However, the distribution density of the distance of movement is nontrivial to select.

In many theoretical papers considering the convergence of SA (see e.g. (Dekkers and Aarts, 1991)) the probability density of moves between two solutions is constant during the optimization run. It seems very likely that the proposed moves in the beginning of the run should be larger on the average than the moves in the end of the run, because in the beginning, when the cost is high, bigger moves lead faster to lower cost levels. In the end (at low temperatures) practically no large moves are accepted, so the proposed moves should be small on the average.

We use a piecewise linear density function $f(x)$ defined by two parameters, $0 < m$ and $0 < w < 1$ according to

$$f(x) = \begin{cases} k_1 x = \frac{2}{wm^2} x & \text{when } 0 \le x \le wm \\ k_2 x + a_2 = \frac{2}{m^2(w-1)} x + \frac{2}{m(1-w)} & \text{when } wm < x \le m. \end{cases} \tag{3.17}$$

It is straightforward to verify that the function is continuous ($f(wm) = 2/m$)

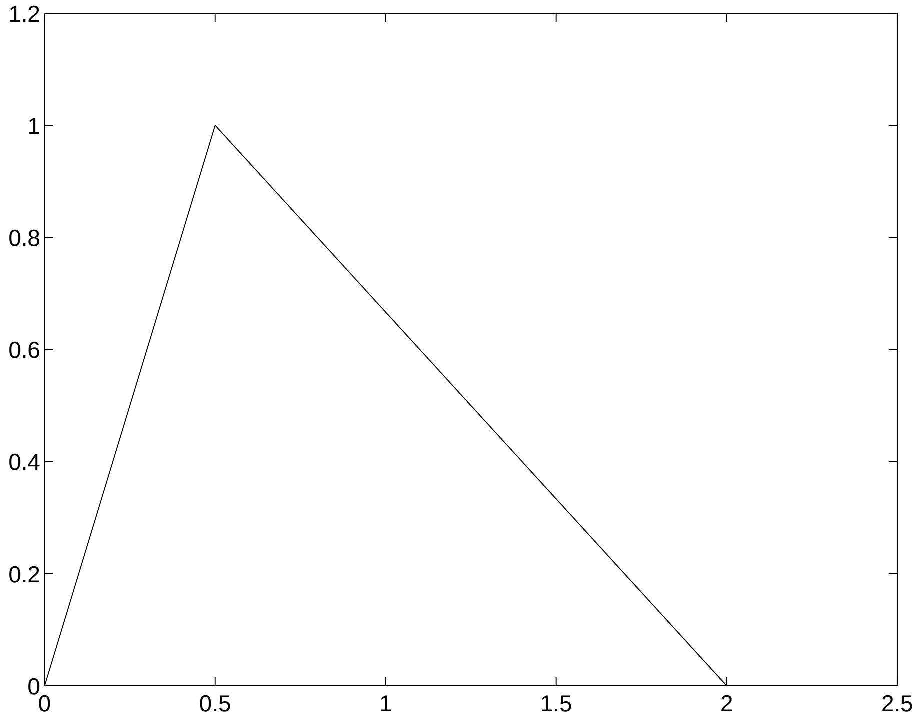

Figure 3.8: Move density function with $m = 2$ and $w = 0.25$.

and that

$$\int_0^m f(x)\,dx = 1.$$

An example of this asymmetric triangular density function (3.17) is shown in Figure 3.8. The parameter m denotes the length of the non-zero region of the function and w determines the relative position of the peak of the density. The cumulative density function is

$$F(x) = \begin{cases} \frac{1}{wm^2}x^2 & \text{when } 0 \le x \le wm \\ \frac{m^2w-2mx+x^2}{m^2(w-1)} & \text{when } wm < x \le m, \end{cases}$$

which is easily invertible in the region $0 \le x \le m$.

The crucial question that remains is how to adapt the density during the optimization run. The adaptation can be done after every L proposed moves, where L is for example 10^3. Let us assume that during L proposed moves i of them are accepted. Without any other knowledge of the solution space, cooling schedule and the order in which the i accepted moves are accepted, we may view the set

$$M = \{(d_j, c_j) \mid 1 \le j \le i\},$$

where each (d_j, c_j) means that a move was accepted with distance d_j and cost change c_j, as an estimate of the distribution from which the i samples are taken. The total change of cost is

$$\sum_{(d,c)\in M} c.$$

Now we have to calculate the new values of m and w. We would like to maximize the expected amount of cost changes, i.e. the sum of the absolute values of the cost changes. This quantity can be seen as the amount of 'action' during the L proposed moves. If we assume that the cost changes behave almost similarly during the next L iteration rounds (if the new density is near the old density) this maximization is achieved by maximizing

$$E\left[\Delta\left|C\right|\right] = \sum_{(d,c)\in M} \frac{f_{m^*,w^*}(d)}{f_{m,w}(d)} \left|c\right|, \tag{3.18}$$

where E denotes the expectation value. We denote the new values of the density parameters m and w by m^* and w^*, respectively. The maximization of (3.18) is hereby called the *principle of action-maximizing.* In this problem we have adapted only two parameters defining the shape of the move distance density. One could more generally adapt a number of parameters that define the move probability density in the solution space around the current solution by using, for example, a suitable m-dimensional normal distribution, where m is the number of variables. However, the more we have parameters to adapt, the more samples are needed before the adaptation step. To make the method computationally feasible one may have to make compromises like we do in this work by using the same move distance density for all the circles and by not adapting the directions of movements.

We claim that the principle of action-maximizing is a very reasonable heuristic to adapt the move density. In the literature there have been several attempts to control the move density by monitoring the proportion of accepted moves. In (Corana, Marchesi, Martini and Ridella, 1987) it is said that one should try to maintain 1:1 acceptance ratio during the annealing process. It is claimed to be the best possible ratio in (Vanderbilt and Louie, 1984), too, although the justification for this claim is very weak and their algorithm does not actually even use the number of accepted moves directly. Among evolutionary strategies optimization a famous rule is the '1/5 success rule' (Schwefel, 1977) which states that approximately one out of five proposed moves should be accepted. The rule is based on analytical justifications in some very simple *local* optimization problems. It seems plausible that the proportion of accepted moves should vary as the temperature is reduced. If we want the algorithm to perform almost like local optimization in the end of the annealing (low temperature) the proportion of accepted moves should be relatively small. On the other hand, in the beginning of annealing more moves should be accepted in order to make the search more diverse.

1. Take a random initial solution s.
2. Find the solution s' with smallest cost in the neighborhood of s.
3. If $c(s') < c(s)$, let $s := s'$ and go back to step 2. Otherwise stop.

Figure 3.9: Steepest descent algorithm for discrete problems

The maximization of (3.18) can be achieved by applying a simple steepest descent method by taking the gradient of (3.18). We obtain the adaptation (or learning) rules

$$\begin{aligned} m^{(i+1)} &= m^{(i)} + \frac{\mu_m}{|M|} \sum_{(d,c)\in M} \frac{|c|}{f_{m^{(i)},w^{(i)}}(d)} \frac{\partial f_{m,w}}{\partial m}(d), \\ w^{(i+1)} &= w^{(i)} + \frac{\mu_w}{|M|} \sum_{(d,c)\in M} \frac{|c|}{f_{m^{(i)},w^{(i)}}(d)} \frac{\partial f_{m,w}}{\partial w}(d), \end{aligned}$$

where the partial derivatives are

$$\begin{aligned} \frac{\partial f_{m,w}(x)}{\partial m} &= \begin{cases} \frac{-4}{m^3 w} x & \text{when } 0 \le x \le wm \\ \frac{2(m-2x)}{m^3(w-1)} & \text{when } wm < x \le m \end{cases}, \\ \frac{\partial f_{m,w}(x)}{\partial w} &= \begin{cases} \frac{-2}{m^2 w^2} x & \text{when } 0 \le x \le wm \\ \frac{2(m-x)}{m^2(w-1)^2} & \text{when } wm < x \le m. \end{cases} \end{aligned}$$

The values of the learning factors μ_a and μ_b are not difficult to tune, because the optimal density function varies only slowly in time. In the optimization runs we have $\mu_w > \mu_m$ so that the distribution primaly adapts itself by moving its peak and the non-zero region of the distribution follows slowlier afterwards. In the implementation the value of w is restricted to $0.05 \le w \le 0.95$. Note that the values of μ_a and μ_b are scaled by dividing them by the number of accepted moves, because otherwise the adaptation becomes slower in the end of the optimization run when the proportion of accepted moves becomes smaller.

3.9 Tabu Search

Tabu search (Glover, 1989; Glover, 1990) is a combinatorial optimization algorithm that can be viewed as an extension to the steepest descent algorithm (Figure 3.9). Tabu search has been applied almost exclusively to discrete

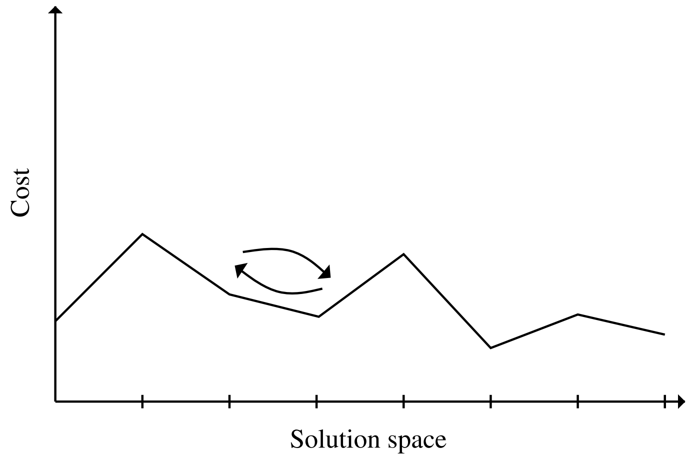

Figure 3.10: Looping search without a tabu list (in a 1-dimensional solution space).

1. Take a random initial solution s.
2. While *end condition* not true do the following:
   (a) Find a solution $s' \in N(s)$ such that

$$c(s') = \min\left\{ c(x) \,\middle|\, \begin{array}{l} x \in N(s) \text{ and } x \text{ does not violate the } \textit{tabu} \\ \textit{condition} \end{array} \right\}$$

Figure 3.11: Tabu search.

optimization problems, but we will see that tabu search can be applied to the problem in this work, too.

In the beginning, tabu search works exactly in the same way as the steepest descent search for discrete problems. When there is no solution s' in the neighborhood of the current solution s such that the cost $c(s') < c(s)$, the steepest descent algorithm stops. Tabu search instead continues to move to the best neighbor in the neighborhood of s. This induces a possibility of looping (see Figure 3.10), because even non-cost-decreasing moves are allowed. Tabu search prevents this kind of looping by introducing one or more *tabu conditions*. One widely used condition is to use a tabu list which keeps track of the inverses of the last l moves and prevents moves that are contained in the list. The general tabu search algorithm is shown in Figure 3.11.

Tabu search is by nature a combinatorial optimization algorithm and it depends heavily on the finiteness of the neighborhood of each solution. That is why applying tabu search to a continuous problem like the problem in this work is not as straightforward as with simulated annealing. The neighborhoods can be made finite by restricting the allowed moves in some ‘sensible’ way. The implementations of two different neighborhoods are described in Section 3.9.1.

In order to explain different tabu conditions precisely, we need to formalize the notion of moves and attributes associated with applications of them. Given a solution s, we use $\mathcal{M}(s)$ to denote a set of functions (the moves) that can be applied to solution s. If a move $m \in \mathcal{M}(s)$ is used, the result is obtained by $m(s)$. Attributes are associated with the proposal of a move and with the acceptance of a move. We denote the set of attributes that are used when the move m is applied to solution s by $\mathcal{A}_{\text{accept}}(m, s)$. When the current solution is s and a move m is proposed, the set of attributes is $\mathcal{A}_{\text{propose}}(m, s)$. For example, when a circle is moved (or a move is proposed) in a spherical circle-packing problem, the attributes can include the index of the circle that was moved, the index of the circle that it previously overlapped, the index pair of the moved circle and the circle that overlapped it, and so on.

Sometimes an additional feature is added to the algorithm: the aspiration function. For example, it may be reasonable to accept a move if it leads to better solution than any of the previously visited, even if the move violates the tabu condition. In the problem in the work we did not find any aspiration function that would have made the search significantly more efficient.

In the next sections we shall first describe the implementation of the tabu search neighborhoods and then the implementations of different tabu conditions.

3.9.1 Tabu Search Cost Function and Neighborhoods

In adaptation of tabu search into the problem of finding good spherical codes we concentrate on finding a code with given minimum distance between the points like we did with simulated annealing. The cost function is the same as with SA, i.e. (3.15). We describe the neighborhoods in the three-dimensional case. The generalization of the first neighborhood in n dimensions is straightforward.

A packing with the desired minimum distance is achieved exactly when there is no pair of circles that overlap, that is

> Whenever two circles overlap in an intermediate solution, at least one of them *must* be moved before a solution satisfying the target minimum distance can be obtained.

This observation gives us a heuristic that can be formulated as our first tabu search neighborhood. A point configuration S' belongs to the neighborhood

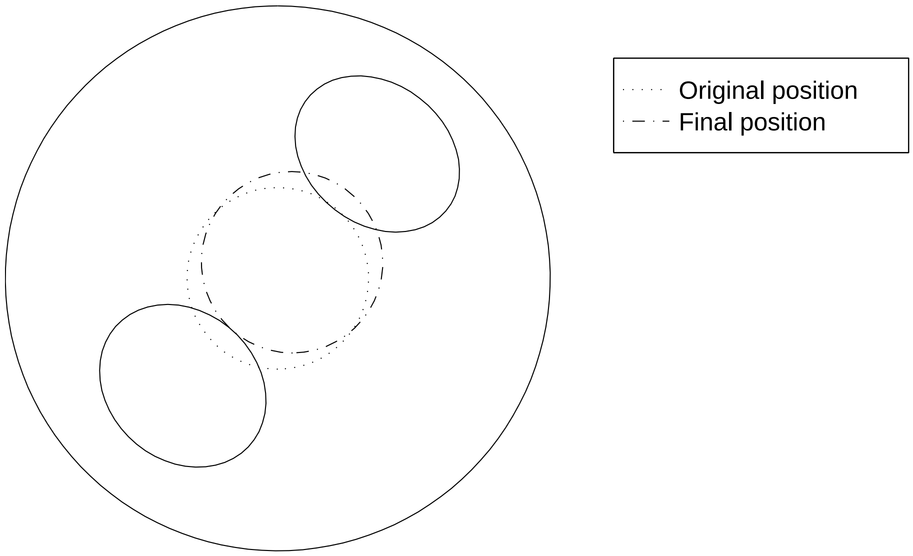

Figure 3.12: An example of a move in the first tabu search neighborhood.

of another configuration S, if there are two points $\mathbf{s}_i$ and $\mathbf{s}_j$ in S such that $\|\mathbf{s}_i - \mathbf{s}_j\| < d_t$ and S' can be obtained from S by moving one of $\mathbf{s}_i$ or $\mathbf{s}_j$ away from the other on the surface of the unit (hyper)sphere until the minimum distance d_t is achieved between them. A move in the three-dimensional case in this neighborhood is shown in Figure 3.12 (most circles removed for clarity).

The new position for a circle in this neighborhood can be calculated with the help of Figure 3.13, where $\mathbf{p}_0$ denotes the original position of the circle, $\mathbf{p}$ is another circle, which overlaps $\mathbf{p}_0$, and $\mathbf{e}$ is difference of these two vectors. The target minimum distance is d and α is the corresponding angle. Now we can project the vector $\mathbf{e}$ on the plane perpendicular to $\mathbf{p}_0$, which corresponds to calculating the vector $\mathbf{d}$ in Figure 3.6 in Section 3.8.1. The desired amount of movement is $\gamma = \alpha - \varphi$ and the new position $\mathbf{p}_1$ can be calculated according to (3.16).

In three dimensions we can go one step further. After moving the circle as described above, the circle may still overlap another circle. If the moved circle still overlaps at least one circle, we turn the moved circle around the center of the first overlapping circle in either direction until it no more overlaps the second overlapping circle. An example of this kind of a two-phase move is shown in Figure 3.14.

The geometry of the situation is depicted in Figure 3.15, where $\mathbf{p}_a$ and $\mathbf{p}_b$ are the first and second overlapping circles. The possible final positions of the moved circle are denoted by $\mathbf{p}_2$ and $\mathbf{p}_2'$.

The arc length (or angle) δ is easily calculated, because the coordinates of both $\mathbf{p}_a$ and $\mathbf{p}_b$ are known. The unknown arc length ϵ is obtained from the

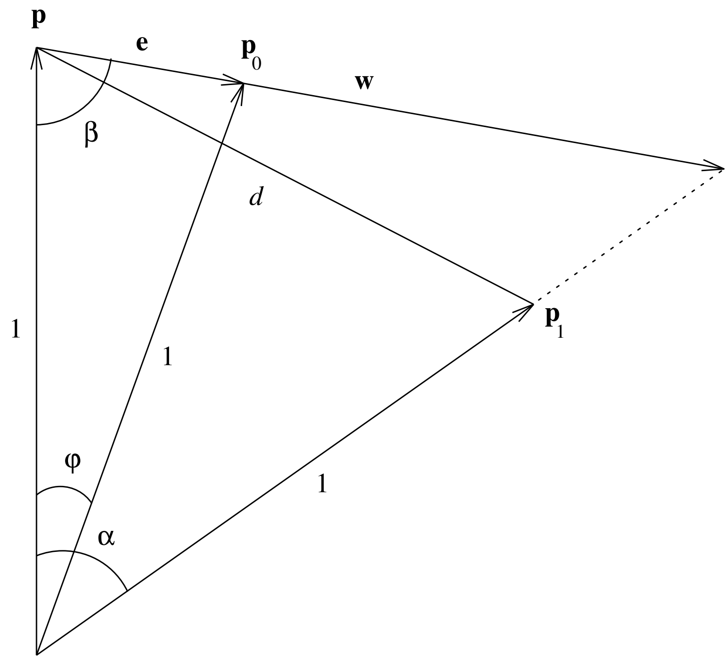

Figure 3.13: Circle movement geometry.

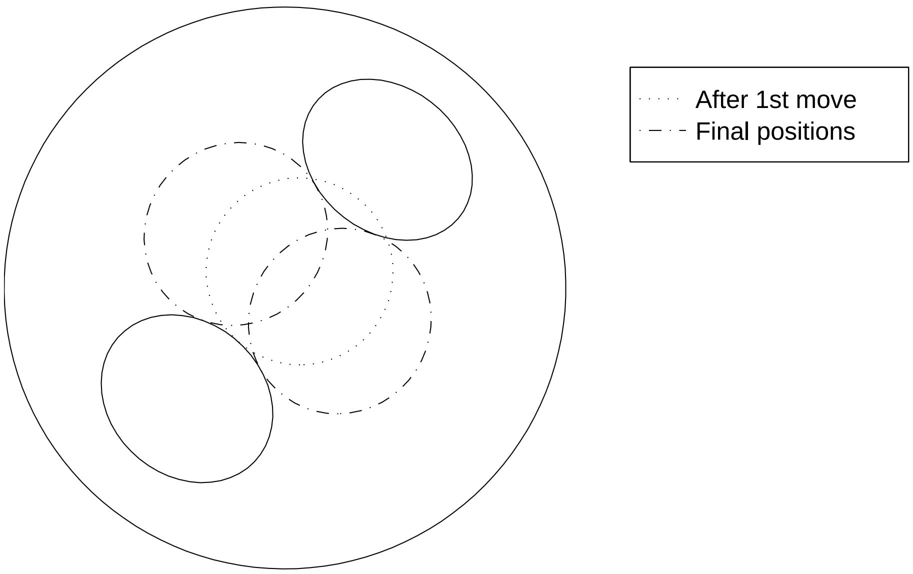

Figure 3.14: The two-phase movement of a circle.

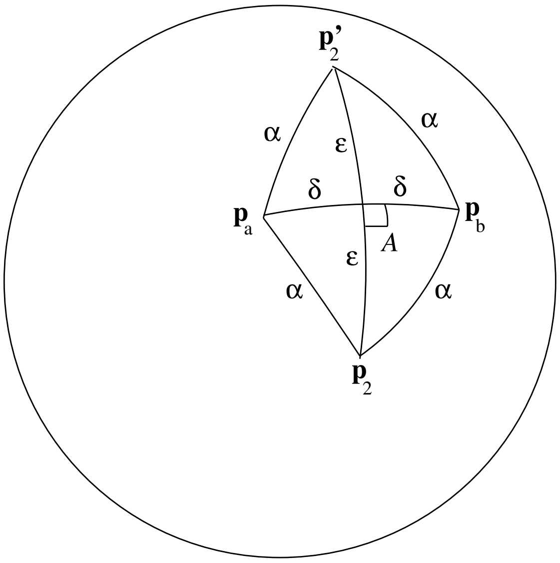

Figure 3.15: Nearest non-overlapping positions for two circles.

cosine theorem for spherical triangles, i.e.

$$\cos\alpha = \cos\delta\cos\epsilon + \sin\delta\sin\epsilon\cos A,$$

where the arc lengths denoted by Greek letters are in radians. Because of the symmetry, the angle A is equal to $\pi/2$, so ϵ is

$$\epsilon = \arccos\left(\frac{\cos\alpha}{\cos\delta}\right).$$

The actual coordinates of the positions $\mathbf{p}_2$ and $\mathbf{p}'_2$ can be calculated by first calculating a vector that is parallel to the tangent of the δ-arcs at point A and then computing the cross product of that vector and the position vector of point A. The result is a vector parallel to the tangent of ε-arcs at A. After that the situation is similar to that in Figure 3.13. We compute the new position vector $\mathbf{p}_1$ when the direction of movement $\mathbf{e}$ and the angular length of movement α are known. The other possible position is obtained by negating the direction vector.

The second tabu search neighborhood is obtained by taking the first neighborhood and adding to it these two-phase-moves. Both of the examples in Figures 3.12 and 3.14 belong to the second tabu search neighborhood.

This neighborhood can not be used in higher dimensions, because there are infinitely many positions satisfying the requirements. For example in four dimensions the possible final positions for a two-phase move lie on the periphery of a circle.

3.9.2 Recency-Based Tabu Conditions

Tabu condition is a function of the program state and the attributes of the proposed move and the current solution. Recency-based tabu conditions prevent the application of the move if the associated attributes appear on the tabu list. For example each time we apply a move m to solution s , we append the attribute set $\mathcal{A}_{\text{accept}}(m, s)$ to the tabu list. The tabu list is limited in length to include only the attributes of the last l moves, so the oldest attributes are dropped as new are added.

Now if a move m is proposed in solution s, we check if any of the attributes $\mathcal{A}_{\text{propose}}(m, s)$ are included in the current tabu list. If so, the proposed move is rejected; it is considered tabu.

We implemented two different recency-based tabu lists. In the first the attribute appended to the list when a move is accepted ($\mathcal{A}_{\text{accept}}(m, s)$) is the index of the circle that is moved. Whenever a move of a circle is proposed we check if the index of that circle appears on the list (here $\mathcal{A}_{\text{propose}}(m, s) = \mathcal{A}_{\text{accept}}(m, s)$).

Another tabu condition that was implemented adds the pair $\{i, j\}$ to the list whenever a move is made to remove the overlap between circles i and j. When a move is proposed we check if that would induce an overlap between a pair of circles that is on the list.

In the situations where there are no allowed moves because of the tabu list we simply take a move in the first tabu search neighborhood at random and apply it to the current solution. There are other possibilities to overcome this problem, one could for example take a suitable aspiration function which sometimes relieves the tabu condition. See (Reeves, 1993, Chapter 3) for such implementation options.

3.9.3 Frequency-Based Tabu Lists

The preliminary tests on the optimization problem in this work showed that the search in fact loops very rarely, this is probably due to the fact that there are very few solutions where the move can be undone by a move in the neighborhood of the new solution. While not exactly looping, the search may be stuck for a quite long time making very small moves with only a few circles, until (hopefully) a larger move is made and the solution is 'shaken'.

The recency-based tabu lists do not work as well in this situation as one might expect. They can be seen as means for preventing looping while we need a way to make the search more diverse. Frequency-based tabu lists (Reeves, 1993, Sections 3.2.8–3.2.9) try to overcome the difficulties of getting stuck around a solution by penalizing a move if the move (or the attributes of the move) have been used excessively in the near past.

We implemented a system where a counter is attached to each circle and the counter is increased whenever the circle is moved. If the counter reaches a threshold value, the cost associated with any move which moves that circle is gradually increased. If the circle is not moved for a given time, the counter is decreased. However, the preliminary tests showed that it is extremely difficult to choose the threshold value, the rate at which the cost is increased and other parameters so that the search would be efficient.

We obtained better results with a simpler alternative: we attach a counter to each circle and it is increased whenever the circle is moved. Moving a circle is forbidden, if its counter value is greater than a threshold. The counter of a circle is reset to zero, if the circle has not been moved during a certain number of moves.

Despite of the tabu lists described above, the optimization process can become rather sluggish in situations where there are no cost-decreasing moves but a few very slightly cost-increasing moves. The algorithm starts to make moves that affect the cost very little. If the cost is still far from zero, this is not what we want, because obviously a relatively large cost-increasing move should be made, thus relieving the process from a poor local optimum. A heuristic that was found useful in this respect is that we do not accept moves that affect the cost too little. To be more specific, let the costs before and after the proposed move be c_1 and c_2, respectively. The move is forbidden if

$$\left|\frac{c_1}{c_2} - 1\right|$$

is smaller than a threshold (suitable threshold values are approximately in the range $10^{-2} \ldots 10^{-5}$).

3.9.4 Using Tabu Search Neighborhoods with Simulated Annealing

In addition to the simulated annealing neighborhood described in Section 3.8.1, we can use the tabu search neighborhoods with the simulated annealing algorithm. The tuning of the tabu search parameters was found rather difficult throughout this research, and we hoped that in this approach good parameter values would be found more easily.

Tests show that these heuristics in the tabu search neighborhoods are relatively powerful in the sense the annealing works rather well over a wide range of temperatures. However, there is a drawback in these neighborhoods: it happens frequently that no cost-decreasing moves are available. In this case it may take a long time before one of the cost-increasing moves is accepted, if the temperature is low. The simulated annealing algorithm was modified so that if no move is accepted for a given time, the algorithm examines the whole neighborhood and uses this information to calculate the probabilities for each move solution to be the *next* accepted move in the same way as in (Greene and Supowit, 1986), except that we used a simpler (and less efficient) algorithm to select the move after all the cost changes in the neighborhood had been calculated. However, the gain obtained by using the efficient selection algorithm in (Greene and Supowit, 1986) would be negligible, because even the simple selection algorithm uses several orders of magnitude less time than the cost change calculations.

Because the range of temperatures where the search works well is rather wide, it is tempting to go to the other extreme and use *constant temperature annealing* (Connolly, 1990). This was tested, but the results were not better than with ordinary annealing. However, there are much fewer parameters to tune (only the temperature), hence this approach is easier to use.

When the cost is near zero, practically all the moves affect the cost only slightly and almost all proposed moves are accepted. That is why we also experimented with a cost function

$$C_{\mathrm{SA}}(S) = \ln \sum_{1 \le i < j \le k} \max\left(0, d_t - \|\mathbf{s}_i - \mathbf{s}_j\|\right)^p , \tag{3.19}$$

which is the logarithm of the simulated annealing cost function (3.15) and we found out that this function worked much better in the low cost region with both constant temperature annealing and ordinary annealing. Note that it does not matter which cost function, (3.19) or (3.15) we use with tabu search, because both rank different solutions in the same order.

3.10 Combining the Methods

When using simulated annealing or tabu search we need a good value for the target minimum distance. In three dimensions we can use the best values previously known, but in four or more dimensions no extensive tables exist. It is therefore natural to first apply the optimization method described in Section 3.4, the variable repulsion-force method, which does not need *a priori* information on the minimum distance.

We can do a few runs to find a good local optimum and then move to tabu search, for example. If we can find a configuration with a slightly larger target minimum distance by tabu search we can apply the variable repulsion-force method again to make the configuration again locally optimal.

After each application of the variable repulsion-force method the solution can be fine-tuned by solving the corresponding system of nonlinear equations in order to get a more precise minimum distance value. Before each application of tabu search we start with a new random initial solution. This strategy is illustrated in Figure 3.16.

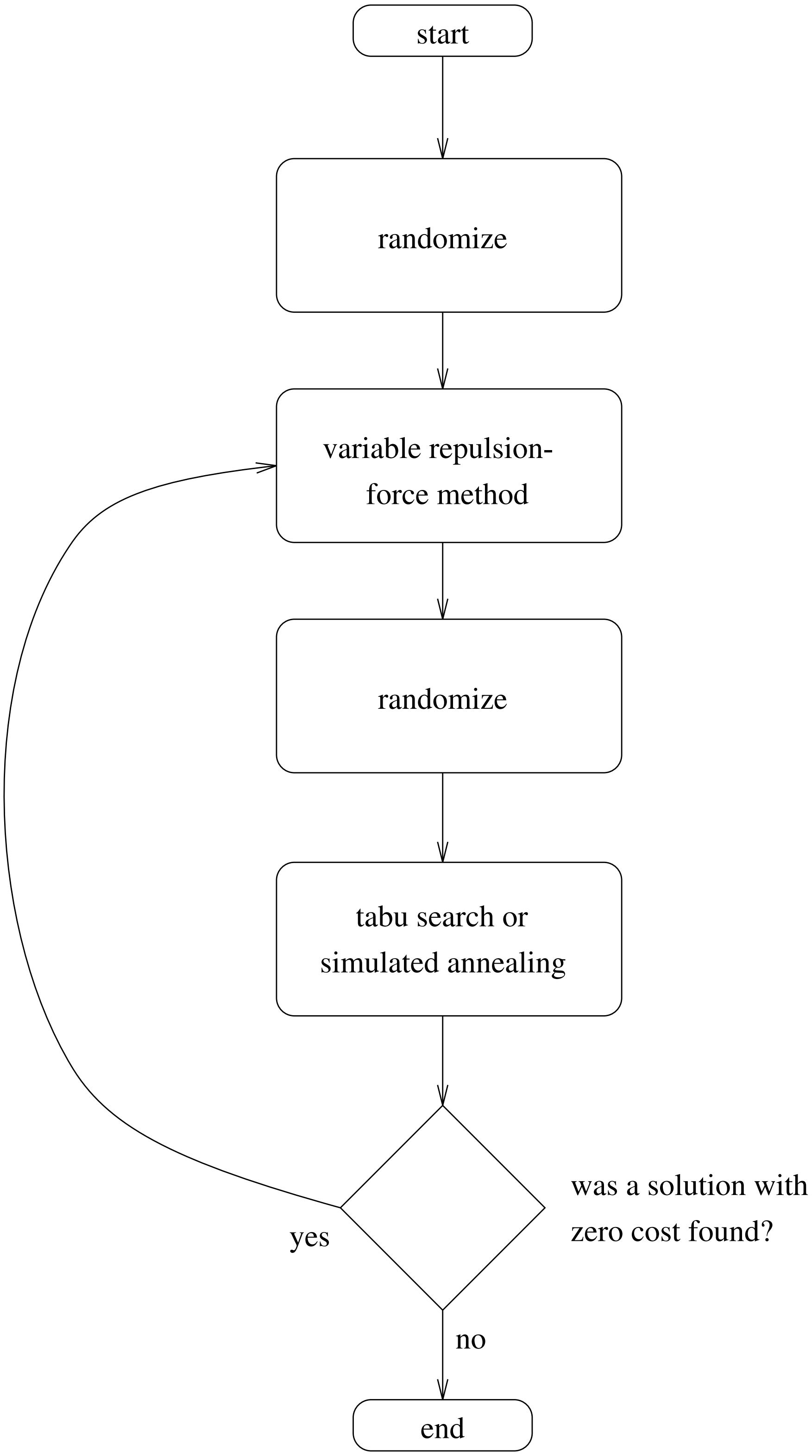

Figure 3.16: Combining the methods.

Chapter 4

Configuration Visualization

It is interesting to visualize the circle configurations resulting from the optimization runs, especially in the 3-dimensional case. It is possible to view the symmetries of the configurations in higher dimensions, too, see e.g. (Mackay, 1980), but the plots tend to be more and more obscure when the number of dimensions increases. In this chapter we present some of the methods found in the literature.

To view different symmetries of a packing, it is advantageous to rotate the packing in different orientations. Kottwitz (1991a) gives a sketch of an algorithm to find a 'standard orientation' based on the symmetries found in the packing. During the research related to this work software was developed which allows the user to view three-dimensional packings in any angle using the visualization methods shown in this section. In all the example figures in this chapter the configuration is a snub cube, the optimal packing of 24 circles.

4.1 Visualization Methods

The most familiar way of viewing a spherical circle-packing is probably the ordinary perspective plot, where the circles are drawn on the surface of a sphere. This method was used in (Clare and Kepert, 1986). An example is shown in Figure 4.1. This visualization method is nice when the densities of two or more packings are compared visually.

Another perspective projection can be obtained by constructing the polyhedron corresponding to the points. The construction method is described in Section 4.2 and an example projection is plotted in Figure 4.2. This visualization method was used in (Edmundson, 1991).

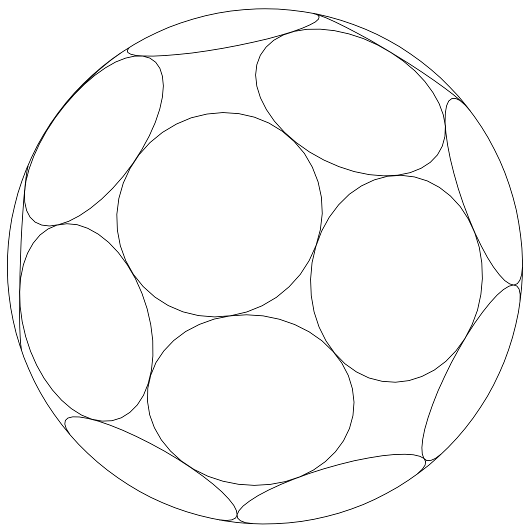

Figure 4.1: Perspective plot (circles).

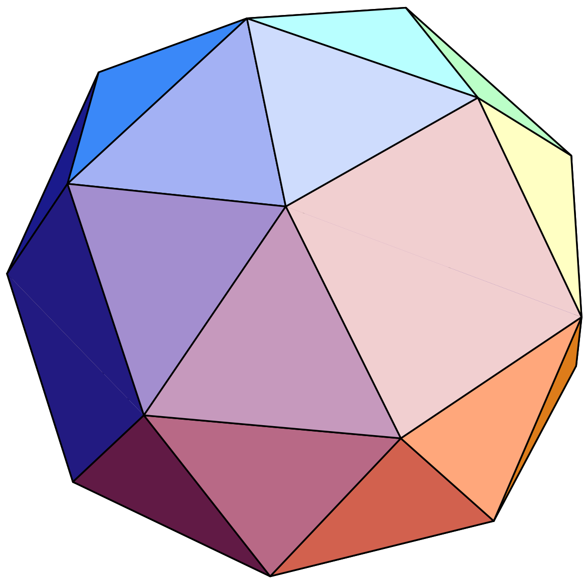

Figure 4.2: Perspective plot (polyhedron).

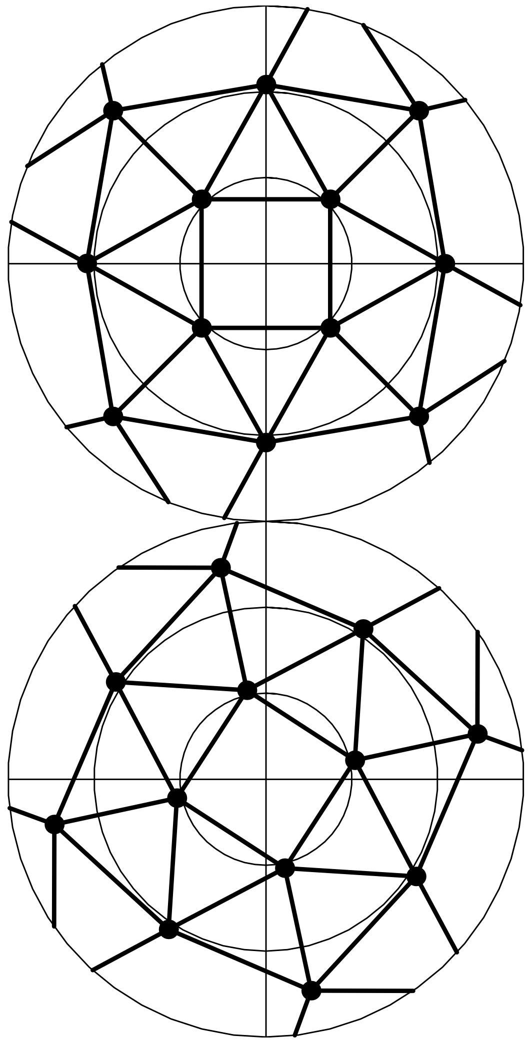

Figure 4.3: Equal area projection.

In usual perspective projections several plots are needed to show the whole configuration, unless the number of circles is so small that we can draw the invisible parts with dotted line as in Figure 2.2 on page 14. In Figure 4.3 the whole packing is shown by projecting each hemisphere in the plane. The circular grid lines correspond to colatitude values 30, 60, and 90 degrees. The set of the shortest edges is shown in solid lines while other edges are plotted in dotted line. Note that in a snub cube all the edges of the polyhedron belong to the set of shortest lines. This method is very useful when inspecting the symmetries of the configuration visually and it was used in connection with point charge packings in (Frickel and Bronk, 1987).

The polar plot method used in (Kottwitz, 1991a) does not divide the sphere into two hemispheres before projection, see Figure 4.4. In some symmetric configurations this visualization method shows the symmetry clearly.

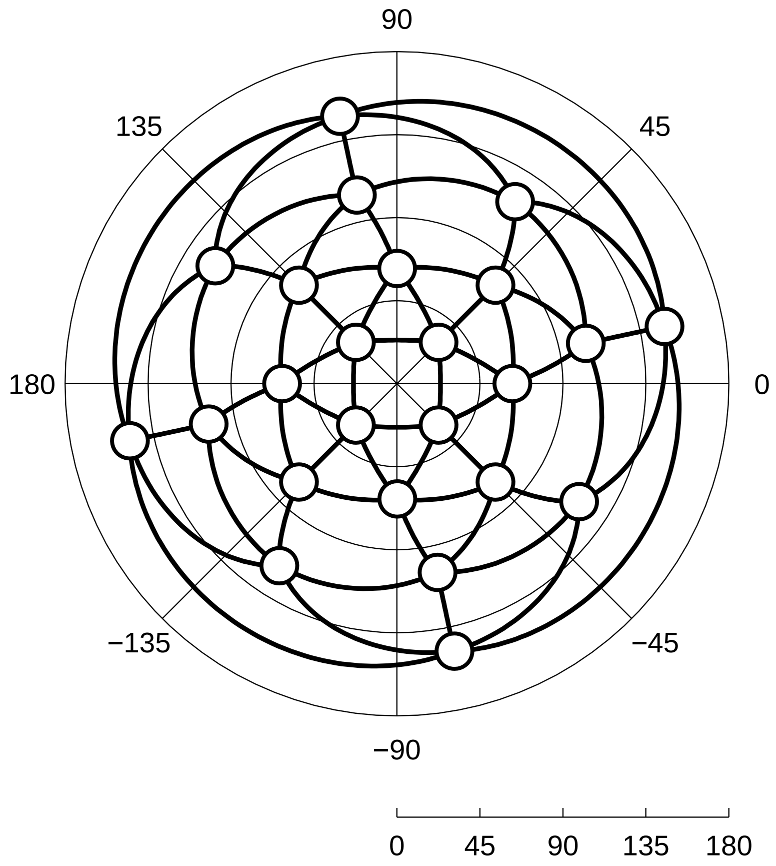

Figure 4.4: Polar plot.

1. For each triple $\{\mathbf{x}, \mathbf{y}, \mathbf{z}\}$ of points do the following:
   (a) If $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ together determine the plane which divides the space in two halves so that all the points lie in one of the halves, then find all the points that are in that plane. They form a facet of the polyhedron. Add the facet to the set of facets.

Figure 4.5: Simple polyhedron algorithm.

4.2 Construction of the Polyhedron

Given a set of points on the surface of the sphere, the corresponding polyhedron (or set of facets) can be constructed by a simple algorithm in Figure 4.5. The major drawback of this algorithm is that the number of 3-element subsets of an k element set is of order k^3 and thus the algorithm becomes rather slow when k grows. However, with the number of points in this work the algorithm works efficiently enough.

There are more efficient methods to compute the polyhedron. We could construct the convex hull of the point set represented as a set of half-spaces. A facet would then be the intersection of all the half-spaces and the plane at the boundary of one half-space. There are efficient, though elaborate algorithms for this kind of computations, see e.g. (Chazelle, 1992).

Chapter 5

Results

In this chapter we give the main results of this work. The full listings of the new codes are given in the Appendix. The rigid framework of each code can be constructed by comparing the distances between the points to the minimum distance given in Tables 5.1 through 5.1.

5.1 Properties of New Codes

In Tables 5.1 through 5.1 we give some properties of the best codes found in three to five dimensions. In the tables δ is the minimum distance of the code, N is the number of contacts between the code points, A and B are the two invariants (2.2) and (2.3), respectively, $\angle(°)$ is the minimum angle of the code (in degrees), D is the density of the code, and F is the number of free code points.

The entries in Table 5.1 are improvements over the packings given in (Kottwitz, 1991a). They were found by applying simulated annealing with the first neighborhood structure of Section 3.9.1. The packings were further improved by using the variable repulsion-force method with a high initial value of m in (3.2) and then by solving numerically the system of nonlinear equations corresponding to the contacts between the circles as described in Section 2.6. In Figures 5.1 through 5.3 we show the best previously known packings (Kot-

Table 5.1: Three dimensional codes.

k	δ	N	A	B	$\angle(°)$	D	F
43	0.55279156	82	1.36208436	0.85220185	32.090422	0.83755620	1
77	0.41584530	146	1.34879711	0.88709713	24.001284	0.84140704	3
80	0.40818020	160	1.34906054	0.88796242	23.552479	0.84191567	0

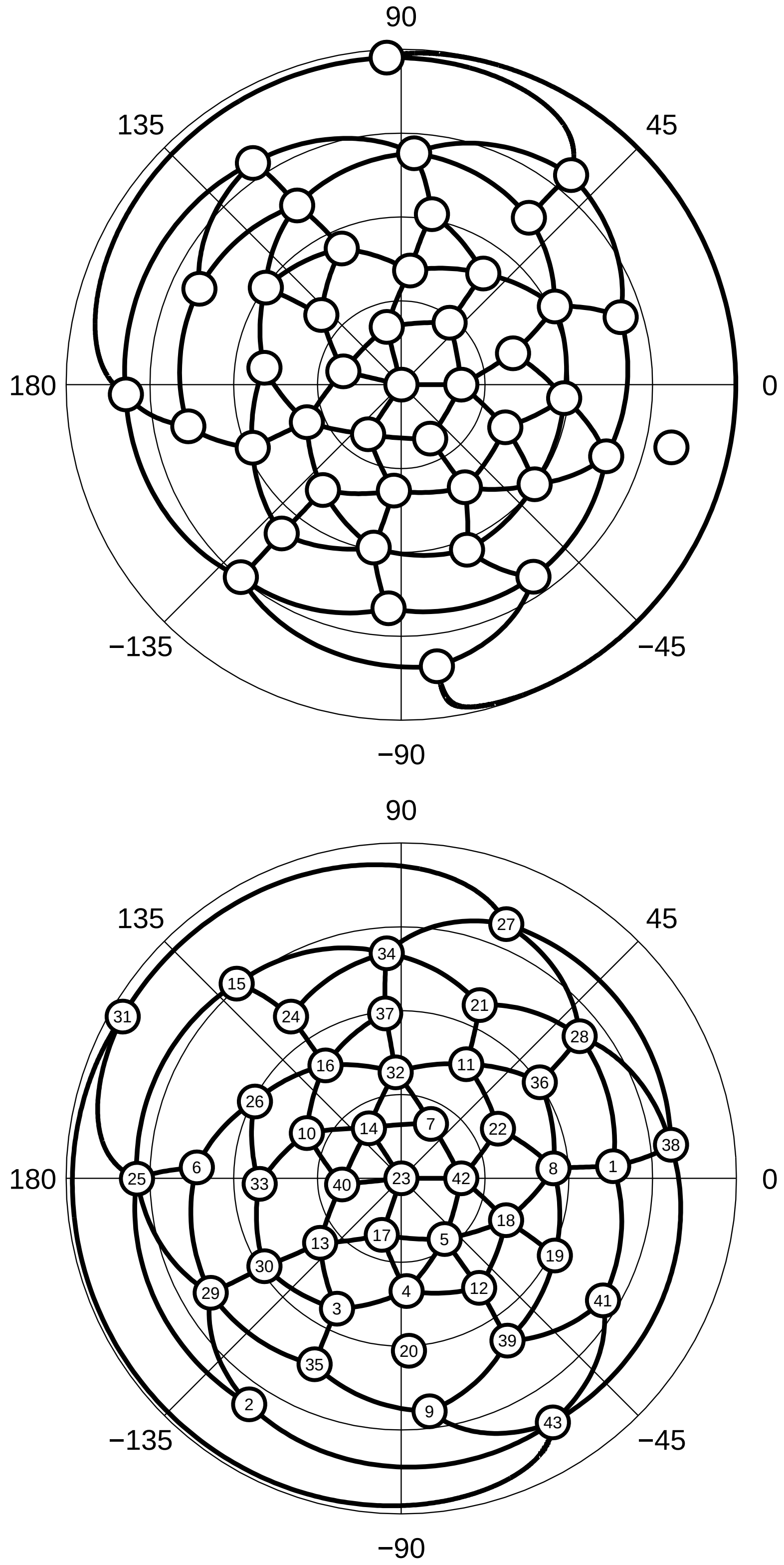

Figure 5.1: Improvement on the packing of 43 circles on the sphere.

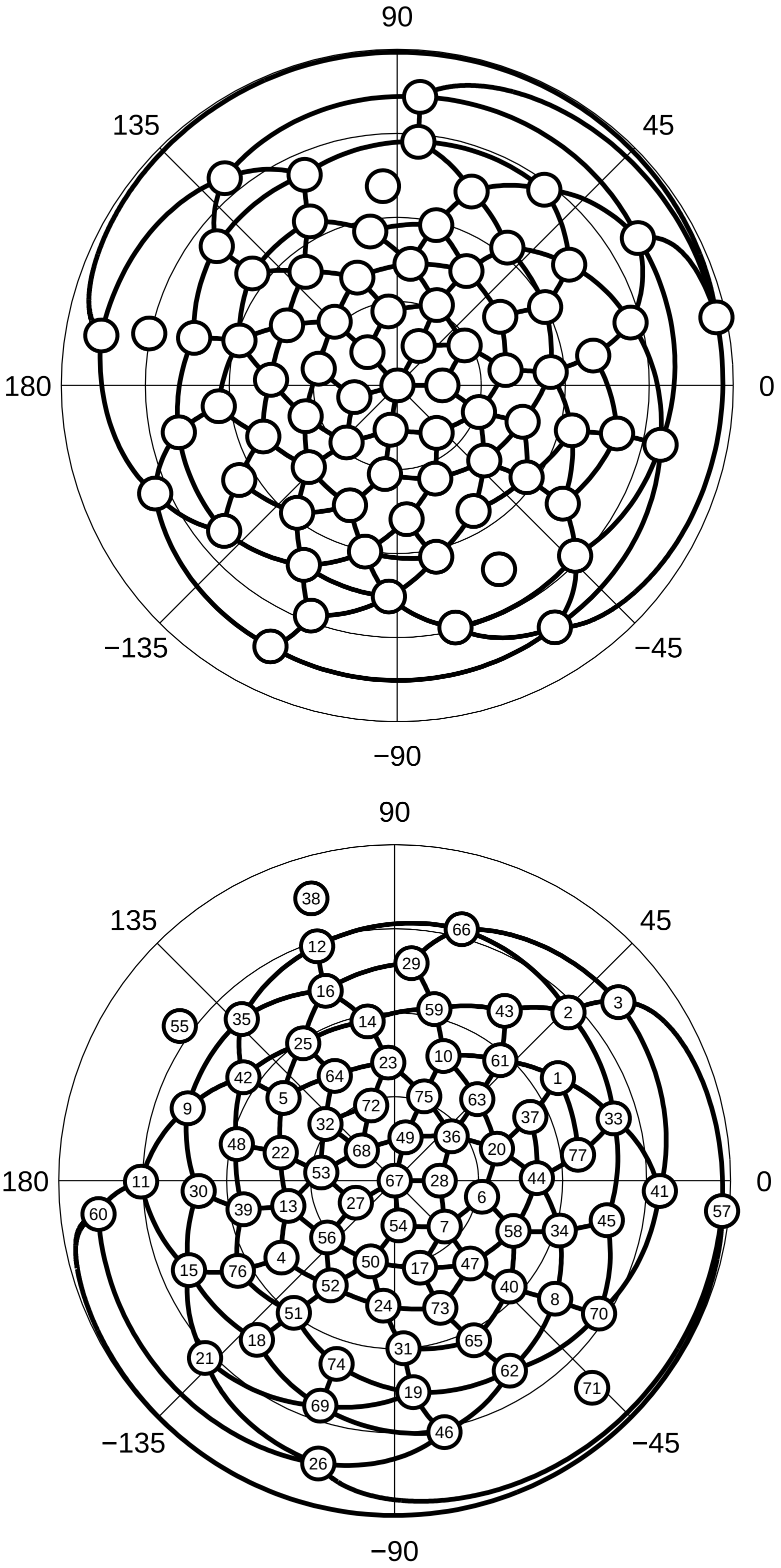

Figure 5.2: Improvement on the packing of 77 circles on the sphere.

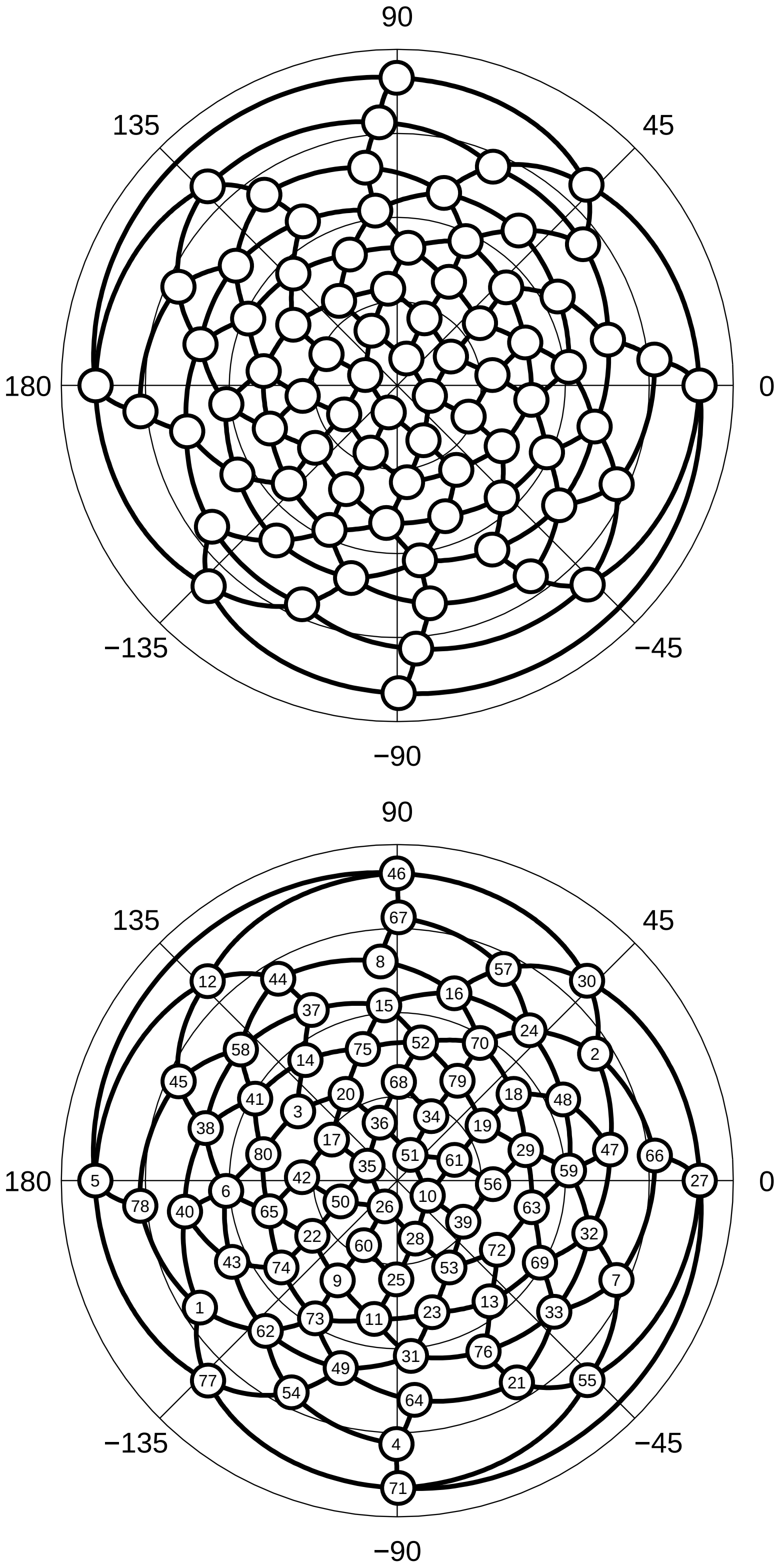

Figure 5.3: Improvement on the packing of 80 circles on the sphere.

twitz, 1991a)[1] and the new packings. The numbers in the plots of the new packings refer to the lists of coordinates in the Appendix. In the case of 43 points there is some resemblance between the two plots in the middle of them, but as whole the configurations are clearly different. Both have very irregular (or chaotic) structure and there is one free circle in both of them.

The new packing with 77 circles is quite different from the old packing and has no regular structure. Both the packings have three free circles but in the new packing two of them are located in the same 'hole' in the rigid framework.

Maybe the most interesting case is the new packing of 80 circles. The previously known best packing (Kottwitz, 1991a) is axially symmetric as depicted in Figure 5.3. We obtained an *almost* axially symmetric packing whose minimum distance is 0.40817896. After deleting and adding some edges we found a rigid framework in Figure 5.3. It is axially symmetric and has minimum distance 0.40818020.

In Table 5.1 we have the best four-dimensional codes found. Note that the minimum distance diminishes monotonically when the number of points is

Table 5.2: Four-dimensional codes.

k	δ	N	A	B	$\angle(°)$	D	F
9	1.29459245	22	1.47457404	0.69469185	80.676115	0.60343281	0
10	1.29099445	30	1.46924359	0.69897197	80.405932	0.66420920	0
11	1.24064051	33	1.45652102	0.70922186	76.679050	0.63936984	0
12	1.22474487	36	1.45167653	0.71180289	75.522488	0.66820293	0
13	1.17703385	39	1.44158336	0.72115608	72.103671	0.63484053	0
14*	1.1676	37			71.440	0.66594	0
14	1.16660311	42	1.43832335	0.72332266	71.366186	0.66398309	0
15	1.13930490	42	1.43312927	0.72755740	69.451987	0.65839405	0
16	1.10668137	48	1.42819492	0.73259082	67.193002	0.63897095	0
17	1.08419614	52	1.42140719	0.73858307	65.653156	0.63527533	0
18	1.07441201	58	1.41853079	0.74163019	64.987283	0.65325237	0
19	1.06371185	66	1.41526384	0.74480754	64.261877	0.66766582	0
20	1.06371185	72	1.41535520	0.74463427	64.261877	0.70280613	0
21	1.02822093	66	1.41125895	0.74849479	61.876032	0.66180211	0
22	1.00211364	63	1.40973229	0.75146266	60.139886	0.63864109	0
23	1.00000000	88	1.40616034	0.75484534	60.000000	0.66319218	0
24	1.00000000	96	1.40616034	0.75484534	60.000000	0.69202663	0
25	0.96196044	70	1.40346766	0.75828389	57.498883	0.63727171	0
26	0.95834272	78	1.40256477	0.75916811	57.262592	0.65489634	0
27	0.93941499	76	1.40078754	0.76153734	56.030622	0.63848634	0
28	0.93022706	79	1.39857538	0.76410210	55.435140	0.64189649	0
29	0.92104611	79	1.39788509	0.76544287	54.841729	0.64434242	1

continued on next page

[1] There is an error in (Kottwitz, 1991b) on page A-13, a minus sign is omitted in front of the first angular coordinate of point 14 of 43.

continued from previous page							
k	δ	N	A	B	$\angle(°)$	D	F
30	0.91186025	85	1.39668334	0.76662809	54.249592	0.64583887	0
31	0.90469232	83	1.39480706	0.76895501	53.788622	0.65099335	1
32	0.89919948	87	1.39464529	0.76914724	53.436013	0.65924420	2
33	0.89313593	90	1.39351300	0.77057605	53.047401	0.66554088	1
34	0.88302945	92	1.39174891	0.77223060	52.401133	0.66163897	1
35	0.86808895	100	1.39043476	0.77416673	51.449016	0.64561657	0
36	0.86124054	103	1.39029714	0.77516479	51.013861	0.64779556	0
37	0.85980379	104	1.39152930	0.77466671	50.922669	0.66231970	3
38	0.84758093	108	1.38933699	0.77690218	50.148256	0.65043253	2
39	0.84032344	105	1.38890083	0.77800282	49.689596	0.64985369	2
40	0.83603059	115	1.38724793	0.77914755	49.418696	0.65594333	0
41	0.83025877	112	1.38645666	0.78014604	49.054929	0.65796329	2
42	0.82652314	116	1.38529801	0.78165763	48.819773	0.66459782	2
43	0.82138030	111	1.38520156	0.78197329	48.496393	0.66730998	5
44	0.81036378	127	1.38472015	0.78267599	47.805052	0.65470554	0
45	0.80383071	124	1.38460838	0.78390507	47.395943	0.65292908	2
46	0.79691233	130	1.38335696	0.78473898	46.963402	0.64973594	1
47	0.79321997	136	1.38322091	0.78519922	46.732845	0.65434690	0
48	0.78761352	136	1.38264251	0.78608560	46.383151	0.65370413	1
49	0.78265430	144	1.38181044	0.78727596	46.074207	0.65436214	0
50	0.78265430	150	1.38181044	0.78727596	46.074207	0.66771647	0

increased except when moving from 19 to 20 points, from 23 to 24 points, and from 49 to 50 points. In these three cases the smaller codes are obtained by removing one point of the larger codes. In the case of 19 points two different structures with the same minimum distance may result. The alternative structure has invariant A and B values 1.41541610 and 0.74451876, respectively. The power of the symmetry constraint in Section 3.4.1 can be demonstrated with the code with 24 points, where the code of the table is found in less than two minutes when the constraint is active (this succeeded every time when we tried it), but without the constraint usually several starts with different initial solutions are needed. This difference is even more clear with 5-dimensional codes with 40 points.

All entries except those with 10, 12, 14, and 24 points are better than those in (Mackay, 1980). In these four cases we reached the same minimum distances, with the single exception of 14 points. Mackay (1980) gives a lower bound 71.440°, which was not achieved in this work despite of considerable effort. We were not able to verify that bound, so there are two entries for 14 points in Table 5.1; the entry from (Mackay, 1980) is marked with a star (*). The best code with 24 points that we reached is the same as the one generated by the D_4 lattice. At least 50 optimization runs using variable repulsion-force method were performed for the entries up to 25 points. Then the number of runs was gradually decreased so that 10 runs were accomplished with 50 points.

Table 5.1 contains the best 5-dimensional codes found. We believe that at least the codes with 3 to 10 points are optimal, but we do not know of any reference where the optimality of some of them is proved. At least 50 runs were performed for the entries up to 18 points.

Table 5.3: Five-dimensional codes.

k	δ	N	A	B	$\angle(°)$	D	F
3	1.73205081	3	1.73205081	0.57735027	120.000000	0.46875000	0
4	1.63299316	6	1.63299316	0.61237244	109.471221	0.46039928	0
5	1.58113883	10	1.58113883	0.63245553	104.477512	0.49065295	0
6	1.54919334	15	1.54919334	0.64549722	101.536959	0.53342343	0
7	1.41421356	19	1.47000275	0.68738233	90.000000	0.40640783	0
8	1.41421356	25	1.47697639	0.68491677	90.000000	0.46446609	0
9	1.41421356	32	1.47930094	0.68409492	90.000000	0.52252436	0
10	1.41421356	40	1.47930094	0.68409492	90.000000	0.58058262	0
11	1.31692555	35	1.46038350	0.69746257	82.365481	0.46326620	0
12	1.30082158	40	1.46053332	0.69873606	81.145162	0.47851579	0
13	1.27494325	46	1.45159502	0.70527933	79.207085	0.47435319	0
14	1.26491106	60	1.44339725	0.71168892	78.463041	0.49338798	0
15	1.26491106	70	1.43955884	0.71338527	78.463041	0.52862998	0
16	1.26491106	80	1.43955884	0.71338527	78.463041	0.56387198	0
17	1.20791073	64	1.43265162	0.71899299	74.307416	0.48977731	0
18	1.20374271	75	1.43066661	0.72159621	74.008083	0.51085911	0
19	1.19010722	80	1.42844025	0.72354743	73.032908	0.51324590	0
20	1.18373503	90	1.42615371	0.72500764	72.579290	0.52784777	0
21	1.17054094	87	1.42370439	0.72695286	71.644199	0.52803453	0
22	1.13449974	72	1.42090724	0.73061499	69.117345	0.48352820	2
23	1.12121454	80	1.41851769	0.73291818	68.195610	0.48062179	1
24	1.11871982	106	1.41688624	0.73406359	68.023086	0.49675960	0
25	1.11389315	111	1.41404724	0.73619367	67.689790	0.50797501	0
26	1.10004783	106	1.41015718	0.74005508	66.737307	0.50080990	2
27	1.09392161	119	1.41180521	0.73897762	66.317521	0.50783080	1
28	1.08729593	141	1.41040552	0.74062868	65.864637	0.51318149	0
29	1.08512712	138	1.40958343	0.74090104	65.716645	0.52700955	0
30	1.08348532	156	1.40816453	0.74187208	65.604697	0.54167950	0
31	1.06403772	152	1.40532478	0.74434490	64.283926	0.51826096	0
32	1.05655062	128	1.40647790	0.74377946	63.777997	0.51919041	0
33	1.03922612	120	1.40498714	0.74563889	62.612600	0.49920812	4
34	1.02728344	123	1.40285029	0.74765944	61.813418	0.48981197	1
35	1.02026861	143	1.40242519	0.74827065	61.345552	0.48984248	0
36	1.01423872	135	1.40256068	0.74837038	60.944283	0.49139714	0
37	1.00300910	139	1.40151071	0.74971818	60.199180	0.48190286	0
38	1.00092695	148	1.40045728	0.75075899	60.061336	0.49061651	0

continued on next page

continued from previous page							
k	δ	N	A	B	$\angle(°)$	D	F
39	1.00000000	228	1.39957948	0.75199226	60.000000	0.50156770	0
40	1.00000000	240	1.39957948	0.75199226	60.000000	0.51442841	0

5.2 Comparisons of the Methods

It is very difficult to compare the efficiency of the methods in this work. The basic Langevin's equations method (Section 3.5) does not work very well with cost functions (3.2) or (3.3), because when m is large even small movements can make both the cost and the gradient very large. If the values of a_k and b_k are not very small in (3.6) the optimization process is not stable.

The modified Langevin method where a line search is performed in the direction of proposed move works much better. Instead of (3.7) the amount of noise was reduced by a constant factor after each iteration, like in the basic simulated annealing algorithm in Section 3.8, because otherwise too much time is needed before the noise becomes small. No new codes were found by this method, but the best codes known were usually found with fewer runs than by using repeated local optimization by BFGS. It is hard to say whether or not the modified Langevin method is more efficient than multistart local optimization in this specific problem.

Simulated annealing with the neighborhoods described in Section 3.9.1 produced three 3-dimensional codes that are better than the best codes previously known (Kottwitz, 1991a). Usually the variable repulsion-force method was more reliable, but annealing seems to work faster when the number of points is not very large. The memory demand of simulated annealing is only a fraction of the variable repulsion-force method with BFGS and if a code does not have to be very near the optimal code, then simulated annealing seems to be more efficient. Simulated annealing with these neighborhoods can be used with more codepoints and/or in higher dimensions than the variable repulsion-force method with BFGS.

Simulated annealing with the adaptive neighborhood described in Section 3.8.1 was slower and gave worse codes than the variable repulsion-force method. However, our implementation of simulated annealing was superior to (El Gamal et al., 1987), where no adaptive move distance density was used. Schoen (1991) warns not to use simulated annealing instead of local optimization but as an extension of it.

Even though our simulated annealing approach with neighborhood described in Section 3.8.1 showed not to be the most efficient method in this work it still outperforms the results in (El Gamal et al., 1987) clearly even without adding the BFGS optimization step and solution of the system of equations

Table 5.2: Simulated annealing comparison.

minimum angle	(El Gamal et al., 1987)	Present work
0.841	18	18
arccos(13/16)	31	34
arccos(5/6)	35	38
0.524	45	48
0.45	59	63
arccos(13/14)	83	85

after the annealing. In Table 5.2 we compare some of the results of (El Gamal et al., 1987) to ours. The first column shows the minimum angle required, the second gives the largest code that El Gamal et al. (1987) could achieve, and the last column shows the largest code that was achieved in this work by simulated annealing with adaptive neighborhoods (even better results were obtained by using the neighborhoods of Section 3.9.1). Our results are in most cases superior to those of (El Gamal et al., 1987). It is hard to say how much this is due to increased computing power, but very probably our cost function and adaptive neighborhood are better than those in (El Gamal et al., 1987).

In (El Gamal et al., 1987) it was noted that the codes obtained in (Tarnai and Gáspár, 1983) are likely not to be beaten by SA. And indeed, the 3-dimensional codes with 17, 18, 27, and 32 points in (Tarnai and Gáspár, 1983) are still the best codes known. However, the codes with 34, 35, and 40 points have been improved by (Kottwitz, 1991a) (34 and 35 points) and (Clare and Kepert, 1986) (40 points). Our implementation of simulated annealing with adaptive neighborhood (Section 3.8.1) was able to improve codes in (Tarnai and Gáspár, 1983) with 34 and 35 points. With neighborhoods described in Section 3.9.1 and simulated annealing we were able to improve the code in (Tarnai and Gáspár, 1983) with 40 points, too. This fortifies the conclusion that our implementations of simulated annealing are more effective than that of (El Gamal et al., 1987).

The performance of tabu search was not as good as we had expected. Partly this may be caused by the fact that in our implementations there are so many parameters to be tuned. It is possible that we did not find the best possible parameter values, because no exhaustive testing was computationally feasible. The good performance of the neighborhoods of Section 3.9.1—which were originally developed for tabu search—when used with simulated annealing shows that the neighborhoods work well; the problem is in the tabu lists.

The construction method in (Sloane, 1981) gives codes with considerably lower density than the optimization methods. In Table 5.3 we have listed the

Table 5.3: The first 25 shells of the D_4 lattice.

shell number	number of points	minimum distance	density
1	24	1	0.6920
2	24	1	0.6920
3	96	0.5774	0.5029
4	24	1	0.6920
5	144	0.4472	0.3469
6	96	0.5774	0.5029
7	192	0.3780	0.2780
8	24	1	0.6920
9	312	0.3333	0.3091
10	144	0.4472	0.3469
11	288	0.3015	0.2108
12	96	0.5774	0.5029
13	336	0.2774	0.1913
14	192	0.3780	0.2780
15	576	0.2582	0.2643
16	24	1	0.6920
17	432	0.2425	0.1642
18	312	0.3333	0.3091
19	480	0.2294	0.1543
20	144	0.4472	0.3469
21	768	0.2182	0.2125
22	288	0.3015	0.2108
23	576	0.2085	0.1390
24	96	0.5774	0.5029
25	744	0.2000	0.1584

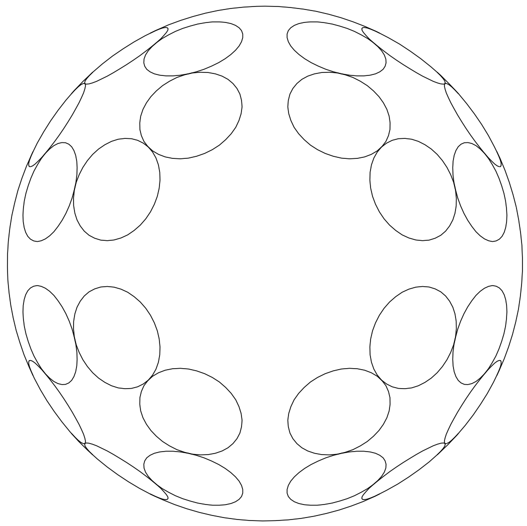

Figure 5.4: The packing generated by the 8th shell of the D_3 lattice.

numbers of points in the first 25 shells of the D_4 lattice and the densities of the corresponding spherical codes. As it is obvious, the densities even of the best nontrivial codes are lower than those in Table 5.1 (except with 24 points where D_4 gives the best code known). The same is true for all the lattices mentioned in Section 2.2. An example showing a typical circle-packing in three dimensions found by this method is shown in Figure 5.4. It is the 8th shell of the D_3 lattice, and as can be seen, the packing is relatively sparse (the density is 0.43246785) when compared to the packings in (Kottwitz, 1991a) and in Table 5.1.

This does not mean that the method is useless, it can be used to construct codes in higher dimensions and with more points than it is possible with the optimization methods of this work. The lattices that we used in this work are designed for high-density sphere packings, some less dense packings might give better spherical codes. In (Conway and Sloane, 1988) it was pointed out that in some dimensions the best known kissing number is generated by a different lattice than the best known sphere packing. This was explained by noting that the kissing number problem is a local problem while the sphere packing problem is global. The problem of finding good spherical codes resembles more the kissing number problem than the sphere packing problem.

When the number of code points and/or dimensions grow too large, the time needed by the variable repulsion-force method becomes the limiting factor. The longest runs that were performed in this work took several hours of CPU-time on a current HP-700 workstation. In (Kottwitz, 1991a) at least 50 runs were performed for each configuration, but they do not give any information about the amount of CPU-time that their implementation used. It would be interesting to know whether Newton's method is be more efficient than BFGS in the variable repulsion-force method, where the second partial derivatives of the cost function are relatively costly to compute. A code with a little more than 200 variables could be searched for by our BFGS-implementation, but to get very good results several months of CPU-time may be required.

5.3 Some Implementation Issues

The variable repulsion-force method with BFGS algorithm was implemented using MATLAB. There is a library function in MATLAB that implements BFGS, but the line search phase of that function appeared not to work properly with this problem when m became large, so we had to use an implementation of ours. The calculation of the cost value and its gradient were written in C for efficiency reasons.

One problem when implementing the variable repulsion-force method is that the scaling factor λ in (3.3) must be adjusted when the optimization proceeds, otherwise the cost goes to zero with large values of m due to numerical underflows. The rescaling was done whenever the cost went below a certain value. The estimate of the Hessian was then reset to the identity matrix.

The different versions of simulated annealing and tabu search were implemented in C++ using a class library, whose development had started during the research related to (Nurmela, 1993). The algorithms had to be adapted to handle continuous cost functions, for example the cumulation of rounding errors when adding cost changes must be compensated. In these algorithms we only move one circle at a time, so only the distances between that circle and the other circles must be recomputed. Rounding errors are a slight problem when calculating the new positions of circles; we have to allow a very small computational overlap between the circles without increasing the cost accordingly. Otherwise it happens that the circles are overlapping after a move that was supposed to remove the overlap between them.

Routines for configuration visualization in three dimensions were implemented in MATLAB. These routines allow the user to use any plot type in Chapter 4 and rotate the configurations freely.

The nonlinear equations concerning the rigid substructure were solved numerically using Mathematica. Mathematica has arbitrary precision arithmetics, so it is possible to find the minimum distances of the codes at any precision required.Neither MATLAB nor Mathematica supports exporting or importing data in the format understood by the other, so the interface between these two programs became rather complex.

Chapter 6

Conclusions

This work focuses on constructing spherical codes by different global optimization methods. In the first two chapters we review the background and applications of spherical codes. We also mention some connections between spherical codes and other classical mathematical problems. Basic results concerning spherical codes and related problems are given. Some implementation issues including representation of the points in hyperspherical coordinates and generation of uniform random initial solutions are discussed. Rigid frameworks are used to increase the accuracy of the locally optimal codes. We discuss a method that can be used to reduce the number of variables in the optimization process.

In Chapter 3 we give different optimization methods that can be used to construct spherical codes. One of the most successful methods in the literature is the variable repulsion-force method by (Clare and Kepert, 1986) and (Kottwitz, 1991a). We apply that method and make experiments by replacing the local search algorithm by two stochastic global optimization algorithms. The repulsion-force method is generalized into n dimensions. We present new heuristics and cost functions which are then used in conjunction with simulated annealing and tabu search.

In three cases we succeed in improving the codes in (Kottwitz, 1991a), which were constructed by thorough application of the variable repulsion-force method. The improvements are obtained by simulated annealing and a relatively simple heuristic. Three new 3-dimensional spherical codes with 43, 77, and 80 points are thus now known. All our different implementations of simulated annealing performed better than that in (El Gamal et al., 1987), although partly this may be caused by increased computing power.

In four and five dimensions we use the generalization of the variable repulsion-force method to search for codes with 9 to 50 and 3 to 40 points, respectively.

Most of these codes are better than the previously known ones.

In Chapter 4 we review methods that can be used to visualize 3-dimensional spherical codes. During the research we have developed software that can be used to visualize the codes in any format shown in the figures of this work.

The search results are discussed in detail in Chapter 5. Some properties of the new codes are tabulated and comparisons are made to earlier results.

When the number of code points is not too large, the methods in this work produce very high density codes in dimensions 3 to 5 compared to those in (Sloane, 1981). On the other hand, they have often much less regular structure. It seems that the variable repulsion-force method accompanied occasionally by simulated annealing is currently the best construction method for spherical codes in small dimensions if the aim is to achieve as high densities as possible. The usability of the repulsion-force method can be extented to higher number of code points by requiring that the solutions have certain symmetries.

The codes formed in this work can be used as building blocks when constructing larger codes. See (Lachaud and Stern, 1994) and their references for the *concatenation method*, which constructs larger spherical codes from smaller ones by using block codes.

The structures of the new codes presented in this work are not yet thoroughly investigated. There may be almost symmetric 4- and 5-dimensional codes which could be further improved by finding the nearby symmetric code as we did with the 3-dimensional code with 80 points. In addition, studying the methods for reducing the number of variables by imposing symmetry constraints on the solutions (see Section 3.4.1) is a promising subject for further research.

The tabu search and simulated annealing neighborhoods defined in Section 3.9.1 can be seen as means of solving a more general problem, namely a system of inequalities

$$\begin{aligned}
f_1(\mathbf{x}) &\leq 0, \\
f_2(\mathbf{x}) &\leq 0, \\
&\vdots \\
f_N(\mathbf{x}) &\leq 0,
\end{aligned}$$

where finding the solution simultaneously to *all* the inequalities is difficult. For example various finite packing problems belong to this class of problems. We assume that we have efficient heuristics to make any *single* $f_i(\mathbf{x})$ less than or equal to zero. The heuristics should be such that they change the current solution $\mathbf{x}$ as little as possible in some sense. We define the moves (and thereby the neighborhoods) as the applications of the heuristics to the positive functions. The cost function is of form

$$\ln \sum_{i=1}^{N} \left[c_i \max(0, f_i(\mathbf{x})) \right]^p , \tag{6.1}$$

where $c_1, c_2, \ldots c_N$ are scaling factors and p controls how much big differences from zero are penalized compared to smaller differences. Either simulated annealing or tabu search can be used as the search algorithm. The logarithm in (6.1) works well in the problem in this work, but it has the drawback that when a solution is found, the cost is negative infinity. Some other functions such as $(\cdot)^{1/q}$ where q is a positive constant might be worth testing, too. The functions $f_1, \ldots, f_N$ need not be continuously differentiable or even continuous.

There are some problems related to finding good spherical codes. For example *bounded energy codes*, which are configurations of points *in* the n-dimensional unit sphere, could be searched by the methods in this work. The approximative cost function (3.3) can be modified by adding one more coordinate $\widetilde{x}_0$ to each point so that the conversion from hyperspherical coordinates is defined by

$$\begin{aligned}
x_1 &= \sin(\widetilde{x}_0)\cos(\widetilde{x}_1), \\
x_2 &= \sin(\widetilde{x}_0)\sin(\widetilde{x}_1)\cos(\widetilde{x}_2), \\
x_3 &= \sin(\widetilde{x}_0)\sin(\widetilde{x}_1)\sin(\widetilde{x}_2)\cos(\widetilde{x}_3), \\
&\vdots \\
x_{n-1} &= \sin(\widetilde{x}_0)\sin(\widetilde{x}_1)\sin(\widetilde{x}_2)\cdots\sin(\widetilde{x}_{n-2})\cos(\widetilde{x}_{n-1}), \\
x_n &= \sin(\widetilde{x}_0)\sin(\widetilde{x}_1)\sin(\widetilde{x}_2)\cdots\sin(\widetilde{x}_{n-2})\sin(\widetilde{x}_{n-1}).
\end{aligned}$$

We have used $\sin(\widetilde{x}_0)$ instead of $\widetilde{x}_0$ to keep the optimization problem unconstrained. The adaptive simulated annealing neighborhood can be modified for this problem, too.

The dual problem to finding 3-dimensional spherical codes, or spherical circle-packings is the problem of finding *spherical circle-coverings*, where the n-dimensional sphere is covered by circular caps (or $(n-1)$-dimensional spheres). The optimal coverings are known in three dimensions only for $k \leq 7$ and $k = 10, 12, 14$, see (Croft et al., 1991) for references. As far as the author knows, there is no such simple continuous approximation of the cost function as (3.3), so continuous local optimization algorithms such as BFGS may be difficult to apply. However, we can use simulated annealing in the 3-dimensional case by defining the cost function as

$$C(S) = [\text{the length of the uncovered circle peripheries}]^p ,$$

where $p \geq 1$ and we have used the fact that the sphere is covered by circles exactly when all the circle peripheries are covered by other circles. The cost calculation reduces essentially to the calculation of circle intersections on the surface of the sphere. The solutions can be fine-tuned by observing the points where three or more circles intersect and by solving the corresponding system of equations.

Bibliography

Aarts, E. and Korst, J. (1989). *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*, Wiley, Chichester.

Biglieri, E. and Elia, M. (1976). Cyclic-group codes for the Gaussian channel, *IEEE Trans. Inform. Theory* **22**: 624–629.

Bilbro, G. L. and Snyder, W. E. (1991). Optimization of functions with many minima, *IEEE Trans. Systems Man Cybernet.* **21**: 840–849.

Blake, I. F. (1971). The Leech lattice as a code for the Gaussian channel, *Inform. Contr.* **19**: 66–74.

Blake, I. F. (1972). Distance properties of group codes for the Gaussian channel, *SIAM J. Appl. Math.* **23**: 312–324.

Boerdijk, A. H. (1952). Some remarks concerning close-packing of equal spheres, *Philips Res. Rep.* **7**: 303–313.

Box, G. E. P. and Müller, M. E. (1958). A note on the generation of random normal deviates, *Ann. Math. Statistics* **28**: 610–611.

Chazelle, B. (1992). An optimal algorithm for intersecting three-dimensional convex polyhedra, *SIAM J. Computing* **21**: 671–696.

Cichocki, A. and Unbehauen, R. (1993). *Neural Networks for Optimization and Signal Processing*, Wiley, Chichester.

Clare, B. W. and Kepert, D. L. (1986). The closest packing of equal circles on a sphere, *Proc. R. Soc. Lond.* **A 405**: 329–344.

Coleman, T. F. (1984). *Large Sparse Numerical Optimization*, Lecture Notes in Comput. Sci. 165, Springer-Verlag, Berlin.

Connolly, D. T. (1990). An improved annealing scheme for the QAP, *European J. Oper. Res.* **46**: 93–100.

Conway, J. H. and Sloane, N. J. A. (1988). *Sphere Packings, Lattices and Groups*, Springer-Verlag, New York.

Corana, A., Marchesi, M., Martini, C. and Ridella, S. (1987). Minimizing multimodal functions of continuous variables with the 'simulated annealing' algorithm, *ACM Trans. Math. Software* **13**: 2–280.

Croft, H. T., Falconer, K. J. and Guy, R. K. (1991). *Unsolved Problems in Geometry*, Springer-Verlag, New York.

de Buda, R. (1975). The upper error bound of a new near-optimal code, *IEEE Trans. Inform. Theory* **21**: 441–445.

Dekkers, A. and Aarts, E. (1991). Global optimization and simulated annealing, *Math. Programming* **50**: 367–393.

Dueck, G. (1993). New optimization heuristics: the great deluge algorithm and the record-to-record travel, *J. Comput. Phys.* **104**: 86–92.

Dueck, G. and Scheuer, T. (1990). Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing, *J. Comput. Phys.* **90**: 161–175.

Edmundson, J. R. (1991). The distribution of point charges on the surface of a sphere, *Acta Cryst. Sect. A* **48**: 60–69.

El Gamal, A. A., Hemachandra, L. A., Shperling, I. and Wei, V. K. (1987). Using simulated annealing to design good codes, *IEEE Trans. Inform. Theory* **33**: 116–123.

Fogel, D. B. (1988). An evolutionary approach to the traveling salesman problem, *Biol. Cybernet.* **60**: 139–144.

Frickel, R. H. and Bronk, B. V. (1987). Symmetries of configurations of charges on a sphere, *Canad. J. Chem.* **66**: 2161–2165.

Fujita, T., Tokunaga, J. and Inoue, H. (1971). *Atlas of Scanning Electron Microscopy in Medicine*, Igaku Shoin Ltd, Tokyo and Elsevier, Amsterdam.

Glover, F. (1989). Tabu search—Part I, *ORSA J. on Computing* **1**: 190–206.

Glover, F. (1990). Tabu search—Part II, *ORSA J. on Computing* **2**: 4–32.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley, Reading.

Goldberg, M. (1967). Viruses and a mathematical problem, *J. Molecular Biology* **24**: 337–338.

Greene, J. W. and Supowit, K. J. (1986). Simulated annealing without rejected moves, *IEEE Trans. CAD* **5**: 221–228.

Gáspár, Z. (1989). Some new multi-symmetric packings of equal circles on a 2-sphere, *Acta Cryst. Sect. A* **45**: 452–453.

Hales, T. C. (1992). The sphere packing problem, *J. Comput. Appl. Math.* **44**: 41–76.

Hales, T. C. (1993). Remarks on the density of sphere packings in three dimensions, *Combinatorica* **13**: 181–197.

Hestenes, M. (1980). *Conjugate Direction Methods in Optimization*, Applications of Mathematics 12, Springer-Verlag, Berlin.

Hilbert, D. (1901). Mathematische Probleme, *Archiv. Math. Phys.* **1**: 44–63.

Hsiang, W.-Y. (1993). On the sphere packing problem and the proof of Kepler's conjecture, *Internat. J. Math.* **4**: 739–831.

Hsiang, W.-Y. (1994). The geometry of spheres, *Normat* **1**: 1–12.

Johnson, D. S., Aragon, C. R., McGeoch, L. A. and Schevon, C. (1989). Optimization by simulated annealing: an experimental evaluation; Part I, Graph partitioning, *Oper. Res.* **37**: 865–892.

Johnson, D. S., Aragon, C. R., McGeoch, L. A. and Schevon, C. (1991). Optimization by simulated annealing: an experimental evaluation; Part II, Graph coloring and number partitioning, *Oper. Res.* **39**: 378–406.

Kirkpatrick, G. and Searle, P. (1975). *Images from Life*, British Museum (Natural History), London.

Kirkpatrick, S., Gelatt Jr., C. D. and Vecchi, M. P. (1983). Optimization by simulated annealing, *Science* **220**: 671–680.

Kling, S. A. (1978). Radiolaria, *in* B. U. Haq and A. Boersma (eds), *Introduction to Marine Micropaleontology*, Elsevier, New York, chapter 9.

Kottwitz, D. A. (1991a). The densest packing of equal circles on a sphere, *Acta Cryst. Sect. A* **47**: 158–165.

Kottwitz, D. A. (1991b). The densest packing of equal circles on a sphere, Supplementary publication, British Library Document Supply Centre.

Lachaud, G. and Stern, J. (1994). Polynomial-time construction of codes II: Spherical codes and the kissing number of spheres, *IEEE Trans. Inform. Theory* **40**: 1140–1146.

Landau, H. J. (1971). How does a porcupine separate its quills, *IEEE Trans. Inform. Theory* **17**: 157–161.

Mackay, A. L. (1980). The packing of three-dimensional spheres on the surface of a four-dimensional hypersphere, *J. Phys.* **13**: 3373–3379.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. and Teller, E. (1953). Equation of state calculations by fast computing machines, *J. Chem. Phys.* **21**: 1087–1092.

Molnár, J. (1975). On a generalization of the Tammes problem, *Publ. Math. Debrecen* **22**: 109–114.

Muder, D. J. (1993). A new bound on the local density of sphere packings, *Discrete Comput. Geom.* **10**: 351–375.

Nocedal, J. (1992). Theory of algorithms for unconstrained optimization, *Acta Numerica 1992*, Cambridge University Press, Cambridge, pp. 199–242.

Nurmela, K. J. (1993). Constructing combinatorial designs by local search, *Research report A27*, Digital Systems Laboratory, Helsinki University of Technology.

Nurmela, K. J. and Östergård, P. R. J. (1993). Upper bounds for covering designs by simulated annealing, *Congr. Numer.* **96**: 93–111.

Nurmela, K. J. and Östergård, P. R. J. (1994). Solving circle-packing problems by stochastic optimization methods. In preparation.

Reeves, C. R. (ed.) (1993). *Modern Heuristic Techniques for Combinatorial Optimization*, Halsted Press, New York.

Robinson, R. M. (1961). Arrangement of 24 points on a sphere, *Math. Ann.* **144**: 17–48.

Scales, L. E. (1985). *Introduction to Non-linear Optimization*, Macmillan.

Schoen, F. (1991). Stochastic techniques for global optimization: a survey of recent advances, *J. Global Optim.* **1**: 207–228.

Schwefel, H.-P. (1977). *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, Interdisciplinary Systems Research 26, Birkhäuser, Basel.

Shannon, C. E. (1948). A mathematical theory of communication, *Bell System Tech. J.* **27**: 379–423.

Slepian, D. (1968). Group codes for the Gaussian channel, *Bell Syst. Tech. J.* **47**: 575–602.

Sloane, N. J. A. (1980). A note on the Leech lattice as a code for the Gaussian channel, *Inform. Contr.* **46**: 270–272.

Sloane, N. J. A. (1981). Tables of sphere packings and spherical codes, *IEEE Trans. Inform. Theory* **27**: 327–338.

Sommerville, D. M. Y. (1958). *An Introduction to the Geometry of n Dimensions*, Dover, New York.

Tammes, P. M. L. (1930). On the origin of number and arrangement of the places of exit on the surface of pollen grains, *Recueil Travaux Botaniques Néerlandais* **27**: 1–84.

Tarnai, T. (1984). Spherical circle-packing in nature, practice and theory, *Topologie Structurale* **9**: 39–58.

Tarnai, T. and Gáspár, Z. (1983). Improved packing of equal circles on a sphere and rigidity of its graph, *Math. Proc. Cambridge Philos. Soc.* **93**: 191–218.

Therrien, C. W. (1989). *Decision, Estimation, Classification: An Introduction to Pattern Recognition and Related Topics*, Wiley, New York.

Törn, A. and Žilinskas, A. (1989). *Global Optimization*, Lecture Notes in Comput. Sci. 350, Springer-Verlag, Berlin.

van der Waerden, B. L. (1961). Pollenkörner, Punktverteilungen auf der Kugel und Informationstheorie, *Die Naturwissenschaften* **7**: 189–192.

Vanderbilt, D. and Louie, S. G. (1984). A Monte Carlo simulated annealing approach to optimization over continuous variables, *J. Comput. Phys.* **56**: 259–271.

Černy, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm, *J. Optim. Theory Appl.* **45**: 41–51.

Wyner, A. D. (1969). On coding and information theory, *SIAM Review* **11**: 317–346.

Appendix: Listings of the Codes

Table A.1: Coordinates of 3-dimensional codes.

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$
$k = 43$		
1 of 43	0.420805066434	-1.699011957230
2 of 43	1.887095704977	-1.064557538355
3 of 43	2.019919271721	0.231476846394
4 of 43	1.532358472518	0.516304349703
5 of 43	1.189847975376	0.972108136001
6 of 43	2.786979551279	-1.715248681795
7 of 43	1.306579415748	2.093840387189
8 of 43	0.157376515880	2.001810604096
9 of 43	1.473633675570	-0.628762169352
10 of 43	2.420472282937	2.145203041472
11 of 43	1.086598835860	2.753437443208
12 of 43	0.987238622362	0.378944929650
13 of 43	2.276530085009	0.832214853960
14 of 43	1.865308598586	2.054390142421
15 of 43	2.028099655134	-2.190607197234
16 of 43	2.134934068235	2.790835701955
17 of 43	1.746395839821	1.036137745514
18 of 43	0.629976811799	0.992418372347
19 of 43	0.466756377103	-0.084784257976
20 of 43	1.527269796217	-0.045261098798
21 of 43	1.156908485879	-2.910809066170
22 of 43	0.713344390620	2.208329768922
23 of 43	1.570796326795	1.570796326795
24 of 43	2.146283540465	-2.823558119848
25 of 43	2.232347701867	-1.570034213949
26 of 43	2.658997953834	3.099903716551
27 of 43	1.365367960591	-2.100671329679
28 of 43	0.849125608079	-2.348586975177
29 of 43	2.414485610160	-0.832642275104
30 of 43	2.569931699207	0.083765322130

Table A.1: Coordinates of 3-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$
31 of 43	1.675058507691	-1.631484235799
32 of 43	1.616542182926	2.565154982424
33 of 43	2.899130804694	1.415118510284
34 of 43	1.627280732318	-2.598604675525
35 of 43	1.980570889072	-0.384831750294
36 of 43	0.607498109823	-3.128128634391
37 of 43	1.669300584640	3.124306839225
38 of 43	0.980260330838	-1.654665380693
39 of 43	1.011576132097	-0.290079032409
40 of 43	2.127650741679	1.507294381432
41 of 43	0.811787685328	-0.959943460580
42 of 43	1.010712811230	1.570796326795
43 of 43	1.338806312026	-1.180470337516
$k = 77$		
1 of 77	0.602236347697	-0.414833209999
2 of 77	0.983991088734	-0.870439178066
3 of 77	1.212444157431	-1.266879619020
4 of 77	2.486847791595	2.652991706298
5 of 77	2.454595238731	0.439028651939
6 of 77	0.759909237782	1.765084219234
7 of 77	1.120985263024	2.032086718972
8 of 77	0.689375633602	-2.673510523351
9 of 77	2.561913647748	-1.007774058203
10 of 77	1.218903415267	0.340114101671
11 of 77	2.338037641330	-1.573866275402
12 of 77	1.805787726010	-0.764295705619
13 of 77	2.552206359148	1.931570574080
14 of 77	1.740966325178	0.060692396168
15 of 77	2.485718745815	-2.169425754710
16 of 77	1.901914863482	-0.339215709079
17 of 77	1.360794863126	2.396535050008
18 of 77	2.219118002784	-2.632223738551
19 of 77	1.491283524285	-2.726265941413
20 of 77	0.638654315933	1.133885802268
21 of 77	2.069662309608	-2.103778720468
22 of 77	2.615934092301	1.132369318364
23 of 77	1.620429231423	0.464600161625
24 of 77	1.656985200796	2.742077911623
25 of 77	2.159037906270	0.032010203562
26 of 77	1.673434591890	-1.960208284204
27 of 77	1.932674795184	1.786619945177
28 of 77	1.151894901830	1.570796326795
29 of 77	1.503221835471	-0.469797274980
30 of 77	2.869613993609	-1.757921880508
31 of 77	1.519114421791	3.138015451104

Table A.1: Coordinates of 3-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$
32 of 77	2.183921221186	0.957768283315
33 of 77	0.616680341236	-1.157894306931
34 of 77	0.296175207524	-3.005737825906
35 of 77	2.213827040155	-0.658345071368
36 of 77	1.057273041343	1.112842924111
37 of 77	0.471115226251	0.384611832611
38 of 77	1.678738366063	-1.196102987394
39 of 77	2.915252836572	2.524367690710
40 of 77	0.752789311027	2.975885970624
41 of 77	0.909080610382	-1.600765563747
42 of 77	2.527943730608	-0.251959583812
43 of 77	1.029334095224	-0.376083807926
44 of 77	0.242933238963	1.493694943245
45 of 77	0.479263306162	-1.937587048303
46 of 77	1.439027532336	-2.308222829854
47 of 77	0.948456465653	2.478222477009
48 of 77	2.907704989256	0.230227748755
49 of 77	1.473518819313	1.162694527880
50 of 77	1.775310789083	2.335639792316
51 of 77	2.224364247663	3.122119799432
52 of 77	2.067954907535	2.656594666511
53 of 77	2.257694821143	1.478262499446
54 of 77	1.537777383837	1.988471208023
55 of 77	2.094817527675	-1.142534265912
56 of 77	2.169064899723	2.178736654798
57 of 77	1.502828718408	-1.577097390025
58 of 77	0.536088932643	2.365407654382
59 of 77	1.345075122612	-0.077194108726
60 of 77	1.920336837496	-1.611845737689
61 of 77	0.854223921102	0.097670300823
62 of 77	1.100492757038	-2.568243364460
63 of 77	0.892540929779	0.644853077798
64 of 77	2.038112869504	0.497911060024
65 of 77	1.113198561858	-3.037326528647
66 of 77	1.402808857409	-0.879543414301
67 of 77	1.570796326795	1.570796326795
68 of 77	1.876665729240	1.277930885230
69 of 77	1.825404136075	-2.471400147264
70 of 77	0.880095899971	-2.139209897558
71 of 77	1.254376819879	-1.921785275250
72 of 77	1.775011943874	0.857546587056
73 of 77	1.244175573321	2.814314981206
74 of 77	1.871333179149	-2.904654277851
75 of 77	1.325404961937	0.764553352925
76 of 77	2.608019546950	-2.899197341390

Table A.1: Coordinates of 3-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$
77 of 77	0.211841952727	-0.852415585848
$k = 80$		
1 of 80	2.321297798179	-2.212494023858
2 of 80	0.820294855411	-0.929098629731
3 of 80	2.409926296756	0.686418946105
4 of 80	1.573215087705	-2.250918716494
5 of 80	1.886090368054	-1.570796326795
6 of 80	3.072383440217	-2.621864943637
7 of 80	0.781558270004	-2.045029303936
8 of 80	1.639140886552	-0.485433974243
9 of 80	2.043945040847	2.593016064856
10 of 80	1.289352195848	1.715384875714
11 of 80	1.733136831544	2.877844618048
12 of 80	1.950407021327	-1.138486105324
13 of 80	0.926909272493	2.950840526179
14 of 80	2.214683381096	0.190752127410
15 of 80	1.647445208686	-0.071825023059
16 of 80	1.286806558666	-0.271289907954
17 of 80	2.167043796913	1.131965272667
18 of 80	0.672191452705	0.349518287879
19 of 80	0.819228646682	0.921903042439
20 of 80	1.998993416628	0.699762549655
21 of 80	1.143755314509	-2.447689305757
22 of 80	2.322364006907	2.219689611151
23 of 80	1.324426086265	2.833531431893
24 of 80	0.888720349301	-0.388469034799
25 of 80	1.581604025010	2.493894213881
26 of 80	1.689730177097	1.812058490341
27 of 80	1.255502285535	-1.570796326795
28 of 80	1.413335994525	2.117367850314
29 of 80	0.409821912316	0.989389964965
30 of 80	1.192512648498	-1.138374075203
31 of 80	1.494147444903	-3.069767630531
32 of 80	0.391753066708	-2.293943529957
33 of 80	0.762179164717	-2.638313845220
34 of 80	1.275792171647	0.949325395874
35 of 80	1.852240457742	1.426207777876
36 of 80	1.728256659065	1.024224803276
37 of 80	2.025790981658	-0.239645409601
38 of 80	2.749839586882	-0.847649123632
39 of 80	0.974548856676	2.009627380923
40 of 80	2.682486361978	-1.870371297937
41 of 80	2.618240435013	0.068658672333
42 of 80	2.463214487649	1.525876046265
43 of 80	2.662670645893	-2.806356608492

Table A.1: Coordinates of 3-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$
44 of 80	1.997837339081	-0.693903347833
45 of 80	2.360034383586	-1.096563349654
46 of 80	1.571428137437	-1.301731255010
47 of 80	0.459106291612	-1.271221355652
48 of 80	0.478922007697	-0.335236045097
49 of 80	1.854786094923	-2.870302745636
50 of 80	2.100005305911	1.784874956103
51 of 80	1.451862476493	1.329534163249
52 of 80	1.408455822046	0.263748035542
53 of 80	1.142599236962	2.441830103935
54 of 80	1.937991671853	-2.444921718729
55 of 80	1.191185632263	-2.003106548266
56 of 80	0.678378165941	1.615716607324
57 of 80	1.203600981736	-0.696670934861
58 of 80	2.379413488873	-0.503278808370
59 of 80	0.069209213373	-0.519727709953
60 of 80	1.865800481943	2.192267257716
61 of 80	1.041587347679	1.356717697487
62 of 80	2.252872304289	-2.753123618791
63 of 80	0.348342601605	2.147167542971
64 of 80	1.502451767038	-2.656158679347
65 of 80	2.731770741274	2.152202688625
66 of 80	0.851052782612	-1.485080014742
67 of 80	1.568377565885	-0.890673937096
68 of 80	1.559988628580	0.647698439708
69 of 80	0.523352218577	3.072933981257
70 of 80	1.033264668583	0.083433769521
71 of 80	1.570164516153	-1.839861398580
72 of 80	0.731666356834	2.455173707485
73 of 80	2.108327985007	3.058158884069
74 of 80	2.469401200885	2.792074365711
75 of 80	1.817166567325	0.308061221697
76 of 80	1.115801671932	-2.901947243989
77 of 80	1.949080005092	-2.003218578386
78 of 80	2.290539870978	-1.656512638847
79 of 80	1.097647612743	0.548576588733
80 of 80	2.793250051985	0.994425110619

Table A.2: Coordinates of 4-dimensional codes.

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
$k = 9$			
1 of 9	1.507130148533	2.132410797995	2.450181356389
2 of 9	1.537112133946	2.331550910634	0.535153996083
3 of 9	2.832856654438	2.871178301476	2.236854732574
4 of 9	1.378811758703	1.432129668308	-0.673880040257
5 of 9	1.340689777945	1.222986231336	-2.694434125947
6 of 9	1.547152351721	2.405590040455	-1.838098333524
7 of 9	0.188337968282	3.140982127101	-2.376417974709
8 of 9	2.040378350559	0.153120326958	-0.902993322597
9 of 9	1.327839725255	1.150074087007	1.377266952073
$k = 10$			
1 of 10	1.701971326833	0.209011224795	2.208055750108
2 of 10	0.359738518953	0.412221542033	-1.547318957498
3 of 10	2.079678967126	2.457241284772	-0.074237127784
4 of 10	1.050335046714	2.539442464364	1.776125388320
5 of 10	1.400764437012	1.508938500332	3.083047584521
6 of 10	1.316546258472	2.172708354005	-1.831408613737
7 of 10	1.619308613810	1.201119846206	-0.792221886534
8 of 10	2.523988838395	1.398511280660	-2.190323828855
9 of 10	1.372226487412	1.371689009908	0.659575524719
10 of 10	2.176703081733	1.653259144194	1.850178404201
$k = 11$			
1 of 11	2.333839199482	1.242033486771	1.954953748510
2 of 11	1.023525279938	1.496958063584	1.797617451425
3 of 11	1.027780026061	1.786725449202	-2.876965146861
4 of 11	2.221575069649	1.416839810710	-2.342592401138
5 of 11	1.740052135415	1.822466844502	0.649531363314
6 of 11	0.723110953205	2.586327404576	-0.204910509970
7 of 11	1.970744289030	2.264881138121	-0.891389054114
8 of 11	1.425755349640	0.422359273627	3.079731169344
9 of 11	1.187558936542	1.267223846987	-1.425411380717
10 of 11	2.146320559450	0.783897264043	-0.393935388045
11 of 11	1.737626595888	2.630996209572	2.239345734492
$k = 12$			
1 of 12	1.007824374885	1.206478759776	1.010725733246
2 of 12	2.705253563758	1.236388812602	-1.914632552156
3 of 12	1.443019626277	1.172833435247	-2.459432587706
4 of 12	1.120851237186	1.649059827974	2.526375382260
5 of 12	2.057227747994	1.499439548161	0.200926983490
6 of 12	2.212820900667	1.976490431240	1.787152435756
7 of 12	1.765684586235	2.439014611796	-2.703449410451
8 of 12	1.323097378979	2.587481193629	0.587758929683
9 of 12	0.351483305816	1.962172990220	-1.559251091338
10 of 12	1.633149739775	1.950182229899	-1.066552300633

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
11 of 12	1.342606090294	0.682134245505	-0.764614145614
12 of 12	1.852228666543	0.553864202417	2.190874104699
$k = 13$			
1 of 13	1.867418975901	1.499453342602	2.987740497910
2 of 13	2.127031717239	1.376422555841	0.030964572502
3 of 13	1.801839401672	2.405204846487	1.008199091435
4 of 13	1.182588351306	0.856904958793	-0.765170894230
5 of 13	1.390130452250	1.423345693472	-2.111246326028
6 of 13	2.666932023063	2.539955904658	-2.024995728841
7 of 13	1.078217382397	1.360150074423	0.778029555976
8 of 13	1.440834771003	2.604988760996	-2.710960800269
9 of 13	2.072732972943	1.179209977289	1.604155637631
10 of 13	1.131393608846	0.539203688581	2.441248817345
11 of 13	0.205156995271	2.276827868126	-2.180327315455
12 of 13	1.332851344393	2.177501095885	-0.636140704148
13 of 13	2.184003995208	0.389983106436	-2.024212246549
$k = 14$			
1 of 14	1.186246056234	1.824334125438	-1.170989155857
2 of 14	2.354175378527	1.863148523341	-1.662407387765
3 of 14	1.681814451104	1.316382877481	1.231289504234
4 of 14	1.446979616222	2.033157020319	-2.680635883662
5 of 14	0.921190924360	1.949042531484	0.363311706243
6 of 14	1.672973380657	2.949474286265	-0.474441838042
7 of 14	1.194204652973	0.538086153385	0.018274739099
8 of 14	2.363344230274	0.105776293234	0.843650300124
9 of 14	2.443853607800	2.127143642101	2.147538705445
10 of 14	1.454422403149	0.936776853955	2.559065478856
11 of 14	2.111401636659	1.722006309277	0.030475997862
12 of 14	1.198592701010	2.131657046528	2.110734907259
13 of 14	0.324536770864	1.160592948577	-2.766459416886
14 of 14	1.504178318772	0.915989748846	-2.081028662263
$k = 15$			
1 of 15	1.170311269834	2.714530824764	-2.909273774954
2 of 15	1.462240692658	1.584571036781	2.704319271433
3 of 15	1.506581595878	1.091582429062	-0.990879496815
4 of 15	2.618380045366	1.253436192768	-0.341587538357
5 of 15	0.916398448694	0.954823742164	-2.477704537041
6 of 15	0.909921602101	1.457140873855	1.508722202931
7 of 15	2.156462206817	0.800040828481	-2.790209699811
8 of 15	2.398361235236	2.417543388671	3.017987531398
9 of 15	0.512144287902	1.909273092260	-0.548512531884
10 of 15	2.151311736393	1.240505090321	1.688248066769
11 of 15	1.438883913489	0.276973202730	0.992259507427
12 of 15	1.502272678697	1.419112669598	0.256873755698
13 of 15	1.724825009256	1.802479222548	-1.983770065737
14 of 15	1.718227708209	2.351650468255	1.090035968763

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
15 of 15	1.690442361940	2.254762339824	-0.673069341059
$k=16$			
1 of 16	2.171649910216	0.948008692261	-2.783703977110
2 of 16	2.210101048610	0.569660120304	0.033010828819
3 of 16	2.338008006648	1.471592048799	1.847124682279
4 of 16	1.436970168893	2.609015062333	1.979985255378
5 of 16	2.226755518253	2.091038619410	0.260794503645
6 of 16	1.327371102323	1.149325351848	0.847216856814
7 of 16	1.250606710220	1.223332982259	-1.968603592348
8 of 16	0.123100424908	0.523309255561	-2.696438887705
9 of 16	1.195852363353	1.037663671561	-0.565692391156
10 of 16	1.391432878142	1.844803028854	-3.022831832424
11 of 16	1.191839415833	1.419320294800	2.114870201691
12 of 16	2.353017481826	2.533623899947	-2.656598507035
13 of 16	2.060472630812	1.620359862334	-1.171729891882
14 of 16	1.063113257521	2.182402315369	0.111381661757
15 of 16	1.284908593975	0.210935964130	2.719522316851
16 of 16	1.305065625306	2.431158266155	-1.699526820104
$k=17$			
1 of 17	2.545481957334	0.688838012492	2.921488731433
2 of 17	1.580810221436	1.978621881839	-0.706288774152
3 of 17	2.015528353661	2.147791468421	2.970252371739
4 of 17	1.571213799592	1.762088074183	0.481422964910
5 of 17	1.109195284365	2.079377935808	-1.943660046123
6 of 17	1.495362797664	3.057750011591	0.614880508411
7 of 17	1.493067108711	1.061888155064	2.378324070928
8 of 17	0.548666015984	2.237713025895	-0.027800313014
9 of 17	1.380426327536	0.670990686622	0.852683415000
10 of 17	1.589998917293	1.239885921362	-2.650213395905
11 of 17	2.174064568599	1.006953624424	-0.277721808553
12 of 17	1.663506286228	1.966288978151	1.666620467590
13 of 17	2.230769875806	1.863979019456	-1.805205899451
14 of 17	0.639929457392	0.469682954724	-2.732628643136
15 of 17	0.891783048730	2.102671591516	2.693319109275
16 of 17	2.544953442272	2.497512780068	0.650713267802
17 of 17	1.452246040287	1.049615677190	-1.396546941516
$k=18$			
1 of 18	1.004648609815	1.584065955945	-2.190436242250
2 of 18	2.331638701331	2.091988452178	2.304855721957
3 of 18	2.041342033199	0.698799680741	2.158557783555
4 of 18	2.078003438960	1.557868845030	-2.575844168884
5 of 18	1.799359538355	1.703566318806	-0.323767803530
6 of 18	1.440150858474	2.459811915238	-3.049665161843
7 of 18	1.603621419648	0.574538497973	-1.996998544859
8 of 18	2.621065178798	0.805461914034	-0.997139358845
9 of 18	1.100250620391	2.442792972849	-0.983034870035

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
10 of 18	1.619368576539	2.559082324070	0.993538232664
11 of 18	0.410204585949	2.288073080974	2.234095059685
12 of 18	1.063589214630	1.583723808560	0.565748484706
13 of 18	0.917056436296	1.092298974781	-0.789808712183
14 of 18	1.404548035368	1.611110898595	1.739404431711
15 of 18	0.961430171701	0.460709664836	1.635314382440
16 of 18	2.160817302094	1.491370519475	0.855361591961
17 of 18	1.291103294442	1.349101046896	2.879756378355
18 of 18	2.204758233697	2.574998144149	-1.392395949094
$k = 19$			
1 of 19	1.595062579776	1.260462250442	-0.490425145943
2 of 19	1.546530073814	1.881130403147	2.651167507647
3 of 19	0.739488525786	1.413591729047	-2.847748630360
4 of 19	1.876938995757	1.809199757514	-1.464426788068
5 of 19	1.797351334400	2.307717161089	1.436955007121
6 of 19	1.837824229431	1.475500517153	-2.599609132018
7 of 19	2.442015422827	0.581568946317	-1.597815326304
8 of 19	1.356347892976	2.485956587605	-2.407844716628
9 of 19	0.699577230763	2.560023707273	1.543777327286
10 of 19	2.402104127804	1.728000924542	0.293844023230
11 of 19	1.303768424159	1.666092136437	0.541983521572
12 of 19	1.604134224285	2.485373332175	-0.343939784465
13 of 19	1.785244760614	0.655636065985	0.733747936962
14 of 19	0.793582063951	1.871937766593	-1.140677325127
15 of 19	1.344241319190	0.833875492501	-1.704637646468
16 of 19	2.456218577845	2.653705763661	-2.758762701680
17 of 19	1.264653657833	1.332392896076	1.677165865522
18 of 19	0.685374075745	0.487886889929	0.382829951910
19 of 19	2.348010589639	1.269654886997	2.000915328463
$k = 20$			
1 of 20	1.602157428551	1.446040733450	2.391451174556
2 of 20	1.292387020701	0.385568760444	-3.021375480335
3 of 20	0.602410755367	0.938814882150	-0.871977085563
4 of 20	1.131740668339	2.394185707566	1.873515051296
5 of 20	2.426606221769	1.554587309484	0.036208325413
6 of 20	1.849205632888	2.756023893145	0.120217173255
7 of 20	2.009851985250	0.747406946024	-1.268077602294
8 of 20	1.496355655769	0.854785615097	0.235012895345
9 of 20	2.209939989521	0.578601764390	1.688099967711
10 of 20	0.859853446365	1.101332996932	1.482889492530
11 of 20	1.645236997821	2.286807038493	-2.906579758245
12 of 20	2.281739207225	2.040259656658	-1.658703161060
13 of 20	1.539435225039	1.695551920140	-0.750141479034
14 of 20	0.714986431820	1.587005344106	-3.105384328177
15 of 20	1.054164054411	1.945868915893	0.328473442726
16 of 20	1.804734874409	1.686159392170	1.183333858126

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
17 of 20	2.539181898223	2.202777771440	2.269615568026
18 of 20	2.087428599179	1.195723737697	-2.813119210864
19 of 20	0.931652664068	2.562990889200	-1.453492685879
20 of 20	1.336857779181	1.455433261420	-1.958258795464
$k = 21$			
1 of 21	0.923411863784	2.543200163652	-1.173115914740
2 of 21	1.866391840590	3.061525078070	-2.171897784947
3 of 21	1.313347496643	1.584210069236	-0.394393311385
4 of 21	1.385641410372	0.759894770552	-2.628852846840
5 of 21	1.756434020385	1.501320171122	1.473514668778
6 of 21	0.672971661819	1.906513448152	0.935949785257
7 of 21	1.008598657564	0.888992214317	2.068548055131
8 of 21	1.993685121767	0.433750103127	1.918439075628
9 of 21	2.142777896833	1.938329288440	-1.041648857070
10 of 21	1.353171483228	0.913118716668	0.572764281463
11 of 21	2.399592805718	1.022601166410	-2.206332046789
12 of 21	1.782125875844	1.406628225789	2.735245716868
13 of 21	1.300866560205	2.238629461712	2.186271760620
14 of 21	1.704228914391	0.671092256628	-0.920619076287
15 of 21	1.339306827065	1.489957367796	-1.662215338958
16 of 21	1.728055708844	1.965013805657	-2.580552373717
17 of 21	1.689095589145	2.121509225924	0.527159704525
18 of 21	2.641916295762	2.095237168306	2.115238889759
19 of 21	0.625950907411	0.604874934105	-0.963977475765
20 of 21	2.363041681210	1.190603210019	0.201341676681
21 of 21	0.712414062482	1.745752516710	-2.952970728018
$k = 22$			
1 of 22	2.254302334175	1.694950759058	-2.208184617473
2 of 22	2.130238283023	1.295308755195	-0.366659375953
3 of 22	1.053805554542	1.701030973049	-2.242508920534
4 of 22	1.045652356171	2.718997542483	1.363920048517
5 of 22	1.009078106173	1.835438698022	2.772182540211
6 of 22	1.088705540853	1.158808776523	-0.375948254537
7 of 22	1.601440649018	1.357696143398	-1.363088967185
8 of 22	2.043996133607	1.839680546485	2.581498590832
9 of 22	2.097266324806	2.448203452005	-0.915466201019
10 of 22	1.633696398422	1.117986052008	-2.968980373679
11 of 22	1.067823389011	2.244993808385	-1.013756826886
12 of 22	2.057789587940	2.432163479121	1.268731546825
13 of 22	1.392275962051	1.674878276097	1.697606861155
14 of 22	2.092702720382	0.414948390959	-1.759656482003
15 of 22	0.041866729259	1.460227857136	-0.164161003943
16 of 22	1.048827126464	0.490036765049	-1.960730431293
17 of 22	1.500749055100	2.021231710047	0.179469253531
18 of 22	1.631110480673	2.531622559069	-2.730607432451
19 of 22	1.060909653087	0.738595675164	1.105381931444

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
20 of 22	1.899653849408	0.802070839261	2.060391216592
21 of 22	1.936832896507	1.302051220608	0.855574728508
22 of 22	2.944513919007	1.105691099909	1.508705878918
$k = 23$			
1 of 23	1.587826630328	1.839881012364	-1.197179816256
2 of 23	0.521957703856	1.532032240838	1.923618493921
3 of 23	1.382645172860	2.341089446177	1.892191195544
4 of 23	1.219799721071	0.992480217419	0.865126064794
5 of 23	1.741639713856	0.269516769758	2.085132627330
6 of 23	0.823304010347	0.287094806558	-1.324853885543
7 of 23	1.913429058190	1.774013976516	1.049497891216
8 of 23	2.110459822010	1.036885579126	-2.616660690245
9 of 23	1.020416559811	2.188973061418	-2.951825430083
10 of 23	1.903716595825	1.835740253127	2.807788967066
11 of 23	1.210023479143	1.061836086300	3.022072063329
12 of 23	2.586431087121	1.084167581925	1.966649482732
13 of 23	1.553766023262	1.301711641226	1.944412837334
14 of 23	2.619634949734	1.609560412752	-1.217974159669
15 of 23	1.758947480730	0.800503207413	-1.249401458045
16 of 23	1.921792932519	2.149112436171	-2.276466588795
17 of 23	1.399952939734	2.872075883832	-1.056460026260
18 of 23	2.318288643243	2.854497847032	1.816738768047
19 of 23	1.228163595400	1.367578677074	-2.092094762374
20 of 23	1.031132831579	2.104707074464	0.524931963345
21 of 23	2.121176093779	0.952619592172	0.189767223507
22 of 23	1.237876057765	1.305852400463	-0.333803686524
23 of 23	1.931569174446	2.079756567290	-0.119520590261
$k = 24$			
1 of 24	1.587826630328	1.839881012364	-1.197179816256
2 of 24	0.521957703856	1.532032240838	1.923618493921
3 of 24	1.382645172860	2.341089446177	1.892191195544
4 of 24	1.219799721071	0.992480217419	0.865126064794
5 of 24	1.741639713856	0.269516769758	2.085132627330
6 of 24	0.823304010347	0.287094806558	-1.324853885543
7 of 24	1.913429058190	1.774013976516	1.049497891216
8 of 24	2.110459822010	1.036885579126	-2.616660690245
9 of 24	1.020416559811	2.188973061418	-2.951825430083
10 of 24	1.903716595825	1.835740253127	2.807788967066
11 of 24	1.210023479143	1.061836086300	3.022072063329
12 of 24	2.586431087121	1.084167581925	1.966649482732
13 of 24	1.553766023262	1.301711641226	1.944412837334
14 of 24	2.619634949734	1.609560412752	-1.217974159669
15 of 24	1.758947480730	0.800503207413	-1.249401458045
16 of 24	1.921792932519	2.149112436171	-2.276466588795
17 of 24	1.399952939734	2.872075883832	-1.056460026260
18 of 24	2.318288643243	2.854497847032	1.816738768047

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
19 of 24	1.228163595400	1.367578677074	-2.092094762374
20 of 24	1.031132831579	2.104707074464	0.524931963345
21 of 24	2.121176093779	0.952619592172	0.189767223507
22 of 24	1.237876057765	1.305852400463	-0.333803686524
23 of 24	1.931569174446	2.079756567290	-0.119520590261
24 of 24	0.555161566469	2.057425071665	-1.174943170858
$k = 25$			
1 of 25	2.946802213741	2.216428301813	1.405031963042
2 of 25	1.978589034208	1.855505033157	2.540283769331
3 of 25	2.259785598329	0.361130717834	-3.082659787154
4 of 25	1.040500606924	2.084983331054	0.690695259576
5 of 25	1.353659992266	2.126527452247	-2.421847196317
6 of 25	1.845525343818	1.073741748076	-1.657261894164
7 of 25	0.278019543775	2.339080863909	-2.643317159030
8 of 25	0.997054019552	0.783485449793	-2.265721990124
9 of 25	1.070289271355	1.687169948262	-1.406919694026
10 of 25	2.056044965887	2.848198664785	-2.808820692321
11 of 25	2.241863318757	1.650262082441	-2.582914452297
12 of 25	1.251897899952	2.694905753866	-0.830149330024
13 of 25	1.127444612007	1.564483938594	1.765706095923
14 of 25	1.190011561947	1.499402029119	-0.306501461061
15 of 25	1.865166217769	2.231401884186	1.388455556219
16 of 25	0.728190011914	0.796495254828	0.717431462299
17 of 25	1.132882382305	1.505457391997	2.979197013397
18 of 25	1.899889461165	2.134216142659	0.062867500727
19 of 25	1.662343409497	1.267124515305	0.586816451499
20 of 25	1.509639032494	0.395691477088	-0.501337126081
21 of 25	1.359958293838	0.622814123899	2.301616326646
22 of 25	1.193676221442	2.513038002118	2.446751127401
23 of 25	1.994489597931	2.045131835575	-1.204256665307
24 of 25	2.050831986127	1.179376206523	1.626909383739
25 of 25	2.235516735503	1.144350878762	-0.439412624846
$k = 26$			
1 of 26	0.254825753413	2.766338478100	1.288482345508
2 of 26	1.075949026440	1.576484867589	1.112880606772
3 of 26	1.937812782056	0.890001701143	2.853030046135
4 of 26	1.424200580706	0.927404649978	-1.199047992806
5 of 26	1.691208333019	1.184711432047	-2.289324197968
6 of 26	2.690986284336	1.506129658316	0.875830363534
7 of 26	1.184527535490	2.245271578114	2.207484627217
8 of 26	1.082118727556	1.869467754860	-0.844683767947
9 of 26	2.193833101164	2.648565263395	-0.479820748961
10 of 26	1.473004761230	1.927761115572	0.186064471823
11 of 26	2.585479168996	1.733051541517	-2.477580600189
12 of 26	1.408891628252	0.179384820862	1.504269654998
13 of 26	1.005316182776	1.123060558949	2.238938833824

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
14 of 26	1.005833209136	2.041913333356	-2.111905433727
15 of 26	1.229564089398	2.908200154876	-0.186476449302
16 of 26	1.851322745159	1.874845922891	-1.536061960968
17 of 26	0.862627137973	0.948340828546	-0.012209269068
18 of 26	1.791126358992	1.012072196222	0.463705993237
19 of 26	2.119269210293	1.920530299182	2.410594820790
20 of 26	1.785028514615	2.500963678148	-2.745252755558
21 of 26	2.277613890860	0.364366087694	-1.267470088582
22 of 26	0.767728096044	0.765629905570	-2.397620512435
23 of 26	1.846829863412	2.226412288173	1.220508269529
24 of 26	1.853387041603	1.169552704217	1.633068258309
25 of 26	2.045588620839	1.467827652912	-0.539406115588
26 of 26	1.338910346112	1.654082965837	-3.124330626715
$k = 27$			
1 of 27	1.166900381199	1.909910585508	-2.165870870261
2 of 27	1.615239198877	1.812045585335	-3.101670918252
3 of 27	2.534728723894	2.163640805945	-0.541300275514
4 of 27	2.078729749716	2.601506280809	-2.495826316998
5 of 27	0.827510516617	2.116389944217	-0.603703773022
6 of 27	1.163917687501	2.954932980616	-2.700228117869
7 of 27	1.416316161263	2.130609200844	2.183722735336
8 of 27	1.994634839275	1.570171585347	0.490992593156
9 of 27	2.712733808076	1.581798287552	2.866117809007
10 of 27	1.011185023994	0.813464703528	1.930237892641
11 of 27	0.552261443590	0.526959868207	-0.955415588831
12 of 27	1.527253159557	1.379690379291	-0.378976827819
13 of 27	1.350333119383	1.210168631562	-1.409776273139
14 of 27	1.607903532081	1.497289119863	1.420497331053
15 of 27	0.504989061306	1.957370820283	2.688507370534
16 of 27	1.817219501509	1.251141611780	2.379029199257
17 of 27	2.190160357486	2.346940694709	1.412279316206
18 of 27	1.648862522880	2.445487719627	0.050588989139
19 of 27	1.995576576347	0.639216314782	-2.823047807855
20 of 27	2.230460329023	0.677285179039	1.315792115505
21 of 27	1.083798010836	1.005884727717	-2.804936240148
22 of 27	2.194570285696	0.825378199823	-0.988389790558
23 of 27	1.474790895683	0.224485763518	-0.072370489862
24 of 27	1.674580026557	2.123828530982	-1.237375803857
25 of 27	0.960694301231	2.185684943279	1.021145781830
26 of 27	2.065651239112	1.530956577094	-2.035962776593
27 of 27	1.116938922134	1.130028531200	0.566203409596
$k = 28$			
1 of 28	1.524777523035	2.128771846014	-0.342829056881
2 of 28	0.944502328379	2.217442086988	0.687012782723
3 of 28	1.244695721846	1.970406523707	-2.492787475839
4 of 28	2.251610759042	0.815539346632	1.258272525186

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
5 of 28	1.902413207244	2.432307796925	3.079443728971
6 of 28	0.710344846254	1.555968015733	-0.576121113000
7 of 28	1.668348164179	1.665247974133	1.130870752976
8 of 28	1.381283038976	1.349901932876	0.252177717070
9 of 28	2.185873283128	0.439298442663	-2.149939534733
10 of 28	1.155062453537	1.500649652248	2.837867373963
11 of 28	2.077767150855	1.380032485824	3.117917627355
12 of 28	1.336086801147	0.937597866838	-1.297913498463
13 of 28	0.991173058855	1.249031141165	1.738827750186
14 of 28	0.671336001663	0.160890829178	-2.665658716888
15 of 28	1.603555506584	0.681220272645	2.445769558470
16 of 28	1.346836156141	1.928227348471	-1.405366490521
17 of 28	2.147868035848	1.377814097285	-0.929988716984
18 of 28	2.012164739509	1.557642372615	-2.037416434725
19 of 28	1.386029893643	0.965225831514	-2.543264865516
20 of 28	3.020253108378	1.958130732410	-2.904799822810
21 of 28	1.247624268309	2.922275842730	-1.685604994481
22 of 28	1.908767970742	2.627112536729	1.037038303882
23 of 28	1.255745082476	2.252064499682	2.060939309775
24 of 28	2.137074690409	1.793037882098	2.045420090389
25 of 28	1.525997597410	0.386055603907	0.190351648190
26 of 28	2.278567462549	1.741177724236	0.252232779289
27 of 28	2.142203459437	2.501919626297	-1.327699246593
28 of 28	0.387507991873	2.449237417905	3.109295192767
$k=29$			
1 of 29	1.698648080774	1.258945272569	1.837829351932
2 of 29	1.341186313055	1.838765048467	-2.771354056669
3 of 29	1.558793667971	2.420740173638	1.451770684388
4 of 29	2.235482157375	1.913882116931	0.784719413609
5 of 29	0.870048623287	1.692418317974	1.547595225736
6 of 29	1.317978535596	2.061224144166	0.301796145480
7 of 29	1.259780650053	1.112117837118	-0.217339431260
8 of 29	1.370005103453	1.904317826830	2.482011528365
9 of 29	0.865347663452	0.633171720439	-1.474294446509
10 of 29	0.607372988666	1.253395120171	2.964923841368
11 of 29	1.562816635602	0.975973526495	2.916061194071
12 of 29	1.728808888254	0.190232076417	-1.658168916216
13 of 29	2.150773344539	0.882892015695	0.124794973694
14 of 29	2.451161653215	0.615002413619	2.239696415764
15 of 29	1.965998978645	1.827353795334	-0.433740744729
16 of 29	1.581191369343	1.114141890295	-2.163706075719
17 of 29	1.521870562546	1.370279815839	-1.180507941041
18 of 29	2.272543433745	1.603140531243	-2.803045713366
19 of 29	1.777844920910	2.686828741903	-2.966943483281
20 of 29	1.421409100626	2.542359491949	-0.912537865726
21 of 29	2.286704993550	2.039355659157	2.192865963814

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
22 of 29	0.412108052256	1.787104701565	-0.100571011987
23 of 29	2.592164261717	2.696358514579	-1.252621251995
24 of 29	1.070565565866	0.530893789931	1.591388604513
25 of 29	0.844541323964	2.806196481718	2.793283578184
26 of 29	0.886475526425	1.914132971558	-1.761910388402
27 of 29	2.387776714494	1.007283789743	-1.493804318184
28 of 29	1.492595127585	1.265705440506	0.850674865101
29 of 29	1.836558739792	2.001066119621	-1.858055405141
$k = 30$			
1 of 30	1.657787908139	2.210535092396	-2.662281233786
2 of 30	1.950511192215	2.070889192821	1.044588063832
3 of 30	2.299589246009	1.083201690118	0.636617345019
4 of 30	1.496371906994	0.822271346468	1.239738182292
5 of 30	1.376044193296	0.928741368854	-1.249251628396
6 of 30	1.848828272824	0.103457812614	-0.990553171774
7 of 30	1.773096715818	2.710798567456	-1.206216604659
8 of 30	2.621229190668	2.468284164982	-0.163923128761
9 of 30	1.499344151365	1.821804277749	-0.910408154819
10 of 30	1.164194094190	1.619324908129	0.729989950069
11 of 30	1.968003784682	1.458026226746	3.040425000071
12 of 30	2.213605597475	1.166930944824	-0.778706424342
13 of 30	1.332877093353	0.952352400665	2.486261812620
14 of 30	1.058086646586	2.407195753276	-0.100306661945
15 of 30	1.376294673900	0.942246843115	-0.018395019555
16 of 30	1.780230570570	1.019014643220	-2.303510947099
17 of 30	1.196811569764	1.917309259808	2.708197779625
18 of 30	1.340361759643	1.692760739616	1.716976025475
19 of 30	1.277183229206	1.713444877239	-1.863923206514
20 of 30	2.142994685985	1.227578711066	1.965940535869
21 of 30	0.906337392180	0.017706227167	0.094223969440
22 of 30	0.544619432973	1.456197693717	-0.929598867778
23 of 30	0.883390123666	1.231118198820	-2.751924338141
24 of 30	2.201769803853	2.344871797709	2.370232030522
25 of 30	2.690911711895	0.610914186788	-2.725710313225
26 of 30	2.215751178516	1.836259282304	-1.813548455704
27 of 30	0.415227645565	1.502744586739	1.948792908288
28 of 30	0.781923086127	2.583781556872	-2.433833249005
29 of 30	1.363636322541	2.663522574967	1.774248393829
30 of 30	1.795808384031	1.796669032611	0.023202289548
$k = 31$			
1 of 31	1.611498828169	0.695002129601	0.993984753764
2 of 31	1.564745254046	1.461931405716	-3.120392250137
3 of 31	1.708294842169	0.635038458218	2.636089211454
4 of 31	1.245748732903	3.059300210737	-0.109291747660
5 of 31	2.728727978735	2.664148326899	-0.536420043115
6 of 31	2.308269321884	1.378558954400	1.201749330853

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
7 of 31	1.541008218437	1.579159593634	0.633274289872
8 of 31	2.202761501369	1.404683918414	-1.382921579581
9 of 31	0.865992146860	1.342992717500	-2.420404220960
10 of 31	0.364576106558	2.691320911300	2.561867777375
11 of 31	1.244755332139	1.023853066635	-0.128258709032
12 of 31	2.186863938668	1.333048013347	-0.060435194565
13 of 31	2.003290813515	2.650706095099	2.707453751445
14 of 31	1.264931950907	1.100704877483	-1.379989538271
15 of 31	0.970949835733	2.105756292639	-1.441642876845
16 of 31	1.234615609386	2.122803337195	2.429920249802
17 of 31	2.243197073682	1.846927821268	-2.520804220769
18 of 31	2.604294086801	0.466412086702	-3.006885287983
19 of 31	0.880664156599	1.106176086253	2.522010867082
20 of 31	1.033800979226	0.094815373454	-1.615652404664
21 of 31	1.881789426539	0.480644715273	-0.826185147055
22 of 31	1.487028398259	1.398296677118	1.736248848572
23 of 31	1.864555146036	2.408048913528	-1.450080884806
24 of 31	1.381850684015	2.243179292031	-2.582772924333
25 of 31	0.919473485635	2.044060211848	-0.001566421726
26 of 31	2.162258220267	1.598629147360	2.400594024937
27 of 31	1.847018646771	2.328345608532	0.077139269090
28 of 31	0.748861514978	1.122086024815	1.004204513092
29 of 31	1.793624807090	0.936865270807	-2.294183894350
30 of 31	1.611160945374	2.255153023981	1.343574294071
31 of 31	1.566552946450	1.729939599231	-0.709990455000
$k = 32$			
1 of 32	2.420620845284	1.226948572469	0.727239092536
2 of 32	0.780341077326	0.854341387854	-2.666374561677
3 of 32	0.822371594428	0.697507120848	1.414801326910
4 of 32	1.224237963631	1.809788207359	2.154290556386
5 of 32	1.634437359111	0.752942496193	-2.094668907094
6 of 32	1.897033325380	1.181330923445	1.768821309659
7 of 32	1.343685253434	2.068646033864	-0.551226835228
8 of 32	2.067407317619	1.805685325976	2.578859088556
9 of 32	1.535487866712	0.937125873468	0.527477147599
10 of 32	1.960113806080	2.347521994849	-2.186135163464
11 of 32	1.895236498286	1.481833289728	-2.741453695401
12 of 32	1.462895353787	1.089850005552	2.761317512850
13 of 32	2.453706914290	0.682935176449	2.903536260804
14 of 32	2.778791839336	2.864433292659	2.829533429444
15 of 32	2.084507661190	0.635088561340	-0.621981061344
16 of 32	1.100669675099	1.941959507691	-2.949976964247
17 of 32	1.115858481330	2.562626943004	0.802284586894
18 of 32	2.014727341604	2.179674872811	1.466901871913
19 of 32	2.342361353360	1.377977709014	-1.746859441848
20 of 32	1.565126870220	2.627001443438	2.629190745913

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
21 of 32	1.297356508886	1.622744410319	1.178986658449
22 of 32	0.664167032502	1.564556894908	-1.200769653019
23 of 32	1.934044094291	2.763764669442	-0.147712409601
24 of 32	1.095511810310	2.602148886298	-1.762027471020
25 of 32	1.343216554627	1.653335094728	-1.997451899156
26 of 32	0.339425429383	2.298459124665	2.251040199628
27 of 32	1.139497658352	0.561629826013	-0.714047668522
28 of 32	1.644752211618	0.266610807560	1.960515650980
29 of 32	1.577603737514	1.351740890685	-1.132464929083
30 of 32	1.723472034075	1.876399255527	0.364441163929
31 of 32	2.230248753435	1.760790424903	-0.529564171514
32 of 32	0.827618446236	1.559926419080	0.231006568212
$k = 33$			
1 of 33	1.227375256493	1.937324854015	1.063904844757
2 of 33	1.827343851845	0.925273397446	-1.221104287266
3 of 33	1.646693112489	0.566981198671	2.059134451384
4 of 33	2.242138642396	0.549176558518	-2.617881952018
5 of 33	1.352296133805	1.174850308143	2.956305713850
6 of 33	2.389594339877	2.668654303020	3.052253223835
7 of 33	0.564131436076	0.596072702842	2.579355354148
8 of 33	1.327908855840	0.526265100220	-2.315393520318
9 of 33	1.827901603705	2.151737577565	1.839049077013
10 of 33	2.167042352421	2.291353104794	0.610253548362
11 of 33	1.339238692786	2.237616412505	-0.009581321262
12 of 33	1.238199788143	0.921874098197	-0.299158720841
13 of 33	0.872710055213	2.470385513501	-1.323659935582
14 of 33	1.526941913188	1.417598069832	-2.035490127218
15 of 33	1.453838685404	3.005227278641	1.735892378979
16 of 33	2.018227873770	1.617690210237	-2.824335938546
17 of 33	2.243128638410	1.530359510973	-0.430156136282
18 of 33	2.418476039933	1.656498233631	-1.733925906074
19 of 33	1.579850766081	2.313897560647	-2.283962613735
20 of 33	0.773839921881	1.170851026934	-1.438554605866
21 of 33	1.957751374280	2.463383852714	-0.972872732433
22 of 33	1.262157628979	1.417466905997	1.960433732622
23 of 33	1.447862221925	2.105724007156	2.854739461217
24 of 33	1.942038030369	1.284969840359	1.316248094869
25 of 33	1.118069804758	0.934907601023	0.986466907283
26 of 33	1.424054522976	1.696560657416	-0.864002391274
27 of 33	1.685234949548	1.456623647317	0.389701865037
28 of 33	0.847846031338	1.771067391560	-2.645048034824
29 of 33	2.857562782942	1.089678196658	1.144791824238
30 of 33	0.528891837770	1.647065257894	0.150433825908
31 of 33	0.671473391280	2.315539724784	2.127710031820
32 of 33	2.037483079898	0.563912281843	0.219845106993
33 of 33	2.141414224079	1.360843704954	2.418277571261

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
$k = 34$			
1 of 34	0.177322875972	2.513849215183	0.010662090129
2 of 34	1.588015457092	1.664232119725	-2.571734860912
3 of 34	0.959924109967	1.394795574636	0.792411858611
4 of 34	1.955218223238	1.470159552742	-0.759512426589
5 of 34	1.152260584405	2.416964464211	1.713405116933
6 of 34	0.822579476269	0.455195712986	-0.274763571817
7 of 34	2.432144854698	1.717098308885	-2.982405647456
8 of 34	2.473092707129	0.987488802148	0.234519472466
9 of 34	1.791491070019	1.756873132993	-1.666590506889
10 of 34	1.666399112298	1.316338905209	2.859942695433
11 of 34	2.298735636909	0.650239385046	2.318790350305
12 of 34	1.042849920519	2.793555641531	-1.998950092361
13 of 34	1.654349898916	0.750897166799	-2.547749098509
14 of 34	1.924982496581	1.932436852461	0.127784611101
15 of 34	2.524533935064	1.967447152253	1.636411973769
16 of 34	1.039649150130	2.001152952394	2.989181549968
17 of 34	2.588391335243	2.377891301999	-0.988708921701
18 of 34	1.387489584814	0.580138305838	2.236255008383
19 of 34	0.891272556874	1.680663661777	-1.823483100950
20 of 34	1.842058463518	2.211281849086	2.515769915936
21 of 34	1.735754930061	0.938768530087	1.049754006252
22 of 34	1.510397130638	1.109058285268	0.011141004822
23 of 34	0.837251622408	1.399232186264	2.003234371746
24 of 34	1.676641743008	2.369508927485	-0.856026703273
25 of 34	2.381168677946	0.922676946716	-1.697653638202
26 of 34	1.906123531991	2.526339219930	-2.369066323105
27 of 34	1.088051711796	2.311573388851	0.174286165139
28 of 34	1.808014910188	0.262711725611	-0.398624339537
29 of 34	1.915047212549	2.778593967824	0.943042849306
30 of 34	1.420910231820	0.963528060039	-1.370160796971
31 of 34	0.782741523219	0.911988580688	-2.908224809959
32 of 34	1.748805548265	1.423151384945	1.924996184352
33 of 34	1.008642805177	1.575024311633	-0.686315794604
34 of 34	1.707138263684	1.862942093499	1.100977589868
$k = 35$			
1 of 35	1.422524358956	0.670885730540	0.320428362167
2 of 35	1.681124119208	1.806722400311	0.253032042528
3 of 35	1.088323429375	2.610534235242	-2.147921207307
4 of 35	2.090613168816	2.204645208300	2.496604132904
5 of 35	1.340838878353	2.402034334642	0.962381608447
6 of 35	1.089373989256	1.474738317297	0.940747536735
7 of 35	0.474770156164	2.616324780820	0.520708321434
8 of 35	2.288532577268	2.239536967920	-0.432060766918
9 of 35	1.705837961927	2.121489820732	-1.447378386347
10 of 35	1.415714165853	0.770904747346	1.769492224009

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
11 of 35	1.788026111457	0.779888130212	-2.098497078665
12 of 35	1.576034728734	2.011671504285	-2.887663526006
13 of 35	0.761092341773	1.457575635375	2.027598795881
14 of 35	2.144044672498	1.280914576130	-3.007835377602
15 of 35	1.350098042842	1.570233488261	-2.103258417761
16 of 35	0.462180225364	0.194776147589	-1.042567339056
17 of 35	2.211793367324	2.198563717050	1.112494987241
18 of 35	1.653301318704	1.570183100744	-0.691631987145
19 of 35	0.816891345968	1.749071187772	-1.275161691303
20 of 35	0.724962349930	1.680921017319	-2.878383817197
21 of 35	2.083829772710	0.152885111852	1.532428059871
22 of 35	1.477329928604	1.475722966173	2.659009458352
23 of 35	1.263440161253	2.363560679694	-0.457745966084
24 of 35	1.835823051737	3.123572383982	-1.975787724137
25 of 35	3.034558352419	0.970788792373	1.187378809511
26 of 35	1.250167968026	0.828016692514	-1.064548559565
27 of 35	1.944128862845	1.204448640072	1.032549518249
28 of 35	1.209804672205	2.366847451290	2.388754776725
29 of 35	1.178328544261	0.713187338369	-3.109296362083
30 of 35	2.305242602072	1.352211702343	-1.357376965009
31 of 35	0.939290383120	1.420602202974	-0.126763972637
32 of 35	2.193726668160	1.063460226039	-0.069457294024
33 of 35	2.174222422588	1.196124544165	2.109543162143
34 of 35	1.564079558379	1.765133014982	1.685394699308
35 of 35	2.345919453234	2.245975723567	-2.334284716887
$k = 36$			
1 of 36	1.392334691465	1.580638732130	2.236251787680
2 of 36	1.090325408115	2.134418359983	-2.354371094857
3 of 36	1.670227306316	1.335988602445	-1.748485769854
4 of 36	1.156731945169	1.895665035203	-1.248174459421
5 of 36	1.088760047388	0.048225947684	1.868138866330
6 of 36	1.957539188680	0.375129201806	2.166990497653
7 of 36	1.563846638900	0.788786866338	-2.647320629474
8 of 36	1.811969036179	2.198354735644	1.702716732361
9 of 36	2.402210717861	0.915749503972	-2.219786205596
10 of 36	1.915235356424	1.558518833946	0.044225203962
11 of 36	0.311081731539	2.287223961605	-0.887247181316
12 of 36	1.109448891761	1.861420288234	-0.205468516554
13 of 36	2.174506247354	2.132283569641	-1.766997649593
14 of 36	1.132147795777	1.478741833817	3.131791545116
15 of 36	1.782096802851	1.543444877017	0.963269207027
16 of 36	0.586211130807	1.098205597018	2.170958185982
17 of 36	1.553378331070	2.334154538797	0.566660198563
18 of 36	1.988056405754	1.253577114189	2.839609589322
19 of 36	2.297103361280	1.335939924081	1.782575480541
20 of 36	1.818466491837	2.280666680754	-0.624927160066

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
21 of 36	1.963921343377	3.042791087968	3.083533388222
22 of 36	2.545090314351	0.726983505925	0.236405180978
23 of 36	1.033735868836	1.886187368355	1.335966687600
24 of 36	2.140687435031	1.405834680071	-0.920399150982
25 of 36	0.842198024781	1.111949278549	-2.047061367959
26 of 36	1.142852033835	2.859858085585	-0.906634933669
27 of 36	1.722075981719	2.150235370584	2.880954105724
28 of 36	0.891811136131	2.377443298505	2.539092187359
29 of 36	2.435530387338	2.242194990364	0.707183177301
30 of 36	0.833834872421	1.087298492662	0.546274016444
31 of 36	1.815710186706	1.650213997504	-2.591077675677
32 of 36	1.037319394141	0.986918334623	-0.758164894507
33 of 36	1.659312368916	0.737508614303	0.404479618254
34 of 36	2.609274365865	2.164154618315	2.998591467753
35 of 36	1.769821797139	0.543873664352	-1.186224943452
36 of 36	1.364107493460	0.952858308547	1.531871422488
$k = 37$			
1 of 37	1.428403613434	0.987901328949	-1.906538308519
2 of 37	2.380084242906	1.938560066241	0.770398751878
3 of 37	2.117218517645	1.053899246054	-2.606881222645
4 of 37	2.095324795290	1.823973065298	-1.858424926345
5 of 37	2.128267728739	0.972925297916	-1.211485994956
6 of 37	1.613283653009	1.480715853954	-0.340344699054
7 of 37	1.345141551303	0.761187965612	-3.081706962510
8 of 37	2.178967685394	0.997811283639	2.381727713346
9 of 37	1.362505624588	0.669819810826	-0.709129052074
10 of 37	1.475818191722	1.606981578905	-1.221225770184
11 of 37	0.846208742491	1.207750168716	0.113149481842
12 of 37	0.985791434405	2.129356881508	-0.576315025073
13 of 37	1.968247624134	1.970003144288	2.138848218554
14 of 37	2.328402330624	1.985102384644	-0.698974554130
15 of 37	2.183683787559	0.896744259747	0.174946966659
16 of 37	0.907751028896	2.321451303274	-2.016792464004
17 of 37	0.661780798075	0.375729986395	1.970019508322
18 of 37	1.583454130794	2.156411880549	1.216219996890
19 of 37	1.323210475365	1.155963953479	2.157276495008
20 of 37	2.043910079229	2.926344623818	1.646480427115
21 of 37	1.079007390462	2.092464797356	2.134436483587
22 of 37	0.989817736499	1.453798897390	1.257005956252
23 of 37	1.856967129905	1.318331126771	1.412353455493
24 of 37	0.304833135076	2.574030821595	1.066389469513
25 of 37	0.768616358738	1.564661993636	-3.140782015599
26 of 37	2.954485721996	0.327221026374	-2.100649957328
27 of 37	1.633551097943	1.568914052439	2.912115002007
28 of 37	1.483947717343	2.415373119099	-3.106148540372
29 of 37	1.659869181226	2.499486621780	-1.259606881020

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
30 of 37	1.702429932250	2.290796689692	0.090597139109
31 of 37	1.167559781401	2.952055424874	0.875335336626
32 of 37	1.917532002705	0.075252863452	-2.729387152433
33 of 37	0.658221779648	1.200334930164	-1.339290767292
34 of 37	1.413913229549	0.675114781880	0.843867061766
35 of 37	2.325666926209	2.106061841723	-3.024587683427
36 of 37	1.460655955946	1.682489835429	-2.503460902187
37 of 37	1.591428348182	1.516694393990	0.561700500036
$k=38$			
1 of 38	1.774358760528	0.648429498985	0.053935067541
2 of 38	1.488419033700	2.261026113474	2.385392293036
3 of 38	1.802460296924	1.839959642933	0.312446803244
4 of 38	1.015455794405	0.359924495384	-0.719803924324
5 of 38	1.915600556151	1.701581936923	-1.305252063559
6 of 38	0.833559510990	1.824215603664	2.941715062813
7 of 38	0.344182335682	1.422650452559	-0.625158086655
8 of 38	1.463958624356	1.147304493541	2.821538683628
9 of 38	2.159668015444	1.198608610678	0.932384305894
10 of 38	2.067386083329	0.986980764071	2.087195196265
11 of 38	0.655411056384	0.787814586185	2.022031549004
12 of 38	0.846372534236	1.943613594310	-2.064315422040
13 of 38	1.601838042111	1.787831516561	-2.865310105219
14 of 38	2.021610671795	2.404820685984	-0.552152698963
15 of 38	2.527133193839	0.085890265991	-0.316742224421
16 of 38	1.337966922377	1.474960747920	1.969599110637
17 of 38	2.053804956832	1.911872204150	1.683777648881
18 of 38	0.799298354341	2.211249619888	1.597632425179
19 of 38	1.667821227040	2.273144839410	-2.033393030888
20 of 38	1.430583238148	0.797356104771	1.314489684929
21 of 38	1.852207825057	0.777085616333	-1.403491592055
22 of 38	1.569632438075	2.528289935162	1.093566650159
23 of 38	1.004542110425	2.193535548909	0.169219262133
24 of 38	2.262346555431	1.383305776036	-0.412363880697
25 of 38	2.497801238496	1.950009797477	-2.155768427595
26 of 38	2.728192644363	2.342500569784	0.696984831078
27 of 38	1.238443840653	2.208843101671	-1.030735791404
28 of 38	0.955470582169	0.920696013499	-2.585786592837
29 of 38	1.116759532395	2.942639459850	-2.627212070079
30 of 38	2.096661995433	1.023347457418	-2.757915705834
31 of 38	1.418469544853	1.661084812707	1.098622805049
32 of 38	2.086008967307	2.865845834009	2.987776751749
33 of 38	1.397365601753	1.527292762415	-0.411855616930
34 of 38	1.692133057498	0.241790728119	2.938553280411
35 of 38	1.572372402790	1.402479342707	-2.074605242164
36 of 38	1.027847368960	1.172137116702	0.394528482282
37 of 38	1.135626563669	1.294392105567	-1.275918081783

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
38 of 38	2.264807801050	1.810604757319	2.771176444601
$k = 39$			
1 of 39	2.154756343752	2.885624575636	2.685243066586
2 of 39	2.152030873692	1.154051629490	-2.105367685253
3 of 39	1.612068320109	1.562983773261	1.644684111915
4 of 39	1.656527810835	2.462251194046	0.367645771267
5 of 39	1.518314999159	2.294984025627	2.229288489343
6 of 39	2.212009363801	1.208090985317	0.207915922078
7 of 39	0.895900492731	1.633267096865	2.171268802108
8 of 39	1.131128621326	3.099146918613	2.748110508022
9 of 39	1.040937934629	2.118212284647	-3.109926306947
10 of 39	2.111059289101	1.322444113204	-1.014276346225
11 of 39	2.199036346919	0.894373421887	2.929662206836
12 of 39	3.062920312947	0.614964253900	3.011153459173
13 of 39	2.260263958431	1.806241863140	2.236637793233
14 of 39	1.634878799947	1.752928215696	-0.248631611500
15 of 39	1.545046067760	0.514258377807	-2.576654641654
16 of 39	1.593149894515	1.704172977720	-1.648579168804
17 of 39	1.093759534780	1.549918535989	-0.925181051475
18 of 39	1.467028909030	0.722723746201	-0.894671296967
19 of 39	1.090388080248	2.165109221108	1.223260689545
20 of 39	1.424225938000	0.796137974394	2.154203937863
21 of 39	0.857956837915	0.075608565912	0.222757888281
22 of 39	1.574899877375	1.522287655845	2.749354627214
23 of 39	0.973563997521	1.152177285860	1.158919034368
24 of 39	1.503701475734	2.354993627801	-0.973614317831
25 of 39	0.878471867007	2.179803862989	-0.056141792837
26 of 39	2.236441381339	0.924353686623	1.470230327541
27 of 39	1.459277738981	1.550296482904	-2.499453457295
28 of 39	2.239227896880	1.976934031508	1.038362661427
29 of 39	0.836490429501	2.146207780105	-1.818391083264
30 of 39	2.349002631168	2.279473850901	-0.465864620161
31 of 39	0.919791712255	1.066800098164	-3.098289374967
32 of 39	2.178086252492	1.887897220765	-2.894114930457
33 of 39	0.043754872193	1.800398536626	1.334167908210
34 of 39	1.054498008108	1.129599421827	0.012247580436
35 of 39	1.004001923663	1.051451231027	-1.847967848902
36 of 39	1.614750064983	2.386234361890	-2.276339707536
37 of 39	2.216189485113	0.255024499055	-0.900695287678
38 of 39	1.582210306121	0.592372433845	0.680872546922
39 of 39	1.530063627249	1.628031135932	0.610829478632
$k = 40$			
1 of 40	0.858877361525	0.746219810255	2.872456216595
2 of 40	2.655397161159	1.102379481670	-0.967246999983
3 of 40	0.691865410502	2.314120780054	-2.024256029274
4 of 40	1.587061096902	0.829739840443	-2.693256567176

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
5 of 40	2.015346405831	0.261488888836	-1.446848879972
6 of 40	1.121015920636	1.140314372593	0.373451483440
7 of 40	1.771182593670	0.652445547115	2.300061365245
8 of 40	2.033737203897	2.213562480222	2.747159781286
9 of 40	2.054750043465	1.496407564670	2.033752419708
10 of 40	1.024034527430	1.673869658390	-1.014109218396
11 of 40	1.954884898009	1.742199223803	-2.619871916333
12 of 40	1.567208749691	1.026309687650	1.238002129713
13 of 40	1.938608583606	3.039512567225	1.060150163546
14 of 40	1.483450961602	2.483396971554	-2.623044906767
15 of 40	1.886282779277	1.659553325084	-1.031549371839
16 of 40	1.420590629193	0.854724988315	-0.901642621331
17 of 40	1.066827069547	1.114691553656	-1.878967367643
18 of 40	1.100346971907	1.774147213377	1.126834092281
19 of 40	2.029224571559	1.750634518946	1.010190252855
20 of 40	1.119118464117	1.582275844925	-2.771569481687
21 of 40	1.257022383332	0.226396850931	0.651902678966
22 of 40	2.406448062067	0.759532867356	1.198094874177
23 of 40	2.236164593140	2.167759738706	-0.214677669711
24 of 40	1.669099799053	1.397937418824	2.848630251649
25 of 40	1.072564063220	2.910235851250	1.531341957428
26 of 40	2.428750519032	0.941638717306	-3.102354284502
27 of 40	1.201190847586	1.368525730532	2.023956387778
28 of 40	1.910740914673	0.949926005283	0.022864881764
29 of 40	0.726753624570	2.089587103016	0.061422263026
30 of 40	1.501679152298	1.680423040132	-0.244351872789
31 of 40	1.528619341820	2.284102280979	0.448911202791
32 of 40	0.512409502039	0.637492052485	-0.653643456776
33 of 40	2.298763668471	2.298878538939	-1.735528417874
34 of 40	1.501305263597	2.416840119501	-0.797332457641
35 of 40	1.183922424690	2.095923360918	2.630978149588
36 of 40	0.410143463416	1.906263381546	1.999809977415
37 of 40	1.979309711826	1.191824563973	-1.842866991246
38 of 40	2.727407412126	2.396822298144	1.615233439787
39 of 40	1.632671679590	2.232591854166	1.724711592993
40 of 40	1.480117461858	1.910325509297	-1.781428660513
$k=41$			
1 of 41	1.507312268696	2.355838274312	2.753720421982
2 of 41	1.654807079489	0.957990250813	2.051399241717
3 of 41	1.962784046296	1.420793708917	-1.064935704177
4 of 41	1.403123897630	3.065100611696	-0.438134390567
5 of 41	1.444053131189	0.704994797676	0.941247197570
6 of 41	0.854381418383	2.374011234180	-1.991294916579
7 of 41	1.250702939121	1.722249497359	2.104349941447
8 of 41	1.564171881956	1.624831218433	-0.311271955582
9 of 41	1.330160917283	1.489235541546	0.510641933146

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
10 of 41	2.308856793691	1.991151499336	0.978153848664
11 of 41	1.713071027187	1.029073271437	-2.607020860768
12 of 41	0.091982472231	0.995596145797	-1.206009927571
13 of 41	1.641304640791	1.487322218652	1.317602967341
14 of 41	3.045589253831	2.487826890269	-2.481993623360
15 of 41	2.317090997796	0.497155932299	-1.352361114667
16 of 41	0.854901518134	2.100369582719	-0.063922523925
17 of 41	1.420475154329	1.836118325149	-2.577324592137
18 of 41	2.348791926264	1.326484620357	-0.064812383702
19 of 41	1.557651136424	1.397120852240	2.854733636912
20 of 41	1.655569806960	2.265663198363	0.309996174064
21 of 41	0.909073705306	1.133708019284	-2.269388814061
22 of 41	1.640994138010	0.214404714804	3.069580996818
23 of 41	1.021029946847	1.357670150738	-0.962905309969
24 of 41	1.386092205811	2.197569547691	-0.981542183405
25 of 41	1.436779233856	1.653747841245	-1.716250014790
26 of 41	0.771696093248	2.502290954363	1.611720475377
27 of 41	1.830843248043	2.392295389951	-1.955302398336
28 of 41	0.955140230356	0.741554272401	2.698019558414
29 of 41	2.132903621474	1.789420291372	2.133564622774
30 of 41	2.208487882112	2.314421609682	-0.690897605490
31 of 41	2.272970074055	0.841992883631	1.211045868159
32 of 41	2.161876999541	1.920743433099	-3.043008135718
33 of 41	1.669634443274	0.774058803208	-0.334111786459
34 of 41	2.201109024189	2.885683230763	2.382962805608
35 of 41	1.516159904493	0.808742188376	-1.559418597297
36 of 41	0.806761317801	1.276027932181	1.339368119405
37 of 41	1.611544284621	2.324593201912	1.520190081500
38 of 41	0.922370316536	0.340478205031	-0.562882761459
39 of 41	2.311435480040	0.892473036492	2.830563243761
40 of 41	2.238711778100	1.535955016656	-2.018542400414
41 of 41	0.804049762626	1.832342359078	2.995355103897
$k = 42$			
1 of 42	1.221700433576	1.505683744463	-2.678152150411
2 of 42	1.198548324101	1.719183957214	1.854335956893
3 of 42	1.419078795626	1.005642018825	-1.931169674371
4 of 42	1.640536028701	2.216589110501	1.243150389942
5 of 42	1.697335625007	0.825845429190	-2.945637531737
6 of 42	2.153723966175	1.349202589635	-2.275715199627
7 of 42	0.596643208154	0.749650368356	-0.288618730730
8 of 42	1.385881416326	1.634513551143	-0.305975935861
9 of 42	1.206569323234	0.105466990913	-2.460236778949
10 of 42	2.736423276551	1.035918889737	-1.137004597314
11 of 42	0.971674402421	0.994823772738	2.774513509410
12 of 42	2.113940388115	1.729763010860	1.880909669564
13 of 42	0.959701788256	0.865137977672	1.427836186948

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
14 of 42	1.797121047368	1.765149746609	0.441477029223
15 of 42	1.517248005990	2.137755475606	-1.037257143911
16 of 42	2.036604838452	0.989723824148	-0.164491362697
17 of 42	2.238366414158	2.825602400554	0.707015337826
18 of 42	2.262008341459	2.266832759615	-1.678416387266
19 of 42	0.921114983318	1.542626519630	-1.349313871890
20 of 42	2.616868534158	2.098112690305	2.946950897909
21 of 42	1.767379302341	2.442688132486	2.427221076509
22 of 42	2.390483620195	0.655681082145	2.496977367137
23 of 42	0.961360805892	1.785406900768	0.866278041588
24 of 42	1.090308323225	1.991242311666	2.793334516801
25 of 42	1.842629705515	2.092933576724	-2.769467855005
26 of 42	1.827632082759	0.526647478506	1.027510701341
27 of 42	1.380610031880	0.880798395797	-0.852905591800
28 of 42	1.525211303622	1.855886437170	-1.931913575070
29 of 42	0.827882810902	2.341036064896	-2.271044766910
30 of 42	0.680165969783	2.279643605626	-0.324415990243
31 of 42	1.314742297818	1.058419760314	0.377076186626
32 of 42	1.022440809601	2.762826872101	1.758694314097
33 of 42	1.585513586007	2.807820539362	-2.022167638101
34 of 42	1.993856871955	0.406220100481	-1.531187563317
35 of 42	1.481741154078	2.503919590544	0.067432921518
36 of 42	0.340821561038	1.835874403243	2.043502548641
37 of 42	1.925493777955	1.416254897541	-1.296351591319
38 of 42	1.635346516725	1.008686436861	2.096828963261
39 of 42	2.458800839081	1.276689125854	0.896016540460
40 of 42	1.646999762479	1.362987845863	1.202314909177
41 of 42	1.870718268942	1.552468188734	2.771061129551
42 of 42	2.176753986867	1.948833957556	-0.423982013685
$k = 43$			
1 of 43	1.310756034743	1.347019672169	-0.334832102162
2 of 43	2.141558094669	2.561513687859	0.280949774661
3 of 43	2.004927930721	0.847820192498	-0.291777048092
4 of 43	0.778140302530	0.798547179570	1.575245021407
5 of 43	0.971997074746	2.354779573289	-2.263111956830
6 of 43	1.926271741242	1.775873088817	2.845742811695
7 of 43	1.976888580639	2.509198923933	2.153651658196
8 of 43	1.113599855298	0.510014053026	-2.976257952077
9 of 43	1.252523155935	2.290639779713	2.743361154718
10 of 43	0.389908769957	1.617471321524	3.080051249883
11 of 43	1.445529128710	2.281114033735	1.007050355121
12 of 43	1.408339446340	3.128432612549	-0.134248896242
13 of 43	2.532307093891	0.567428271995	-2.321586853027
14 of 43	1.388831366097	2.253809770687	-0.135162915408
15 of 43	1.827695276027	2.409860996536	-2.636063706572
16 of 43	1.854149197543	1.593814090632	0.402112635748

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
17 of 43	0.871589937806	1.177072630429	-1.906168099260
18 of 43	2.363670719372	1.194948950593	2.160484314575
19 of 43	1.850917980900	0.904640214634	3.044522074489
20 of 43	2.069186220627	0.884053639177	1.004451196192
21 of 43	1.230424723667	1.548700892121	-2.751988441280
22 of 43	1.322004575456	0.788536634538	0.461900212172
23 of 43	1.533771376745	1.758732427257	-1.881503377130
24 of 43	2.051971322437	1.530336304573	-2.542312184582
25 of 43	1.173626526404	1.352858610957	2.634142190980
26 of 43	0.498628378464	1.024069177333	-0.420062876175
27 of 43	0.875443146381	1.965143282844	1.752636931718
28 of 43	0.984655402108	1.914053666497	-1.112916982078
29 of 43	1.812104940019	0.039325074515	2.257823614992
30 of 43	1.008687869110	1.654043352211	0.487458147480
31 of 43	1.637791603395	1.344681882465	-1.144584287754
32 of 43	1.654599254297	1.705098191029	2.010326577188
33 of 43	2.152652307362	1.847083444403	1.269542663542
34 of 43	1.419793133976	1.402051760670	1.250056327854
35 of 43	1.241734294472	0.595670156576	-0.977239622164
36 of 43	2.671708222894	2.301530023384	-3.050960590682
37 of 43	2.030796931536	1.779222898407	-0.486520291294
38 of 43	0.632290772453	2.707446775877	0.111951636130
39 of 43	1.582937962926	0.861353086919	1.969084842699
40 of 43	2.325904672739	1.733998796028	-1.545528547131
41 of 43	1.695151706635	0.840041633337	-2.062947045908
42 of 43	2.716982556117	1.463005994486	0.248483871116
43 of 43	1.721790393279	2.345770625625	-1.201315845436
$k = 44$			
1 of 44	1.389044908176	1.034905901696	-1.722854073939
2 of 44	0.885320203245	2.033880887021	1.772084401771
3 of 44	1.340338597332	1.289551790141	1.343016515850
4 of 44	1.742056400314	1.961940691646	-3.099647463022
5 of 44	2.249406038511	1.100321321519	-0.129265725485
6 of 44	1.348642804606	2.341570032317	-2.302447269385
7 of 44	1.019294784848	2.028579002103	0.608939533814
8 of 44	0.226742907517	2.499774791768	-0.375393143624
9 of 44	1.552955017809	1.484596439422	-0.988384980138
10 of 44	0.947429909907	1.831996425377	-1.549646043598
11 of 44	2.426405548661	2.244932140596	-1.071810994078
12 of 44	1.908760362270	1.522078010843	1.927643680637
13 of 44	1.000847902004	3.072479508630	1.951200883124
14 of 44	2.227754652756	1.106749262608	1.102855202575
15 of 44	1.322243429614	0.924649173865	3.129783409134
16 of 44	2.149945674843	0.080668632791	1.609610587359
17 of 44	2.238646758772	1.114566964034	-1.364939331514
18 of 44	2.983970259504	0.141652492094	1.645258280899

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
19 of 44	2.409035109750	2.307483100949	0.793062184576
20 of 44	0.782996750329	2.054927594786	-3.140092484339
21 of 44	1.977611638928	2.890063549054	-2.567201010214
22 of 44	1.542997221325	2.400342631316	2.319785057228
23 of 44	2.390910766203	2.157142832206	2.512101423062
24 of 44	0.771256350479	0.979646755789	0.611706015136
25 of 44	1.570317699460	2.399774872664	-1.100705778715
26 of 44	1.782654915105	1.683293080278	0.540946461405
27 of 44	0.736785947742	0.943518950497	2.222552935134
28 of 44	1.022004900776	1.954449224298	-0.483746674349
29 of 44	1.965613583307	0.881120840338	-2.459095136331
30 of 44	1.760556262895	1.785993574458	-1.747841138047
31 of 44	1.386941690717	1.250594172838	-0.067098515479
32 of 44	0.866975629727	0.978600907653	-0.843218587903
33 of 44	1.648251432592	2.593037999901	0.343044114951
34 of 44	2.364755543342	1.790558369200	-2.440573018055
35 of 44	1.631772076659	2.076790457746	1.316267402522
36 of 44	1.648731818952	0.657746355264	-0.748510713955
37 of 44	1.331427447290	1.587987252832	2.536112725194
38 of 44	1.559353795939	0.671737142011	0.676690090319
39 of 44	1.841112246173	1.915400180172	-0.311720453587
40 of 44	1.132253576614	0.071217166321	-2.518904812320
41 of 44	0.675421770549	0.996038362010	-2.347802031021
42 of 44	1.397572600264	1.504895140175	-2.473288940516
43 of 44	1.596305128117	0.738494638097	2.052048141866
44 of 44	2.062028613024	1.197784773471	2.795013913650
$k = 45$			
1 of 45	1.583433839062	1.767319479752	-0.559205055131
2 of 45	1.137776838183	0.909153847428	2.426371552416
3 of 45	1.448771210592	1.653778900462	-1.379502632835
4 of 45	2.076541400778	2.217779822396	0.035080798940
5 of 45	0.307867172492	2.575787081161	0.026681991297
6 of 45	1.901861738652	0.490923210573	-1.762029906834
7 of 45	0.525114800713	0.643377063897	-2.717948278141
8 of 45	2.324381972924	1.358981251050	-0.507099052468
9 of 45	0.992933860196	2.105880500121	1.183199981896
10 of 45	2.198240053587	1.367695002646	-1.619426538770
11 of 45	2.009059600119	2.960198360261	-2.772522292057
12 of 45	2.790803244886	2.767118703358	-0.882532961970
13 of 45	0.853533999631	1.400029345515	-0.782333024160
14 of 45	2.656085016446	0.192445336372	3.138170127978
15 of 45	1.440314301299	0.846959193016	-2.788267889433
16 of 45	2.029073116206	2.233815776813	-1.180640810109
17 of 45	1.878362984841	0.530934945485	2.430269741821
18 of 45	1.358058304377	1.502020819080	1.764431538793
19 of 45	2.434767925594	1.415051063678	0.749970187624

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
20 of 45	2.010216847337	0.591002736095	0.211459473983
21 of 45	1.611765247972	2.220811056674	3.137512347277
22 of 45	1.469798667496	2.285515194470	2.067119415670
23 of 45	2.128475432029	1.236061743589	-3.120736228846
24 of 45	1.098049538186	2.369885112556	-0.970511508842
25 of 45	1.503569157686	2.322468045517	-2.054812116097
26 of 45	1.649863114970	1.606665719663	0.992326026094
27 of 45	1.047409837428	1.878050883763	0.117855522884
28 of 45	1.176859589644	0.896721864560	-1.698439395808
29 of 45	2.093709002298	2.290647138792	1.307214418928
30 of 45	1.437038851148	2.708747239033	0.436372851375
31 of 45	0.800406847268	1.835944606529	-1.949647287478
32 of 45	2.313589607646	2.096238510013	2.564222097632
33 of 45	1.224507208902	0.022413742554	-3.013523932658
34 of 45	2.218360691911	2.021327990311	-2.460568181057
35 of 45	1.049357618360	1.611219799279	-2.951159378194
36 of 45	0.715382373947	1.043837116342	1.304537099113
37 of 45	0.750643729978	1.868209820843	2.332824802274
38 of 45	1.618371505048	1.593827681947	2.549909424055
39 of 45	2.157595093014	1.295861652283	1.850233541306
40 of 45	1.109594528219	0.874151331992	0.133253026299
41 of 45	1.562054820659	1.492323725002	-2.206546565409
42 of 45	1.744737009330	1.418007455013	0.182271469420
43 of 45	0.993611610189	2.802695483391	-3.100486869435
44 of 45	1.519693915972	0.822243105647	1.248026727020
45 of 45	1.579337001333	0.962728318951	-0.762070447981
$k = 46$			
1 of 46	1.711877721067	0.803146833565	1.003696792141
2 of 46	2.395332409774	1.472392662014	-1.790154231901
3 of 46	1.506419825642	0.050503570602	2.399049817358
4 of 46	1.398646190452	2.170441256899	1.071092364282
5 of 46	1.119744898334	1.502941212103	-2.693132797946
6 of 46	1.149456124722	2.383702179648	-2.435343111678
7 of 46	1.437681874393	1.342323250593	2.335125328806
8 of 46	0.330781990262	2.427401454805	-2.353957315283
9 of 46	0.768119925605	0.370956258629	-1.506300951345
10 of 46	1.727683154879	2.085463708113	-0.656171929998
11 of 46	1.766802338389	1.843345669766	-2.299890672058
12 of 46	1.661157626728	1.728539186006	-1.450971534897
13 of 46	2.052471115644	1.933038622896	0.557951712959
14 of 46	2.237593019553	1.414588989389	1.393783121728
15 of 46	1.263955610848	0.761756972407	-0.002889293699
16 of 46	1.507271964132	0.775700496925	-1.190748240608
17 of 46	2.941000396117	0.698252087701	2.345027056560
18 of 46	1.618054258226	2.078916596124	2.008088539822
19 of 46	1.000762488619	2.942403229096	1.323934980479

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
20 of 46	0.687029657149	1.877894204606	1.518000208253
21 of 46	2.210391022888	1.146002011831	-2.894374128384
22 of 46	0.711899257278	1.068353229107	2.724466721740
23 of 46	0.996352954486	2.120588356978	2.664197931557
24 of 46	0.570893224387	1.348098265502	0.064348695847
25 of 46	2.518435803501	2.328416836838	-0.509454492647
26 of 46	1.010586979611	2.296167787955	-1.099765760698
27 of 46	2.096530953143	0.646682610871	2.199927970968
28 of 46	1.743522231622	2.732920531134	0.217497335676
29 of 46	2.390232460421	2.249709094779	-2.853226861652
30 of 46	1.628230113980	1.034302902330	-2.193613155985
31 of 46	0.930088622799	1.459796932731	-1.743707576199
32 of 46	1.657565259686	1.488255118202	-0.065239156904
33 of 46	1.261209053094	1.430212629309	0.664461645151
34 of 46	1.686273369314	1.860186009691	3.089535562554
35 of 46	1.424232857053	1.445369770354	1.499172872307
36 of 46	1.468476328681	0.859856797991	3.103620399232
37 of 46	2.229723758708	0.375575560244	-1.437163938076
38 of 46	2.303803344721	2.492540675835	1.603804605154
39 of 46	1.717381616925	2.693257075303	2.874616638950
40 of 46	1.160235993506	2.144041705110	0.069350485107
41 of 46	2.211297843685	1.614671454796	2.432085213951
42 of 46	2.315443330890	1.013667255233	0.235785363501
43 of 46	1.162009625730	1.438962552639	-0.736826147765
44 of 46	0.913594503492	0.759395771205	1.279937584514
45 of 46	2.070117816759	1.303942195925	-0.803353143757
46 of 46	1.923001728762	2.512552240103	-1.648589066284
$k = 47$			
1 of 47	2.775704400014	0.435129423301	-1.825930518981
2 of 47	2.023031775407	0.928668449474	-2.161490765535
3 of 47	1.524137226860	0.660773781661	-3.042749100762
4 of 47	1.541666000960	2.086841733120	-1.981167504331
5 of 47	2.680743886289	2.518173553780	1.842475192900
6 of 47	1.087619542274	2.155739628860	-1.103935709569
7 of 47	0.744165057767	1.211361769874	-1.289307836505
8 of 47	1.410931539835	2.023912779784	0.870292212446
9 of 47	2.244500492051	1.072247968775	2.707521640474
10 of 47	2.273993478821	1.776297385135	-1.721593514742
11 of 47	1.705156643441	0.726823509648	1.939444516302
12 of 47	2.236362286359	0.935527021479	0.208017386359
13 of 47	0.758392067013	0.891503521101	-2.716923204999
14 of 47	1.868840754086	2.396664253713	-1.074674990131
15 of 47	1.934218322732	2.027544706708	1.618639144425
16 of 47	1.861178166622	2.770249712673	0.879636268943
17 of 47	1.543564035086	1.507623096769	2.150947469379
18 of 47	0.928353363934	1.625021230104	-0.245561811774

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
19 of 47	0.832332784541	0.293179795757	0.122006741871
20 of 47	0.807203889733	1.835034407467	2.764569637733
21 of 47	1.402580287323	0.908975115283	-0.428632300777
22 of 47	0.907062949930	1.316826419888	0.763104920754
23 of 47	1.359718300030	0.619136938371	-1.609258534157
24 of 47	2.131607608571	1.685768936466	-2.746118558946
25 of 47	1.575063573687	1.452568432802	0.263842549487
26 of 47	2.451694978855	1.854383162070	-0.506518357131
27 of 47	1.419940398202	2.181318140750	-2.945391112864
28 of 47	2.191824239950	1.884993892555	0.646310511528
29 of 47	1.402309380623	2.459246873794	2.221386888134
30 of 47	1.410343280403	1.415937507750	-2.451626268069
31 of 47	1.635947829800	1.649834768362	-0.706277524748
32 of 47	2.077883334562	0.985192749929	-1.008490227014
33 of 47	0.749715554955	2.059304871882	-2.235274741377
34 of 47	0.953760880689	1.862183912414	1.652210492677
35 of 47	0.102386841211	1.501146699928	1.177506012586
36 of 47	2.340974875445	1.180091350545	1.469612147134
37 of 47	2.078426763099	2.637634368015	-2.627586192320
38 of 47	1.524479509740	1.446206100101	2.968595710088
39 of 47	1.539997685391	2.207158712601	-0.064589361870
40 of 47	2.028697742741	2.015028740371	2.617182646990
41 of 47	1.515519825016	0.724659583597	0.693667051174
42 of 47	1.327232268870	2.883166563801	-1.998489649073
43 of 47	1.966144670317	0.080377031780	-0.963638588928
44 of 47	0.842166588410	2.584664702024	0.361359768620
45 of 47	0.922545702462	0.933961000337	2.119033613817
46 of 47	1.521356252026	1.421171070571	-1.490641516750
47 of 47	1.548465943473	1.349420556292	1.327772894073
$k = 48$			
1 of 48	2.213338359892	2.070949852744	2.752955914569
2 of 48	1.981922258437	1.906682736152	-2.541081308423
3 of 48	0.829754351151	1.108389153337	-0.781210456669
4 of 48	1.624886625223	2.734583054273	0.243247885337
5 of 48	1.581309467795	1.586886050746	3.076844835706
6 of 48	1.759566125962	2.676348912372	-2.130034266603
7 of 48	1.255074975237	2.176198944596	-1.475327431930
8 of 48	2.570506843712	1.656879036488	1.574297880422
9 of 48	1.060341775194	1.405881830668	0.207034578909
10 of 48	2.709981491725	1.824348582595	-1.971469329375
11 of 48	2.129048922009	0.886085325764	-1.862473910566
12 of 48	1.510347315119	0.782801588795	3.001481068548
13 of 48	1.721717032830	1.577233075386	0.662342532603
14 of 48	1.770971344879	1.147477491505	-0.115799467654
15 of 48	1.138256788955	2.144268385594	0.768624421511
16 of 48	2.248418802269	1.360361010549	-0.862448002410

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
17 of 48	2.262665950480	1.137105898371	-3.034386276644
18 of 48	1.252244194838	1.974673666559	1.741878986957
19 of 48	1.126436720852	0.509979047351	-2.060374516102
20 of 48	2.382423685975	2.995305803911	1.092726788633
21 of 48	1.245226451210	0.540961879454	0.093355895825
22 of 48	2.103278601601	0.801875472263	2.087068241620
23 of 48	1.545750590204	1.211707173269	-2.441588489006
24 of 48	1.813186290944	0.070138985010	-2.120268593120
25 of 48	2.007788736524	2.285900997750	-0.858633502882
26 of 48	0.526930660406	0.292031624879	1.366696666238
27 of 48	1.341455519464	0.635096893826	1.733672197834
28 of 48	2.580454653814	0.299689355427	0.017240631534
29 of 48	1.903127639888	2.194668090201	1.235499941687
30 of 48	1.928766653654	1.746375577298	-1.647905877611
31 of 48	2.398625594391	1.811315355930	0.181236070240
32 of 48	1.181304269671	2.006928056986	-2.463666206684
33 of 48	0.993725251537	1.312766768023	1.179828239647
34 of 48	1.483241910483	1.644657018023	-0.835492433362
35 of 48	0.485838190225	1.953042805307	-1.799173321963
36 of 48	0.933576218343	1.297855105800	2.268448996360
37 of 48	1.755359046443	1.385437583884	1.465378409671
38 of 48	1.223518096228	1.334709644806	-1.654563096478
39 of 48	1.592456847089	0.858004784953	-1.011607145988
40 of 48	1.898592836663	0.772566019693	0.814469437068
41 of 48	1.022934425601	2.144890102918	2.792353760019
42 of 48	1.631055486210	2.590595749012	2.282468448416
43 of 48	1.621351393884	1.949476489503	-0.073745273774
44 of 48	0.832627483925	1.230545015195	-2.896457174903
45 of 48	1.710070262364	1.541564802387	2.275859748322
46 of 48	0.465572815733	2.229689002856	1.612641813103
47 of 48	0.979637742117	2.999727186602	-2.141618499905
48 of 48	0.895130696888	2.149016200901	-0.449894914271
$k = 49$			
1 of 49	1.717208909762	1.232324077200	3.048150181993
2 of 49	0.709902865058	1.908759435541	-2.422298206893
3 of 49	1.568537960055	1.599243050222	2.314341114458
4 of 49	1.021077436845	0.909915141402	-2.548994857723
5 of 49	1.557420316169	2.193331498517	-1.953518787899
6 of 49	1.172968827620	2.691351168919	-2.887029069891
7 of 49	2.285852912303	1.935368920573	2.538196795033
8 of 49	1.699070822337	1.320567304101	0.974785796090
9 of 49	0.483872398446	0.499176617046	2.280290805941
10 of 49	0.989550186510	2.165948389929	2.122907478571
11 of 49	1.573329937639	1.895010433740	1.534926888459
12 of 49	1.796706645987	0.105850657693	3.120553057666
13 of 49	1.593021773744	2.029919812818	3.021266232176

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
14 of 49	1.228737145575	1.460927398544	-1.781788296314
15 of 49	2.282131052475	2.713658527724	0.823352068307
16 of 49	2.267249931426	0.670739223446	0.830313749377
17 of 49	1.563082418728	1.441793273627	0.173674084198
18 of 49	1.662252164540	2.794313364125	-0.954591020446
19 of 49	2.371007530279	0.786848998484	2.668349865750
20 of 49	2.291060415602	1.575866669157	-2.687215124011
21 of 49	1.328549694519	0.605187579525	1.069335929341
22 of 49	2.401143328588	0.615313830418	-1.454110905446
23 of 49	2.110546045758	2.564062387042	-2.691127802531
24 of 49	1.585580043994	1.852210252363	-1.132977057847
25 of 49	0.953394842161	2.379125828177	-1.227737795961
26 of 49	0.581588088662	1.271659960047	-1.150160995100
27 of 49	1.482235753780	1.613709315689	-2.552585149124
28 of 49	2.030111854367	1.524507590368	-1.755596517285
29 of 49	1.645164243083	0.915920850478	-1.298829521970
30 of 49	1.277700221382	1.317692581109	-0.595060086690
31 of 49	1.821406232617	1.002063299464	1.796926616103
32 of 49	2.303217806120	2.255414913047	-1.125732474626
33 of 49	2.077285631214	1.363698944279	-0.672744341354
34 of 49	3.058277903876	2.686399153125	-3.074261767811
35 of 49	0.832995049634	1.030578341817	0.208792019823
36 of 49	1.077893460115	0.339851771315	-1.037949840013
37 of 49	1.269823309927	0.700330346111	2.523415501452
38 of 49	1.716722096454	2.075936248285	0.666436178638
39 of 49	1.693944995551	0.706428122062	-0.187757802756
40 of 49	1.001365246631	2.005651198535	-0.069280118437
41 of 49	1.713594371851	2.533061950567	2.160951006781
42 of 49	2.354363411963	1.666211250805	0.251681396512
43 of 49	1.779503493799	2.079793133302	-0.292795227747
44 of 49	1.003004851006	1.770773902902	0.924385865829
45 of 49	0.966452827729	1.485208097991	2.874127197463
46 of 49	1.175874619233	2.665420096092	0.893718670386
47 of 49	1.811192985371	0.898029928458	-2.351111246472
48 of 49	2.342347350555	1.656052276035	1.419569680332
49 of 49	0.372322825248	2.702793806619	0.816707421441
$k = 50$			
1 of 50	1.717208877855	1.232324063056	3.048150153926
2 of 50	0.709902843389	1.908759415791	-2.422298193787
3 of 50	1.568537932195	1.599243050836	2.314341094170
4 of 50	1.021077418777	0.909915105895	-2.548994863230
5 of 50	1.557420308554	2.193331475729	-1.953518815593
6 of 50	1.172968811101	2.691351158659	-2.887029051397
7 of 50	2.285852881523	1.935368908192	2.538196796959
8 of 50	1.699070836949	1.320567328464	0.974785778079
9 of 50	0.483872388104	0.499176570627	2.280290651045

Table A.2: Coordinates of 4-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$
10 of 50	0.989550165606	2.165948402232	2.122907453388
11 of 50	1.573329932120	1.895010451736	1.534926873424
12 of 50	1.796706645286	0.105850648073	3.120552858193
13 of 50	1.593021741112	2.029919796784	3.021266220765
14 of 50	1.228737144621	1.460927376706	-1.781788303205
15 of 50	2.282131057649	2.713658568608	0.823352111410
16 of 50	2.267249945418	0.670739237625	0.830313766197
17 of 50	1.563082451726	1.441793294329	0.173674064970
18 of 50	1.662252170211	2.794313362576	-0.954591116250
19 of 50	2.371007508700	0.786849016889	2.668349836222
20 of 50	2.291060387289	1.575866647877	-2.687215162975
21 of 50	1.328549703358	0.605187605251	1.069335883295
22 of 50	2.401143336803	0.615313808177	-1.454110961832
23 of 50	2.110546027778	2.564062350040	-2.691127833952
24 of 50	1.585580062479	1.852210244381	-1.132977086069
25 of 50	0.953394851964	2.379125807533	-1.227737809478
26 of 50	0.581588108271	1.271659956009	-1.150160962685
27 of 50	1.482235728395	1.613709290942	-2.552585167734
28 of 50	2.030111854134	1.524507571478	-1.755596554887
29 of 50	1.645164256864	0.915920838072	-1.298829529017
30 of 50	1.277700253040	1.317692587798	-0.595060096863
31 of 50	1.821406222728	1.002063314875	1.796926591264
32 of 50	2.303217819973	2.255414918728	-1.125732547985
33 of 50	2.077285661894	1.363698945616	-0.672744366583
34 of 50	3.058277886276	2.686398784296	-3.074261983772
35 of 50	0.832995078990	1.030578375846	0.208791992966
36 of 50	1.077893470489	0.339851776463	-1.037949749403
37 of 50	1.269823291836	0.700330333442	2.523415444127
38 of 50	1.716722115438	2.075936275177	0.666436160660
39 of 50	1.693945020517	0.706428133414	-0.187757799882
40 of 50	1.001365276936	2.005651204147	-0.069280150596
41 of 50	1.713594355574	2.533061952382	2.160951027468
42 of 50	2.354363443396	1.666211278493	0.251681388847
43 of 50	1.779503522688	2.079793149624	-0.292795261871
44 of 50	1.003004866013	1.770773923707	0.924385825387
45 of 50	0.966452793277	1.485208081157	2.874127172411
46 of 50	1.175874624324	2.665420114063	0.893718627618
47 of 50	1.811192971633	0.898029907098	-2.351111275995
48 of 50	2.342347349502	1.656052298714	1.419569694820
49 of 50	0.372322830706	2.702793783906	0.816707226679
50 of 50	1.063349024003	1.274390602449	1.736519025841

Table A.3: Coordinates of 5-dimensional codes.

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$	$\widetilde{x}_4$
$k=3$				
1 of 3	1.812733106048	2.563080731327	2.177384061881	1.718714008520
2 of 3	1.690383202123	0.479450729959	2.695716111763	0.958938187248
3 of 3	1.203722793590	1.643735281395	0.693553076383	-1.653513867135
$k=4$				
1 of 4	1.524661400074	1.278057995671	1.910370010459	0.149593897931
2 of 4	1.184496066179	1.433205222935	1.658367391257	-1.945849739580
3 of 4	2.579906380674	1.067528093543	1.335074261935	2.673930317918
4 of 4	1.133517703371	2.406861527914	1.073237622875	1.859531903878
$k=5$				
1 of 5	1.255004988629	1.159299114273	2.205077850687	-1.813410017615
2 of 5	1.945327385825	2.440522254999	2.735092758238	0.887587715288
3 of 5	1.538698548498	0.909885685080	1.683625170810	1.529277133067
4 of 5	2.346425939967	1.368589311154	0.496129089425	-1.648955808316
5 of 5	0.762167838095	2.235536140147	0.087156358702	1.309992898468
$k=6$				
1 of 6	1.873936958353	0.591541727772	1.532419007333	1.945981258369
2 of 6	1.658849362709	1.713977778445	1.505153563055	-2.544872489955
3 of 6	1.740071085024	1.311279881269	1.397724150204	-0.767390119504
4 of 6	0.146692933856	1.203935377383	1.383887751770	1.218714722194
5 of 6	1.758958245897	1.943133049183	2.902339208324	0.780152178274
6 of 6	1.820668785271	2.236057953227	0.633361904305	1.271875465063
$k=7$				
1 of 7	1.159966050265	3.141567265330	0.746669037190	-0.026482478489
2 of 7	1.818453268584	1.681143917366	1.284206019216	-0.226561797223
3 of 7	1.603620732327	1.585101086419	1.328170708134	-1.881069842033
4 of 7	0.553200525191	0.787734550900	0.488238271854	-0.036949215153
5 of 7	1.537971921263	1.556491567171	1.813421945456	1.260522811557
6 of 7	1.981626603325	0.000025388264	2.394923430274	3.115110175101
7 of 7	1.808386102829	1.676472058464	0.446403633016	2.261107410170
$k=8$				
1 of 8	1.243631582388	1.095857716205	1.181910192689	-1.260878320107
2 of 8	1.909188689954	1.302956123089	1.437498257789	0.366914477537
3 of 8	1.532064211991	1.433000783853	0.503153371973	2.235475658748
4 of 8	1.897961071201	2.045734937385	1.959682460901	1.880714333482
5 of 8	1.991777661187	2.487309004024	0.872524571126	-1.198780500213
6 of 8	2.475342857248	0.743809771630	1.722989393906	-2.593825930099
7 of 8	0.666249796342	2.397782881960	1.418603259684	0.547766723491
8 of 8	1.149814992403	0.654283649566	2.269068082464	1.942812153377
$k=9$				
1 of 9	1.279051950955	2.010337686267	3.141558938673	-1.501041500166
2 of 9	1.062336704742	2.480794756792	0.436545268863	1.572229348934
3 of 9	1.658970325750	1.445244870024	1.659860312317	2.265655002786
4 of 9	1.862540702635	1.131254967323	0.000033714917	1.640551153423

Table A.3: Coordinates of 5-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$	$\widetilde{x}_4$
5 of 9	1.482622327840	1.696347783566	1.481732341273	-0.875937650804
6 of 9	2.079255948847	0.660797896798	2.705047384727	-1.569363304656
7 of 9	1.642110503006	1.469372609854	1.642561390159	0.669066322201
8 of 9	0.614664002747	0.000106567711	2.573651235625	-2.585208892201
9 of 9	2.526928650842	3.141486085876	0.567941463929	0.556383761389
$k=10$				
1 of 10	1.038220020944	2.145022517702	2.201468013822	-0.793100941406
2 of 10	1.738141325099	2.067195381960	1.412033185496	-2.517895948182
3 of 10	2.112033662136	1.105377613863	1.618469527186	-1.100109707416
4 of 10	1.170971265936	0.860028650397	2.338225934198	-2.796916693880
5 of 10	2.147696822218	1.844637977632	2.764923478786	1.813698185436
6 of 10	2.103372632646	0.996570135887	0.940124639768	2.348491712184
7 of 10	1.403451328491	1.074397271630	1.729559468094	0.623696705407
8 of 10	1.029558991453	2.036215039727	1.523123126404	2.041482946174
9 of 10	1.970621387654	2.281564003193	0.803366719392	0.344675959710
10 of 10	0.993895831372	1.296954675958	0.376669174804	-1.327894468154
$k=11$				
1 of 11	1.905551867075	1.557697236350	0.253892605015	-2.328548312344
2 of 11	1.912953855328	1.555116650286	1.714297285132	-1.476320048054
3 of 11	1.720654349114	0.105075234099	0.945488003346	-1.701075581820
4 of 11	2.861044308218	1.504589744476	2.158553497968	1.474761526497
5 of 11	1.550312278030	1.371175814853	3.136198640362	-1.704753457255
6 of 11	0.953823434673	2.188360260829	0.765947497710	0.828205612044
7 of 11	1.697638192812	2.763396352842	2.657528921281	-2.657520161717
8 of 11	1.475552116728	1.336902166214	1.665598702889	1.853078002415
9 of 11	1.747456180762	1.476981418563	1.692601400733	-2.984793932647
10 of 11	0.574805880542	1.141091579944	1.712211532475	-1.969819653514
11 of 11	1.665973205186	1.431159581644	1.682868459805	0.002285811542
$k=12$				
1 of 12	1.478823506945	1.301500165062	2.463720787901	-1.752310258210
2 of 12	1.232249559483	1.170824103971	0.303192299829	0.535133918718
3 of 12	1.325846294791	1.503550796693	1.746579372986	-0.160295223373
4 of 12	2.224341500148	1.960621450888	0.633259372970	2.322001126586
5 of 12	1.430751293921	1.527024044754	1.164471420949	-2.442110704175
6 of 12	1.582107514464	1.808313907895	2.055153141849	2.724105232951
7 of 12	1.647546960704	0.490029198246	1.984664835812	1.991175259845
8 of 12	1.411316931737	1.749036612780	1.595837861129	1.277554257541
9 of 12	2.337138604236	2.075641255227	2.627996268292	0.388791433539
10 of 12	2.252950967845	0.944162430179	1.161668321268	-1.110956542982
11 of 12	1.494045692885	2.651563455343	1.156927817778	-1.150417393745
12 of 12	0.208087868805	2.285681231055	2.440617966677	3.139469926623
$k=13$				
1 of 13	1.761983662208	1.512851625799	0.996469699975	-2.984309532231
2 of 13	2.070282972491	1.272873154929	2.467891151608	-2.990321768276
3 of 13	2.716682371396	2.578322893574	0.595718029903	1.058649307542
4 of 13	1.555463164188	1.611693834021	1.124403898403	0.829740609429

Table A.3: Coordinates of 5-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$	$\widetilde{x}_4$
5 of 13	1.945847024652	0.782593001201	0.647272921666	-0.757583928596
6 of 13	1.822898278117	1.779118610346	1.598474965363	-1.627120508237
7 of 13	0.779007945504	1.406149498581	1.259977027687	-0.737192033127
8 of 13	1.364827407282	2.612354440205	0.175680966294	-1.627053148347
9 of 13	1.655331514211	2.273703554905	2.746575808493	-0.124198829456
10 of 13	1.595818188930	1.968294681573	1.726565021607	2.111994156592
11 of 13	0.968993080062	0.720695055836	1.231258978514	2.124981141523
12 of 13	0.880693540174	1.885516511483	2.070573100464	-2.703900269864
13 of 13	1.528609870602	0.964344439727	2.329798580551	0.265591685104
$k=14$				
1 of 14	0.412187081282	0.763675991625	0.854237332172	-3.097951056318
2 of 14	2.116430685123	1.232537355117	2.662238309133	0.227341238522
3 of 14	1.204526853497	2.268814039988	0.807596496411	1.189959465212
4 of 14	1.253464709577	2.321460073586	1.430926091379	-2.151381492298
5 of 14	2.427093099195	1.529507954595	1.511089995530	-2.128267508303
6 of 14	1.916878179438	2.199555286221	1.505564225651	-0.470241045398
7 of 14	1.391447388546	1.327686647256	1.858105936214	1.704758653846
8 of 14	2.359327835394	1.464133314896	0.898217442101	1.150423722402
9 of 14	0.990235065574	2.058000745982	2.668434846746	0.188745221907
10 of 14	1.542253520885	1.442850356582	0.457985220951	-1.252110976280
11 of 14	2.010670175696	2.456366431083	2.241161275354	2.276334940849
12 of 14	1.350112001479	1.219070750882	1.861871624833	-1.250906870588
13 of 14	1.438120764467	1.379085638568	2.327766226877	-2.931655451279
14 of 14	1.722585453105	0.220928741228	0.960303962304	3.076568683651
$k=15$				
1 of 15	2.343583333362	1.914360798004	2.147217606006	-2.819222028662
2 of 15	1.325552218539	0.908118331478	2.129637955463	-2.714003728490
3 of 15	1.683811139214	1.365799771141	1.022172788624	-0.009483366174
4 of 15	1.199114707234	0.841242806937	1.014125120362	1.706544736777
5 of 15	1.452711832216	1.674212799900	2.232275014511	1.971779527039
6 of 15	1.106576865077	2.279717398376	2.414090421271	-2.065064276325
7 of 15	1.821235035764	2.373163639754	0.722968406605	-1.426074200148
8 of 15	1.937078672140	2.163140907386	2.216939610259	0.204230252751
9 of 15	1.806126419114	1.386648157531	1.851810847206	-1.215194319729
10 of 15	0.805019076049	1.373330763847	0.849381056256	-1.551400710511
11 of 15	2.040655666770	0.999178186075	0.629124836986	-2.355080857923
12 of 15	0.966583307750	2.315762596612	1.116343760538	0.994685200004
13 of 15	1.340641652035	1.860155500879	1.169457330355	2.964505025628
14 of 15	0.948651291640	1.170440662393	2.310508102091	0.064464285169
15 of 15	2.186758031796	1.839851192336	1.030060092411	1.609129417474
$k=16$				
1 of 16	2.215231386953	2.234271344863	2.750638822732	-2.889375536288
2 of 16	1.517138106974	1.236863328946	0.725011835890	-1.348882960976
3 of 16	1.535127854092	1.391827256133	0.726685520765	1.409613703464
4 of 16	2.263275473096	1.964460923678	1.112742593668	-0.054689019475
5 of 16	1.778083471413	2.249893261252	0.688800853696	-2.929280692007

Table A.3: Coordinates of 5-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$	$\widetilde{x}_4$
6 of 16	1.919105014361	1.072060980844	2.167829355469	-0.983245284175
7 of 16	1.354806046754	2.458606249084	1.701602033110	1.152928995301
8 of 16	1.479336038238	1.439701940577	1.764976845860	-2.426309080456
9 of 16	1.938292708927	1.243288340453	2.085794614396	1.104732043146
10 of 16	1.077248642882	1.360001276762	1.526828609925	0.058515818496
11 of 16	0.436127006313	1.896464374461	0.679201901651	-3.074608936016
12 of 16	1.336368636093	2.246422801198	1.715247037291	-1.187528931588
13 of 16	2.464505177713	0.784928529530	1.016529166115	3.071392977328
14 of 16	1.497369096808	1.590064153130	1.733641259967	2.460070119010
15 of 16	1.033920897837	1.585750911020	3.109905778214	1.952098517691
16 of 16	1.231525448305	0.207754949284	1.795834852756	2.512446407784
$k=17$				
1 of 17	2.590617795303	2.825418807043	1.238944988962	2.351174686582
2 of 17	0.933459077996	1.572553477647	2.452209383567	-2.200013018892
3 of 17	1.407336869592	2.428551044951	0.866390248720	-2.569202374724
4 of 17	1.539104320123	2.350493975258	2.295297060355	0.441993206820
5 of 17	0.711743817958	1.608442668958	1.246914756770	1.607868790829
6 of 17	2.363466910950	0.327568991754	0.499036958711	-0.555379944053
7 of 17	2.058343188949	1.384910586739	3.115573585903	-1.951781752819
8 of 17	1.394657235366	0.681016779100	2.013902576850	2.521022984062
9 of 17	1.930714244130	1.437250793274	1.656316517273	-2.621511393226
10 of 17	1.380403178846	0.904052598306	1.980437629221	0.056761743699
11 of 17	0.862338551800	1.916090144601	1.280522592410	-0.657106845626
12 of 17	1.535343338271	1.979617164305	2.035572054177	2.497876109629
13 of 17	1.780658258687	1.464199718811	0.831496732053	2.454238436959
14 of 17	1.901371847786	1.419321495131	1.714054629223	1.310523756036
15 of 17	1.988044854450	1.696321651101	1.678041672105	-1.212582478989
16 of 17	1.244759917165	0.933506685186	0.701276606453	-1.963209758086
17 of 17	1.816336235504	1.713158240158	0.705820343208	0.305049349988
$k=18$				
1 of 18	2.244052902868	1.245593800545	2.525622836136	2.955750577331
2 of 18	1.202741478897	1.249300280004	1.147511636170	1.326961938261
3 of 18	2.130745306188	1.021387194937	0.672776719641	2.529953989751
4 of 18	1.437895920395	0.139139303626	2.935165058921	-0.833827474352
5 of 18	1.199905387936	1.233708617807	0.837225718318	-2.178465998495
6 of 18	1.450053221286	2.280446438812	1.355267297504	-1.382947830447
7 of 18	2.160750118691	1.159094215024	1.754992901594	0.544115530890
8 of 18	2.539544466028	2.864731953245	0.173582426463	0.297375862882
9 of 18	1.426931013338	2.158094554312	0.614431305589	2.521565675738
10 of 18	1.231333562879	1.379118070784	1.828213080400	-0.445115790899
11 of 18	2.158032820907	1.144800690837	1.500790571604	-1.364017346117
12 of 18	1.219610303216	1.305536247141	1.663607129851	2.746551349174
13 of 18	1.246299378832	1.426543952449	2.569454033999	1.215940360596
14 of 18	0.249738534537	3.127967070720	1.744865051567	2.922445153604
15 of 18	1.687368296947	1.574544628370	0.626838005492	-0.188591132244
16 of 18	1.243913942378	1.414172711223	2.261673628870	-2.013471364161

Table A.3: Coordinates of 5-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$	$\widetilde{x}_4$
17 of 18	1.519796958877	2.406855351176	2.406519274716	2.925005060716
18 of 18	1.452329518183	2.295542185540	1.613115903575	0.576632968775
$k = 19$				
1 of 19	1.576274413833	1.301007152066	0.803446858780	-0.312512558826
2 of 19	0.897549159268	1.298185171373	0.977804307341	1.356343831617
3 of 19	0.951479922084	1.323340561628	2.743684864383	-2.287597423016
4 of 19	2.652235131177	0.398526644561	1.005449552217	-0.196955658812
5 of 19	1.447021551126	2.272961810677	1.885446220022	1.435236559070
6 of 19	1.728944879018	1.213526408249	0.546619049267	2.977892482991
7 of 19	2.100812149647	2.315719487874	1.548926586514	-0.173646260234
8 of 19	1.827226498103	1.170223150785	1.740707260812	2.013626703812
9 of 19	0.965348548596	1.400873980135	1.100464345962	-1.859074855140
10 of 19	1.727450865441	1.338997637165	1.916076499310	-1.087783331458
11 of 19	2.288370613154	2.307323636695	1.326274458626	2.798650574624
12 of 19	1.672589026831	1.266188843265	1.861333470614	0.584312458068
13 of 19	0.861223229254	2.035379524973	1.737392312493	-0.203769496774
14 of 19	1.409583592824	0.050750574935	1.636577674190	-0.693497044413
15 of 19	1.100018352256	1.840275249021	1.517917982325	2.852687109013
16 of 19	2.083584702741	1.712258619506	3.021434076351	1.646524118439
17 of 19	1.883914474867	1.243619941282	1.799359683980	-2.522134796267
18 of 19	1.349394020681	2.331741552269	0.001985992181	-2.298751396226
19 of 19	1.501679926716	2.366115523754	1.995823238524	-1.959106701154
$k = 20$				
1 of 20	0.441396832139	0.227820811940	2.711738255596	-0.001987632866
2 of 20	1.529916713813	1.978483220781	0.305240120616	2.336633835609
3 of 20	2.474354895384	1.442903685661	2.939964406370	-1.061018401766
4 of 20	1.416394546911	1.309272741781	2.521644352892	2.254677431274
5 of 20	1.612814676599	2.053363467783	2.161571070563	-2.559264539608
6 of 20	1.637109711970	0.723637580845	2.030726911054	0.344531942948
7 of 20	1.498996898644	0.780837074904	0.185320715423	-1.535155695059
8 of 20	1.867800113358	2.713263693434	1.357922225020	-0.723085023482
9 of 20	1.562059175624	0.853679620804	2.053420350050	-2.065251669572
10 of 20	0.867590246225	2.711433982197	1.863034747755	1.973212602566
11 of 20	1.274696937882	1.342771801687	1.347399532678	3.030619195979
12 of 20	2.097260504522	2.002864068881	1.628307751711	2.306765640071
13 of 20	2.378255977574	1.381286705415	0.868684365479	0.744008641526
14 of 20	1.711931468372	1.926201436782	2.038674183668	0.867374962092
15 of 20	1.908663410437	1.374596913500	1.415402420152	-0.883515290796
16 of 20	1.130052085730	1.794912386205	1.110160563924	0.063807670193
17 of 20	1.352271531325	1.251123871190	1.283121538717	1.603388611499
18 of 20	2.275136869450	1.518720113114	1.044975576284	-2.416503683453
19 of 20	1.198653961953	1.845072827330	2.378133073760	-0.718177710982
20 of 20	1.045332241904	1.914568990449	1.146293729682	-1.686560135513
$k = 21$				
1 of 21	0.995861807421	1.187703229372	0.904568438169	2.296905672116
2 of 21	1.026772418019	2.004063952893	1.015283767776	-2.289166713166

Table A.3: Coordinates of 5-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$	$\widetilde{x}_4$
3 of 21	1.892881431985	1.581517910213	1.795350611616	-1.635427955643
4 of 21	1.680134221506	2.219748581479	0.629594178534	1.829610673143
5 of 21	0.799807255208	1.960212968487	1.075796522542	0.353043432231
6 of 21	1.751792670501	1.890038322072	1.839086075831	0.951845263544
7 of 21	1.023309860563	2.253301467717	2.072282812106	2.204822026845
8 of 21	2.083839643970	1.330504882899	1.554849988533	2.126642021166
9 of 21	1.148341185263	0.965803764239	2.396897216309	1.401810800970
10 of 21	2.075751584164	0.342882962130	2.708607808818	-1.550547514763
11 of 21	1.731743056952	1.757612749205	0.745215252736	-0.810835941403
12 of 21	2.852899031551	2.249167598800	1.473214569512	-0.134373467699
13 of 21	1.975060732810	2.192332538347	1.738837837280	-2.971390999444
14 of 21	1.298006747854	1.144622435869	1.962360663862	-2.840508589589
15 of 21	0.833583920767	1.707785079129	2.428253487814	-1.240045647048
16 of 21	1.108816636202	0.734511656808	1.048967528577	-1.341955503770
17 of 21	1.939325762061	1.202437174711	0.771321816324	-2.702961577574
18 of 21	1.905991741538	1.694435531015	3.126920845040	-2.038894332206
19 of 21	1.724046973900	1.418639428431	1.886523958086	-0.304569413809
20 of 21	1.586585948559	2.716126606601	1.933606052351	-0.634345136145
21 of 21	1.673284707816	0.997629110028	0.948996997277	0.646945666933
$k = 22$				
1 of 22	1.813843580944	1.764601281281	0.789196995442	0.472167868531
2 of 22	1.040508591251	1.361986140815	1.704394529819	-0.808134586351
3 of 22	2.070921088151	0.845446318989	1.856908718767	0.422446928951
4 of 22	2.087097311308	0.832153504796	1.933630868676	-2.550135990661
5 of 22	1.953057577713	0.939469418516	0.684397830620	2.106377767123
6 of 22	1.902833779790	1.708425316909	1.770821897481	1.618407558767
7 of 22	2.934818543532	2.454742978059	0.069149818158	1.300544329475
8 of 22	1.446194416512	1.244316915088	2.858394182461	1.598816265142
9 of 22	1.203690273465	1.709230046743	1.862878560302	0.533737608986
10 of 22	1.178662981559	0.017587062706	2.856922508630	2.367700493444
11 of 22	0.891455558930	2.386882801538	2.314993397390	2.214644254298
12 of 22	1.747399999186	2.151933379878	0.888044792614	2.657038704925
13 of 22	1.011079639838	2.337873715683	0.527806062953	-0.998894771602
14 of 22	1.815225710275	2.719451818417	2.010359470397	0.066096518993
15 of 22	2.032270986804	1.616634782578	2.353782311947	-1.023092013223
16 of 22	1.098581868464	1.499232776348	2.130990734481	-2.280342625559
17 of 22	0.889730321085	1.400813620752	0.894855130453	1.557829242399
18 of 22	1.235708077227	1.292282010802	0.818611992180	-2.532475898539
19 of 22	1.949830796208	1.096583844868	0.909441047467	-1.084721371236
20 of 22	1.851286103442	2.054317069116	1.369701834476	-1.938569948665
21 of 22	1.337901356040	1.274932774475	1.718282407272	2.678263608119
22 of 22	1.955067466304	2.044204743942	2.362672253895	3.119310431128
$k = 23$				
1 of 23	1.944812201706	1.725212413322	2.360740388814	-1.420321994802
2 of 23	1.157064224356	1.806618069486	1.986380331216	1.306108722748
3 of 23	2.024879379760	0.733596511570	0.935549828993	0.903992386017

Table A.3: Coordinates of 5-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$	$\widetilde{x}_4$
4 of 23	2.158774906146	1.125528595276	1.114452342101	-1.445168733860
5 of 23	1.205536962038	1.557417747638	0.999192788978	0.435220703501
6 of 23	1.295997007783	2.713926355209	2.937128623575	-1.960485640070
7 of 23	1.010225359585	0.477022328581	2.052690518613	0.865648937112
8 of 23	2.138045883006	2.016588045162	1.281887693439	1.454505249935
9 of 23	1.208488566646	0.988795220888	0.253859572408	-2.335253230198
10 of 23	0.192703971978	2.568618845516	0.340263863866	-2.489600882576
11 of 23	1.335815813947	1.867414160820	1.481953085836	-2.612246514444
12 of 23	1.109552125898	1.083864140903	1.762446367236	-1.592738534417
13 of 23	2.103060309948	1.390767265120	0.891689383605	2.943268795238
14 of 23	1.144049560868	1.465217329924	2.790556356575	-3.085380164189
15 of 23	1.357572120694	2.161686247543	0.353402529846	2.387182227599
16 of 23	2.143318123345	1.794118366433	1.531909763805	-0.106675014080
17 of 23	1.153015893975	1.308853108220	1.533539510860	2.490530789234
18 of 23	1.124772730456	1.657197421207	2.156005125530	-0.259922000009
19 of 23	2.224050419199	2.589113268480	1.025351922605	-1.992982270825
20 of 23	1.945419567960	1.187260190144	2.542508554443	0.751055555270
21 of 23	1.326421742275	2.015266310157	1.094828006415	-0.992136188040
22 of 23	2.153244747538	1.819552773812	2.271623457044	2.769281570493
23 of 23	1.826946447461	0.625838291731	2.129041374799	-2.863327358988
$k=24$				
1 of 24	1.405793378506	1.899182513780	1.456937954435	-2.196719738308
2 of 24	1.972428382713	2.094390114803	0.296892859275	-1.978067563032
3 of 24	2.085219603690	2.191650995887	1.582477398500	2.459072847054
4 of 24	1.247113673910	3.056050528591	0.176675150864	-2.365490737870
5 of 24	1.057243079276	1.352949015066	0.446330092889	-1.328993358609
6 of 24	1.190319175859	1.898672929393	1.835564322931	1.625057135054
7 of 24	0.179848818061	2.016362657238	2.775902965236	-1.451421043008
8 of 24	2.031967402053	1.052834181484	0.861054518937	2.864432908932
9 of 24	1.163324085504	1.489977823855	1.595154858212	0.374188701062
10 of 24	1.227153948241	0.934155261452	2.698023007740	1.277571269004
11 of 24	1.517608806148	2.040089941397	2.732631845756	0.159224595658
12 of 24	2.102185962127	1.214419030452	2.043377845966	-0.222100214650
13 of 24	1.100972132501	1.798452535052	1.031356682150	2.791833999068
14 of 24	2.125113852624	1.152185332061	1.790352435887	1.456468470586
15 of 24	2.327148844034	2.275917127512	2.103751154586	-1.603043790852
16 of 24	2.132044363068	0.944768653223	2.472254702781	-2.707002321351
17 of 24	1.304159651983	1.758353763282	2.308199868296	3.067778674535
18 of 24	1.139240228752	0.761399745964	1.582936215559	-2.677106187211
19 of 24	1.535007650606	1.533689555496	0.741716813789	1.294604083651
20 of 24	1.296332332634	1.331786994210	2.186142628691	-1.331656320951
21 of 24	1.536680462350	1.948563979223	1.348211882861	-0.703168953218
22 of 24	2.062679957191	2.256244020186	1.444383884428	0.591825400626
23 of 24	2.010162412686	1.114796118241	1.181423832431	-1.507491937577
24 of 24	1.569460088203	0.460525258146	0.881812056026	0.335640988882
$k=25$				

Table A.3: Coordinates of 5-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$	$\widetilde{x}_4$
1 of 25	2.359679139417	1.044362258092	1.999276546775	-2.668445259376
2 of 25	1.208586513137	1.814748548571	1.577898238839	-0.054526699093
3 of 25	2.106868272117	1.699224262099	1.470374541575	-0.854588035185
4 of 25	2.558080048221	2.825945283516	1.942297204501	1.537707225948
5 of 25	2.014381622798	1.863612997398	0.802936723549	-2.272527606074
6 of 25	1.612426223455	0.271022535465	2.601468012524	1.349038447529
7 of 25	0.877638925813	1.937771270829	1.050159745546	-2.596371565300
8 of 25	1.557352669630	1.030337170038	0.994285459370	2.999576064762
9 of 25	1.485762731516	2.800665143062	0.723754707815	-0.857553657808
10 of 25	2.097297976558	1.311196677872	2.009986567591	0.435472498594
11 of 25	1.824711399702	1.723649969851	2.840048723232	2.186244267711
12 of 25	1.616511113382	1.927366388844	0.686594217794	1.974121901960
13 of 25	1.530351555968	0.979611931221	1.336969078396	-1.717518320998
14 of 25	1.431905808846	1.273632282713	2.400686319723	-0.884178822455
15 of 25	1.721227548436	2.044504207253	2.080033948755	-1.964085893995
16 of 25	0.532670130274	0.480094175395	1.477828846881	-0.643153548276
17 of 25	1.201359843344	1.195291206146	2.206789796468	-2.781053203327
18 of 25	1.348471420036	1.167977176573	1.151645601219	0.963965159439
19 of 25	1.292907553343	1.504088047454	0.440651327278	-0.673384250346
20 of 25	1.732475428674	1.998459743094	1.684563100532	2.934873098044
21 of 25	2.048064746383	1.279760628897	1.659900655549	1.925008535136
22 of 25	0.905165114812	1.513044672435	1.631649462042	2.145943438088
23 of 25	1.524700991613	2.143648258802	1.816417628807	1.185906414835
24 of 25	2.270817993145	0.885288134246	0.335401187220	0.370022523556
25 of 25	0.813395332051	2.280354897537	3.025524172901	-1.165216252931
$k = 26$				
1 of 26	0.736734051700	0.790942917628	1.105296776788	2.792262243518
2 of 26	1.687803307366	1.617296454844	1.505017569605	-0.803701262159
3 of 26	1.362306250645	0.879845547424	0.705065102262	0.216857514676
4 of 26	1.324360300544	2.036601115473	2.636227070132	-0.163951477718
5 of 26	2.091885046142	1.153299302837	2.836103208159	-0.436122838191
6 of 26	1.031894858314	0.823015027119	2.371014302193	-0.319519458866
7 of 26	1.777890443256	1.220034042009	0.642885096232	2.595041234286
8 of 26	1.123596964055	2.107669415506	0.898385120971	2.040610456739
9 of 26	1.029936957937	1.526605552819	2.069020673581	1.484374274438
10 of 26	2.016852436338	1.679976048262	1.445707019256	1.576048635916
11 of 26	2.347418834825	1.365692423653	1.860035868836	2.928252350393
12 of 26	1.633653031447	2.282922104143	2.212413147211	2.428735902840
13 of 26	0.750223684462	1.814813413121	1.923428603267	-1.844381594916
14 of 26	1.322854261828	1.276230235859	2.459878710676	3.139100559340
15 of 26	1.695767727188	1.451302551991	1.769148150956	0.456196087192
16 of 26	1.407087463011	1.793153503982	1.391583979428	-2.920587785034
17 of 26	2.349468133794	2.503419182005	2.163855430434	0.205596082433
18 of 26	2.233351211497	2.042857534079	0.751332298536	-2.161144234238
19 of 26	1.779382822189	1.970540806177	0.685275424587	0.389242208353
20 of 26	0.738904875766	1.942444386621	1.285508253496	0.126864527810

Table A.3: Coordinates of 5-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$	$\widetilde{x}_4$
21 of 26	1.161880561968	1.674116066260	0.500719204167	-1.777845336661
22 of 26	2.521942499642	0.776399962285	0.809883788266	0.209065279070
23 of 26	1.694296123861	0.570354480457	1.866788352349	1.829985670556
24 of 26	1.408567332599	2.814668447121	1.135241381718	-1.308502320700
25 of 26	1.872460546926	1.984941976974	2.196121097689	-2.034785975143
26 of 26	1.695609172973	0.799704888589	1.429548350841	-2.047106387885
$k = 27$				
1 of 27	2.217708218722	0.982501496493	1.804585858113	-0.795753959421
2 of 27	1.363761525883	1.052483196398	1.363658124769	2.143683731830
3 of 27	2.863966890697	2.617260963779	1.942674102267	2.127423009290
4 of 27	1.055222144651	2.168979479660	1.905522384287	-2.331350246210
5 of 27	1.722233769344	2.918849760123	2.319868297353	2.087301285895
6 of 27	1.239511902438	1.177643333405	1.152784902665	-2.715553320175
7 of 27	1.613584617249	2.135942951242	0.603881622262	-2.396449617435
8 of 27	1.908228834988	1.897850057791	2.121803997520	0.433356627031
9 of 27	1.669064055601	2.082856250005	1.936905056663	-1.013758262789
10 of 27	1.405091772630	0.239973426259	0.757026247742	-0.663639850191
11 of 27	1.929033681673	1.801594393266	0.886962956952	-0.106668497384
12 of 27	2.161313740703	1.063649015935	0.356082601428	-2.456182170545
13 of 27	1.888525256710	1.814175169006	2.871871516610	-2.667964684901
14 of 27	1.145423707290	1.352001716168	0.199342807182	1.060316288624
15 of 27	1.837869397484	1.855053369729	1.579082555833	2.917361309242
16 of 27	1.292687744129	1.135180361481	2.278428297196	-1.588303203662
17 of 27	1.087450473273	1.918422842138	2.240830868289	2.159506534047
18 of 27	0.274714022647	0.208463887528	2.844079429961	3.061750061479
19 of 27	2.101654280566	0.958699962598	1.389598470351	1.007632381920
20 of 27	1.120842591280	1.594846212439	1.482908494783	0.999873447703
21 of 27	1.413084981360	1.008616743827	2.577443769775	1.022058896914
22 of 27	1.848875958205	1.945151183042	0.895581032208	1.724064376579
23 of 27	1.205393229218	1.305206306903	1.719489446374	-0.236785581387
24 of 27	0.917063803758	2.442299017355	0.960195720670	-0.298900060602
25 of 27	2.043989568944	1.616405445840	1.650605628098	-2.130138576054
26 of 27	1.410003683359	1.371309634815	1.070466970904	-1.314014244853
27 of 27	2.114200242424	0.734540566700	2.178807681430	2.998030452195
$k = 28$				
1 of 28	2.278860234894	3.041917775786	2.310483149295	0.232750054776
2 of 28	1.462576749631	1.199535889272	1.484346052917	2.827062419229
3 of 28	1.861953031837	1.877510229643	0.877816436185	0.788916516559
4 of 28	1.181551785009	2.701659440672	2.224041160239	1.138923443751
5 of 28	1.878168959456	1.832550050689	1.903544935595	1.588430442404
6 of 28	2.390806227513	1.425416090463	2.114328062664	-1.643683199570
7 of 28	1.371179158658	1.225083916235	2.140347059487	-2.328670800598
8 of 28	1.610252184324	2.174898817817	0.978675265677	2.431425587377
9 of 28	1.766272369066	1.901530673799	1.581709200962	-0.416046081347
10 of 28	1.879023586612	0.394124331021	1.123215357998	-2.260891183812
11 of 28	1.784693442044	1.856251036615	2.782902592354	0.062205077067

Table A.3: Coordinates of 5-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$	$\widetilde{x}_4$
12 of 28	1.490928110394	1.313741649320	0.290876464139	-3.055716362240
13 of 28	2.311663664874	1.106828912018	1.856839355685	0.448631099316
14 of 28	1.397317256357	1.341282134989	1.219511900454	-1.483981126685
15 of 28	0.782834688419	1.722320346996	0.851509600214	1.341653319966
16 of 28	1.707003195877	0.921416365256	2.612010758944	2.048627565789
17 of 28	1.682974932932	2.046486652441	2.242462655879	2.996534738584
18 of 28	1.513300392892	0.967278487598	2.188968909121	-0.704827935056
19 of 28	0.773806605903	2.079580218986	1.452745667938	-2.816055991360
20 of 28	1.853346963410	1.787233193711	1.252423759198	-2.523665854554
21 of 28	0.644828673750	1.797314716835	1.773488895822	-0.631684612063
22 of 28	1.302578730454	2.281421190555	0.512594970445	-1.057430898692
23 of 28	1.327967604362	1.080502012961	0.944850533891	-0.072883887367
24 of 28	2.539195111662	1.342630494387	1.112033634569	2.497190944541
25 of 28	1.486698222765	2.225269091759	2.001806612398	-1.636617018775
26 of 28	0.807353366260	0.155534148141	2.130114903962	2.454382068999
27 of 28	1.207190141079	1.336404470306	1.990868446080	0.705474998376
28 of 28	1.636824662757	0.918597755850	1.165205443399	1.515756814806
$k = 29$				
1 of 29	1.865574224762	1.313748652173	0.773007002529	-0.574748456946
2 of 29	1.340068624667	2.028623650340	1.483212293832	-0.905023666873
3 of 29	1.162346136572	0.773627480924	0.451245922550	1.365489150203
4 of 29	1.207109463748	1.778890687724	2.349752135914	2.504870218284
5 of 29	1.145480412417	1.589027648025	1.275392568015	-3.072680295574
6 of 29	1.131602306259	1.909958112631	1.296513722635	0.430252739185
7 of 29	2.013903393635	1.619217092216	1.866584785296	-0.011270386882
8 of 29	1.998431698379	2.376031823471	2.620281142231	-1.505365564346
9 of 29	0.041749179056	0.670648074708	2.407078225127	1.951569435515
10 of 29	1.975625313626	1.409114659356	1.051328124921	1.069791956741
11 of 29	1.957437934262	2.253825369183	1.415804948745	2.661995713505
12 of 29	1.193332405601	1.115549091941	1.126083976935	-1.764877946000
13 of 29	1.167334965533	1.919750205064	0.135320925042	-1.669726265174
14 of 29	1.999084148214	1.319855979486	0.539476923036	3.002540010029
15 of 29	2.001930646256	0.287671893538	1.753577804186	0.268289267241
16 of 29	1.769415201474	2.129390641660	2.034414343055	1.232549915922
17 of 29	1.796841386075	1.271948081843	3.036539132615	0.062243294860
18 of 29	2.046030806541	1.284781318553	1.858374490772	-1.349989015747
19 of 29	1.182874203513	0.620501365643	2.268275008609	-3.048150718850
20 of 29	1.849496542678	2.034350688326	1.150979545081	-2.081532065074
21 of 29	1.116552320018	3.036018577238	2.046390847532	2.521279284312
22 of 29	1.147928369776	1.806411397625	1.203486081766	1.854880860606
23 of 29	2.010239083475	2.481975637381	0.705571012920	0.096709192240
24 of 29	1.161023791808	0.987235249105	2.006847042894	-0.533224185846
25 of 29	1.886095496088	1.126849018095	1.696893424842	2.214350704658
26 of 29	1.983535611983	1.418864440135	2.011547411938	-2.771748388069
27 of 29	1.146177269501	1.666514688429	2.290426756342	-1.895732890433
28 of 29	3.075485665742	1.645597668817	1.921101112805	1.999034326333

Table A.3: Coordinates of 5-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$	$\widetilde{x}_4$
29 of 29	1.222077423697	1.108384146197	2.011331432660	1.137616032670
$k=30$				
1 of 30	1.281957681241	0.910645351007	1.919114348700	-0.194529667285
2 of 30	1.970753461817	1.350442707497	1.396082468986	-2.416654740410
3 of 30	2.230168319921	1.443804323263	1.391973836360	1.876064377947
4 of 30	0.968812370671	3.101986340135	1.654108851040	0.940484017230
5 of 30	2.851562430486	2.291609953501	2.104666880379	-1.464925174790
6 of 30	1.119707402488	1.757902583591	1.791473812394	0.794878508194
7 of 30	1.298995830391	0.770049711951	0.244369155280	0.359139594213
8 of 30	1.954706106228	2.490646165443	1.383805828454	0.912000191373
9 of 30	1.129274409195	1.693752874632	0.679850825163	-2.254215697416
10 of 30	2.120190851728	1.547941212947	0.097170582436	-2.396838264080
11 of 30	2.195908582058	0.052134627404	1.134138361406	-1.453342964770
12 of 30	0.851058501814	1.658928152587	1.809038553034	-1.329951566662
13 of 30	1.326281665498	1.773018604165	0.767739179808	1.569061581533
14 of 30	0.329994150159	1.081814397052	1.219051410450	2.169502892898
15 of 30	1.749521527959	2.290121968242	1.409015623639	-1.627866606479
16 of 30	1.113550524735	1.822847514283	1.899723321415	-2.804398454337
17 of 30	2.033339451489	1.462065672290	2.445433055906	0.842198297790
18 of 30	1.794558264232	2.310976911536	2.842660661229	-3.008771755487
19 of 30	1.199662721671	0.612763841488	1.980914339271	-2.351948549510
20 of 30	1.604794125329	2.009522259169	2.032259997861	-0.365649810414
21 of 30	1.428520621829	1.201415411504	1.177262431695	2.764586743924
22 of 30	1.157016497519	1.809292208734	0.897086781911	-0.234884436351
23 of 30	1.297140936220	1.889207182001	1.897624967598	2.111423586023
24 of 30	1.753423521234	1.225817954111	1.193086612719	-1.141786005373
25 of 30	1.958849410109	1.304793388395	1.196537031776	0.479747614948
26 of 30	1.889272307516	1.239683882760	2.311134063842	2.879780172548
27 of 30	1.057411532375	1.451621038953	3.049433073250	1.559450783941
28 of 30	1.378160961547	0.798709179390	1.898852644287	1.594469324718
29 of 30	1.879421111957	2.254203682441	1.191435520889	2.957205762493
30 of 30	1.834849074246	1.382074892567	2.353546872247	-1.537304708058
$k=31$				
1 of 31	1.930449630069	0.571914060613	2.655470262655	-1.814136017393
2 of 31	1.855028048891	2.057266768754	0.678233679753	-1.582888270172
3 of 31	0.447937011854	0.415225604976	1.592082317032	-3.097204761927
4 of 31	1.719067339135	1.227331557510	1.314033516241	1.679613767666
5 of 31	1.307118849924	1.368165233772	2.249086406375	-2.724405482946
6 of 31	1.130637083138	1.241198542193	2.411108567995	-0.768898860318
7 of 31	1.228110934056	2.563787886604	0.524815213416	2.064213484326
8 of 31	1.142639943273	1.747998887659	1.117780401120	-2.616321246055
9 of 31	1.292825962656	1.086120421805	2.461874964872	1.685459847588
10 of 31	1.684039916828	2.573324406781	1.973383799566	-0.359921767786
11 of 31	1.482104326990	1.560239601658	1.555571535788	0.015107554745
12 of 31	1.438615607912	1.915476341331	1.830100937621	-1.661062966275
13 of 31	1.789246957737	2.209070828682	1.924766699429	1.592081302753

Table A.3: Coordinates of 5-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$	$\widetilde{x}_4$
14 of 31	0.802855994144	2.392347487583	2.944513369039	-2.601305476241
15 of 31	2.731978862523	3.124252168601	1.426762607281	2.311238888655
16 of 31	1.521027144993	0.811910715217	1.241616074984	2.993496615902
17 of 31	2.145942903814	1.391978065190	1.573106474929	-0.931585369120
18 of 31	1.379766737727	0.531109253866	1.106527594055	0.432409417405
19 of 31	1.358503563861	0.934428795306	1.219239474247	-1.641271465862
20 of 31	1.738016427332	2.361761322429	1.946253499839	-2.952872032402
21 of 31	2.131624328945	1.396906305318	2.087336075004	2.583718094608
22 of 31	2.182538291853	1.282827437321	2.118170806766	0.764835443348
23 of 31	1.142950602922	1.734953215109	1.655944602954	2.476983124770
24 of 31	1.975920788163	1.775955922406	2.868149203363	-1.489277119055
25 of 31	2.131290598822	1.410920810012	1.503341555020	-2.387084340919
26 of 31	2.347133423188	0.741858498153	0.104933442299	1.276644783882
27 of 31	1.927529967182	1.878370174672	0.754433584727	0.735082031783
28 of 31	1.172208174665	1.367005231660	0.288181447005	1.918546796465
29 of 31	1.957703390546	1.797491781010	0.882228221609	2.730342420307
30 of 31	0.901000494691	1.750504768444	1.551261358698	1.068069021636
31 of 31	0.861681096784	1.869546769052	0.950387225266	-0.869923463736
$k = 32$				
1 of 32	2.212498982553	1.270436112115	2.207699359327	-0.821087042920
2 of 32	1.501748550618	3.044241825328	2.633388520923	1.542456745256
3 of 32	1.661035831351	1.963537111426	0.295334817911	0.212343686343
4 of 32	1.793004790953	1.242170164925	0.919509599856	1.529556228680
5 of 32	1.742983795702	1.312092114138	0.637037433477	-2.872133644540
6 of 32	1.569359859984	1.832784411516	1.783557931844	-2.961981373642
7 of 32	0.945344583175	2.006180320426	2.783220859674	2.808017195726
8 of 32	1.836833262610	1.093297049096	1.676252426721	-2.124812661591
9 of 32	1.213067592882	2.117701416581	0.952616313554	-2.119568299958
10 of 32	2.152739725071	0.983077846007	0.675836445446	-0.455959482576
11 of 32	1.117290172355	1.190352582236	2.413045455960	-0.780310857951
12 of 32	1.700076848343	1.142609964459	2.722055951964	3.091023590139
13 of 32	1.285714419845	1.268803033587	1.188151428244	-1.172764806280
14 of 32	1.937703555465	0.945740435688	2.084032155019	0.926790353472
15 of 32	1.426423373760	0.137945148976	1.330762303543	-0.994089838167
16 of 32	2.304563720094	2.064203391680	1.022141083287	-1.812243541242
17 of 32	1.397210953148	1.007183961173	1.541280995493	2.586240313327
18 of 32	2.301873944337	2.210441527173	2.738747422123	3.099367316969
19 of 32	0.892348078996	1.113620679318	0.204950431947	0.982915172002
20 of 32	2.495098592330	0.821894784046	1.550506352043	2.723549267494
21 of 32	1.544906824146	2.000658485534	2.127859728920	-1.668532990738
22 of 32	1.718394571561	2.080852959102	1.468639637719	-0.545489666681
23 of 32	2.044801088087	2.152213106578	1.058694047457	2.441972009493
24 of 32	0.853545830424	1.010399972337	2.012981442482	1.277186542395
25 of 32	1.629241300314	1.919626852291	2.415434270626	0.359962955552
26 of 32	0.938649840121	2.066532288501	1.170787149687	2.403685407305
27 of 32	2.383778452437	2.015786824323	1.373530119677	0.670587447314

Table A.3: Coordinates of 5-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$	$\widetilde{x}_4$
28 of 32	0.750190019322	1.083353884782	1.680205045751	-2.429962343290
29 of 32	1.657852646460	1.737380758782	2.032487889634	1.874987573107
30 of 32	1.298177069096	2.005980098014	1.379917765621	0.978208367807
31 of 32	1.336880482942	1.223345345270	1.407183428934	0.091896709432
32 of 32	0.615687729487	2.222527035996	1.447883902889	-0.411199924891
$k = 33$				
1 of 33	0.825180185649	1.330124047390	1.693473660149	-2.518881853598
2 of 33	0.396432510421	2.392541232216	1.560877241813	1.134790012217
3 of 33	1.212760983310	2.106591734778	2.607609111399	-1.955936639597
4 of 33	1.366321249029	2.271513202029	1.918491608124	2.439796579463
5 of 33	2.121486204430	2.475138310541	1.680928675819	1.225569721297
6 of 33	1.348459039954	1.966926250284	2.338031402460	0.664315075853
7 of 33	1.995042280106	1.775030763043	0.303201351673	-1.075789952530
8 of 33	1.866239876648	1.473223216584	1.447473254820	-1.714051000618
9 of 33	0.951013922925	1.268904659553	0.164767148450	-0.719193735784
10 of 33	1.372290527525	2.000954589662	0.783422158197	1.368338955789
11 of 33	1.859052818236	2.619449397302	1.491776599892	-1.688823667164
12 of 33	1.237531928921	2.141654396942	0.784562341804	-2.846780246992
13 of 33	2.039913582546	1.810289967895	1.052200312343	2.522496025505
14 of 33	1.191063786768	1.259213416984	1.364643723228	0.576841929425
15 of 33	1.861984136941	0.964751788412	1.897123044644	2.682656147307
16 of 33	1.538979978345	1.194003892889	1.224679541410	-0.611193694841
17 of 33	0.922080830956	1.219117611121	2.283701494964	-0.560023404318
18 of 33	2.264747134653	1.354958010857	1.355761770249	0.346029906504
19 of 33	1.770491516560	0.985992974450	0.683940044765	1.530994971220
20 of 33	2.867069015855	1.988300434260	1.724517269000	-2.287028689702
21 of 33	2.013953503570	1.770382468647	2.774415039761	2.376797850601
22 of 33	1.057969899378	1.346295301691	1.210841074368	2.407019829505
23 of 33	1.639295608031	1.114864434589	0.888481478105	-2.753256415646
24 of 33	1.123753409315	0.129325153974	1.525662176211	1.745326912045
25 of 33	1.802113016465	1.805376001714	1.823412043846	-2.755202041799
26 of 33	1.746768422304	0.948954167160	2.462078506481	0.637560617397
27 of 33	1.615952112824	1.078988277681	2.473191984386	-2.067148280323
28 of 33	1.036728072308	1.312796485097	2.553038678510	2.241706962039
29 of 33	1.613880113110	1.482848874310	1.734777794620	1.552805964361
30 of 33	2.134425120209	0.311202578777	1.301991568942	-1.362829862661
31 of 33	1.968271774963	1.737717021284	2.207836957803	-0.709163179341
32 of 33	1.532092557687	2.161387125949	1.282586779709	-0.050077662578
33 of 33	0.990918409143	1.996878062321	1.319380584791	-1.270151155748
$k = 34$				
1 of 34	2.014156038356	1.852211478028	2.236969038430	-0.421356940810
2 of 34	0.484412309867	0.359177863000	1.226348555012	-1.468923603732
3 of 34	1.155735477017	1.598344466016	0.860384815540	-1.973954136770
4 of 34	1.231883891204	2.138010596857	2.952659657336	0.407504274600
5 of 34	1.840595554723	2.032214705597	1.792349846105	0.887663093159
6 of 34	2.375069739488	0.667225342679	2.022969687637	0.396830009614

Table A.3: Coordinates of 5-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$	$\widetilde{x}_4$
7 of 34	1.112985075898	1.296822022535	0.901832386723	2.697437604263
8 of 34	2.151212742913	1.390334486641	0.946452553812	1.636889028590
9 of 34	1.025235746415	1.699864363353	0.438756820827	0.508894395950
10 of 34	1.008194565506	1.565780217245	2.134050027057	2.526585913331
11 of 34	1.900758565318	2.010898205823	2.245462865295	-2.131309322164
12 of 34	1.704547639635	3.102687769328	1.833616854001	-2.271589216312
13 of 34	1.730276209699	0.931847096919	0.156883270647	-1.779095522688
14 of 34	1.831029103126	2.048860087448	0.277733577021	-3.083002893920
15 of 34	1.983844804333	1.738966272831	0.894200669348	0.073199794645
16 of 34	1.939120686425	1.131067531877	1.646548068908	2.754448306720
17 of 34	1.050628066252	1.759115265247	1.960820008416	-1.406956673017
18 of 34	0.816258538562	1.690346114520	1.775781877348	1.036552915944
19 of 34	0.666483440849	2.776359583307	1.152242631192	-2.870903784938
20 of 34	1.982268703523	2.048007567559	2.293150784632	2.296320308653
21 of 34	1.655774089442	1.167729375775	2.006648535957	1.567404314243
22 of 34	1.268276393222	1.206612660370	1.851684234516	-2.518732999209
23 of 34	1.207260369260	1.150220465037	2.293824650997	-0.128524896516
24 of 34	1.869372030554	2.030455094028	1.259014226842	-1.249934433764
25 of 34	1.511492989909	0.179958908531	2.370245931666	-3.102332277428
26 of 34	1.887782849722	1.052846550625	1.964009618440	-1.407762102072
27 of 34	1.326931888374	1.222040926565	1.191638119349	-0.718056755142
28 of 34	1.412058271728	2.120245087194	1.260614998144	1.927363968961
29 of 34	2.775753255424	2.955181686797	0.428892888179	-0.984864036623
30 of 34	1.538473271048	2.048293127243	1.477057515085	-3.084423491828
31 of 34	1.673854183656	1.220645443327	2.907741861720	-3.025488588765
32 of 34	1.197610193024	2.189511479387	1.544032295420	-0.195067324540
33 of 34	1.453932101568	0.935041237779	1.226456178204	0.602917206405
34 of 34	2.124292721845	1.432594610663	1.201156675814	-2.355385231022
$k=35$				
1 of 35	1.300617693133	2.932854773764	1.141683427880	2.480234528528
2 of 35	2.007398199417	1.765555914158	1.969163095103	-1.805333911789
3 of 35	1.200465938207	2.172437584240	2.188232796094	-1.086957667057
4 of 35	1.893900914668	1.394346196876	1.974816323391	0.925969223655
5 of 35	2.212614410123	1.721025737966	1.934653642108	2.166604930226
6 of 35	1.912906711646	2.392503432519	1.139512269634	-0.865639870701
7 of 35	2.002536748984	2.431642320694	2.968013700378	0.356952022886
8 of 35	1.008978483997	1.338498679591	1.471557670113	-0.493865078015
9 of 35	1.435791936373	0.942495979802	2.214176228999	2.066575611651
10 of 35	2.018478382089	2.277911140244	1.178135321087	1.094392271413
11 of 35	0.922604120224	0.838664424007	1.424295996611	-2.997307220832
12 of 35	1.418453244521	0.813430337030	0.141829862363	-0.788111926528
13 of 35	1.396270941486	1.687972217705	1.069371092266	2.619231435303
14 of 35	2.069421269791	2.244397753171	1.311421231505	-2.914934885251
15 of 35	1.269510040879	1.854595845313	1.702090466799	1.669576633550
16 of 35	1.139850240821	1.982351016907	1.600193216018	-2.414395756744
17 of 35	1.812791521414	1.231911194513	0.999920947238	1.563828194093

Table A.3: Coordinates of 5-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$	$\widetilde{x}_4$
18 of 35	1.892479760598	1.650630763947	1.991634368703	-0.300720333710
19 of 35	1.197045431610	2.122148431350	1.739299890107	0.420719520805
20 of 35	0.208896477907	3.099112596407	0.846648164503	1.402147575117
21 of 35	0.945986982217	0.953679165451	1.393126631559	0.951157320452
22 of 35	2.935063069552	1.962492678209	1.586880749246	-0.386256366135
23 of 35	1.285751499476	0.986989454040	2.109455409841	-1.584503887373
24 of 35	1.695851139120	0.589297769689	1.834567006938	-0.066304132755
25 of 35	2.231474487617	0.455561947952	1.476986978543	2.799951186914
26 of 35	1.763804400643	1.443026445113	1.024953362747	0.218270151027
27 of 35	2.098595098111	1.221955237109	3.071267530746	-1.222745399325
28 of 35	1.144696914681	2.001951627931	2.631611945928	2.763509808895
29 of 35	1.739591198199	1.389411619310	2.035932900284	-2.993538462990
30 of 35	1.139757920185	1.915796164939	0.569285618560	1.132113527736
31 of 35	1.146521689095	1.310330505676	2.608882822698	0.222598022620
32 of 35	1.973583617733	1.245678694354	1.175699643604	-1.052221515838
33 of 35	1.779098244543	1.260054703184	1.085691107437	-2.371828580282
34 of 35	2.079452443264	1.710152463990	0.056179943675	-2.405854138035
35 of 35	1.213745741078	1.765247999181	0.692590468777	-1.466515499658
$k = 36$				
1 of 36	1.011812336764	1.272555266842	1.979185042456	-2.015903579147
2 of 36	1.896989818862	0.559278668111	1.563459518146	-2.781075503835
3 of 36	0.954367822926	1.691428128162	2.759233598768	0.340415081751
4 of 36	1.716795937277	1.606262785283	0.837974866695	-0.047393592312
5 of 36	0.300008837136	2.453313243361	1.294629164692	3.096868696539
6 of 36	1.005256013958	1.282057683064	0.208123860048	1.119748788012
7 of 36	1.465295809924	1.239545643254	1.874025632030	0.779522540761
8 of 36	2.281706199784	1.740921338795	3.022929401197	0.752093229761
9 of 36	1.752458091318	2.120753394839	1.974916228885	0.180105685544
10 of 36	2.049766992861	2.168269166671	1.191042232162	-1.160723632207
11 of 36	1.123626962312	1.324822937962	1.441528634536	1.809410520961
12 of 36	1.819542693568	1.308725702417	0.572192644329	-2.127440681785
13 of 36	1.539737079980	0.866718362692	1.367722651337	-1.144972530171
14 of 36	1.607980864797	0.902012453061	2.792757750550	-1.053411773705
15 of 36	1.913631362961	1.437168571396	0.918132134989	2.384783397314
16 of 36	1.603031625199	2.190012082522	0.152487641461	2.428922350094
17 of 36	1.617182542914	2.065012384830	1.300160668607	-2.443937758719
18 of 36	2.095298615830	1.639339077113	1.882535723831	3.062272282956
19 of 36	1.084870195287	1.785967743839	0.984559326812	-1.397886989715
20 of 36	2.184766994221	1.101797256609	1.791025538323	-0.149755390372
21 of 36	1.793650299298	0.695463442223	0.778542488424	0.910531182115
22 of 36	0.873198819343	1.305089061839	1.466362903682	-0.187827756593
23 of 36	1.481926663038	2.078632560699	2.522807675018	-2.506991977627
24 of 36	1.150023536630	1.255140471998	1.169344269124	-3.042447337919
25 of 36	1.120620356499	2.081500812880	1.173604744466	0.848651618127
26 of 36	2.029890149275	1.031711622559	1.981508913856	1.861535372069
27 of 36	2.118865917739	1.767571214710	1.244606570584	1.124029984843

Table A.3: Coordinates of 5-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$	$\widetilde{x}_4$
28 of 36	1.327453094198	2.166460959581	1.446204325986	2.592908106317
29 of 36	1.248236986892	2.873262199732	1.595323984197	-1.016693522977
30 of 36	2.066057361969	1.461964488922	1.967707205962	-1.899716214302
31 of 36	1.240130443763	1.255925915999	2.333955019722	2.707869888311
32 of 36	2.162829982641	2.835443067452	1.679855584677	2.330926915248
33 of 36	1.587471141068	1.995299640910	2.155899464293	1.570544931464
34 of 36	1.444701180704	1.721590860873	2.027121509620	-0.950457362190
35 of 36	0.967205997883	0.101466857597	1.906810674434	2.186274958801
36 of 36	2.873401394562	1.322940549909	0.415188355852	-2.592421778325
$k = 37$				
1 of 37	1.283203012427	1.611616026897	0.623033196074	-2.253787854129
2 of 37	0.911308355981	1.536499198143	0.661339420715	0.232656120546
3 of 37	1.976762423659	1.811793374748	1.945527384876	-0.493141302228
4 of 37	1.475345381339	1.235172439958	1.176454166911	1.295854720419
5 of 37	1.744750139332	1.506420115201	2.891878117525	0.367173524898
6 of 37	1.018920450707	2.073980255209	1.879719411763	-0.077981101276
7 of 37	2.127738586802	0.718823720502	1.980012742434	-1.092305927094
8 of 37	0.667703283154	1.798760536196	1.734819828368	1.518891514237
9 of 37	0.912774467654	0.855284056244	0.818352372637	2.566606635001
10 of 37	1.612916046640	1.957487057631	1.976833845502	1.008215529183
11 of 37	1.709916437800	1.885287417142	0.380369100031	1.533770233092
12 of 37	0.997221780984	2.068833228179	2.836060613848	-2.767527068163
13 of 37	1.480877252260	1.366734157310	2.178028234782	2.094150032813
14 of 37	1.554682255377	2.086160764519	0.900198650584	-0.821031307707
15 of 37	2.127139800029	1.613899644375	1.337485287131	2.116188308183
16 of 37	1.878334488120	1.530852171665	1.098544668091	0.231406942781
17 of 37	1.321306297878	0.801555886684	1.418809593530	-1.988855133119
18 of 37	1.189410049627	2.808049016122	0.588980715367	1.500896068365
19 of 37	1.826697255620	2.336847625853	2.255523464760	2.420535931623
20 of 37	2.616866569136	1.391031406593	0.136373775353	-1.342260094343
21 of 37	1.322045742294	1.777150009627	1.693870282223	-2.503703121897
22 of 37	1.766856757090	1.099827462648	2.351547439899	-2.784512134376
23 of 37	1.318153372405	1.176933019202	1.428275082890	-0.703358743250
24 of 37	2.001582562914	2.233246589717	1.015767469641	-2.881786439243
25 of 37	1.443544003403	1.485846367758	2.211610368485	-1.461737802048
26 of 37	1.794849366185	1.162292604934	1.182779459255	-3.095629772341
27 of 37	2.361311559208	1.116241342288	1.972005770732	0.932676640154
28 of 37	1.972906462477	1.612812918887	1.330803746295	-1.746573717300
29 of 37	0.884989241656	0.748292068415	3.005751149454	2.998282960328
30 of 37	1.699615031030	0.369724472472	1.790178310592	1.985184753730
31 of 37	1.386841895667	1.077962112336	1.947846858364	0.404067740940
32 of 37	0.542014340388	1.680121042388	1.487271867974	-1.553807601105
33 of 37	2.539364103591	1.904125515745	2.532201955732	-2.112808941443
34 of 37	1.654901994948	0.866726707960	0.263607522165	-0.459535583818
35 of 37	1.225857541170	1.904216848868	1.368984344159	2.681228838624
36 of 37	2.176463787300	2.661146929238	1.152783322820	0.478154660633

Table A.3: Coordinates of 5-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$	$\widetilde{x}_4$
37 of 37	1.676373481202	2.490866760436	2.019918311711	-1.648929390756
$k = 38$				
1 of 38	2.092789967594	2.920867624304	1.308979309861	2.978520153440
2 of 38	1.985716724468	1.905764819382	1.579648342548	0.199695272633
3 of 38	1.391347420154	2.502756724115	2.471699014786	0.372235992595
4 of 38	1.427131154840	1.852361458361	2.664100760842	-2.176919449601
5 of 38	1.650770693746	1.841350442765	2.338576683713	2.037064450794
6 of 38	1.816465770812	0.980338928495	2.867470447495	2.827548239851
7 of 38	0.977703462030	1.754865858016	1.600972101170	-0.071221719492
8 of 38	1.611915173542	2.263069802128	0.855423337281	-0.743862484288
9 of 38	2.189630135953	1.390399679830	1.732297239534	1.302715116284
10 of 38	2.141985673261	1.741908404919	1.965562127505	-2.818344642271
11 of 38	1.538161407858	1.154017831708	0.907282482037	-2.337022742559
12 of 38	2.257672270199	1.835999453874	0.449515946140	1.036943482937
13 of 38	1.235423244944	1.653915744206	0.482428461150	0.660255498719
14 of 38	0.839737225585	1.402261869368	1.786300763846	-2.402482838279
15 of 38	1.652053627241	1.875369263795	1.989593487590	-0.878234595258
16 of 38	1.285447259399	2.250816949929	1.778310066666	3.070515126657
17 of 38	0.667541059987	2.567654763797	0.475623123405	-1.733302212030
18 of 38	0.643699877145	1.782491231287	2.646168421309	1.752036629396
19 of 38	1.566885938739	2.095391949455	1.559245420816	-1.959423681348
20 of 38	1.807397245459	1.137065509545	1.923412153458	-1.802432862341
21 of 38	1.697620374408	0.957835827628	0.712430422845	1.907733871584
22 of 38	1.059038620194	1.053609679689	2.400650310354	-0.915011063233
23 of 38	1.507365393286	0.142109759130	1.452215447193	-1.686027308020
24 of 38	2.057307941387	0.947305679858	2.029772037104	-0.385748759215
25 of 38	1.277948923982	1.876302872247	1.531260848126	1.150525082076
26 of 38	1.993014977110	1.768052064382	1.328811769557	2.396898080404
27 of 38	0.982720557478	1.636179184129	1.104503056145	2.245729077816
28 of 38	2.461577995569	1.842061977361	1.295643352089	-1.358117851994
29 of 38	1.552969128423	2.048855787661	0.550253199038	-3.022791120642
30 of 38	1.499875773840	1.057939816090	1.667460471181	2.964217901817
31 of 38	1.289364023375	0.972940558857	1.910599219034	1.695284256600
32 of 38	1.111498197011	1.457906033232	1.226318223722	-1.238247920243
33 of 38	2.494146862798	0.407761782618	1.336423330661	3.078107436796
34 of 38	1.861141092067	1.080252626252	0.702959432853	-0.671640580695
35 of 38	0.560297260530	0.546285273854	0.497777149305	0.289819735250
36 of 38	1.517378429243	1.026792205501	1.342860488288	0.547658586643
37 of 38	1.503105319486	1.461896439286	2.367727734627	0.592129784336
38 of 38	2.320138715351	1.939718673593	2.887990871266	-0.131969603938
$k = 39$				
1 of 39	2.092647587099	1.679897703258	2.316504502193	-2.745096759901
2 of 39	1.435804794535	2.103833057781	3.083887280161	2.724306819498
3 of 39	2.052834073098	1.773353070655	0.817275454288	0.396525510490
4 of 39	1.436197960061	2.035388194772	0.139322664275	2.476216497644
5 of 39	2.049587565721	1.784012719557	0.796140989755	-1.451450362816

Table A.3: Coordinates of 5-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$	$\widetilde{x}_4$
6 of 39	2.256306174428	1.013924038739	1.151846052640	-2.685462485907
7 of 39	1.842916546346	0.180623921907	1.717667078769	-0.594708056997
8 of 39	2.865293230523	3.107986727732	0.007945662561	-3.005436571990
9 of 39	1.532618282737	1.724452472130	2.146841872143	-0.541805557744
10 of 39	1.397570192983	2.221838635906	1.122756931352	-0.520331287393
11 of 39	1.474233180454	1.930747356983	1.914726099191	2.606757595753
12 of 39	0.881561605620	2.144252779602	1.974094051872	-1.522331533855
13 of 39	1.045620831387	1.472592307512	0.803747261938	-1.451332902300
14 of 39	1.238595038862	0.750537915910	1.180578221592	1.619139450483
15 of 39	1.197922241069	0.954368795408	1.966138144186	0.242684636807
16 of 39	1.573675343426	1.561237055231	1.583703425460	1.035476524370
17 of 39	2.207936310030	1.246407071901	1.933863839269	-1.328913884995
18 of 39	1.906044742266	2.377445120463	1.976974940479	0.456168331698
19 of 39	0.930069825406	1.908694017164	1.186601603213	-2.898896973314
20 of 39	1.194828920904	0.966053942003	1.945057873404	-1.328812200862
21 of 39	1.048945066491	1.461694950332	0.825088151397	0.396495893688
22 of 39	1.705787859055	1.037759595809	0.057705373429	-0.417285834092
23 of 39	1.088758580492	1.368239582934	2.324317199302	-2.745067143100
24 of 39	1.705394693529	1.106204458818	3.002269989315	-0.665376155946
25 of 39	1.092005087869	1.357579934033	2.345451663834	1.690142290774
26 of 39	0.885286479162	2.127668614851	1.989746600950	0.456130167683
27 of 39	1.298676107243	2.960968731682	1.423925574821	2.546884596592
28 of 39	0.276299423067	0.033605925857	3.133646991029	0.136156081600
29 of 39	1.608974370852	1.417140181460	0.994750781447	2.599787095846
30 of 39	1.744022460607	0.919754017684	2.018835722237	2.621261366197
31 of 39	1.667359473136	1.210845296607	1.226866554398	-0.534835057837
32 of 39	2.260031047970	0.997339873988	1.167498601718	1.619261119735
33 of 39	2.095971822203	1.669000346078	2.337845391652	1.690259751289
34 of 39	1.902997614728	2.391054737680	1.961014431998	-1.522453203107
35 of 39	1.943670412521	2.187223858182	1.175454509404	-2.898908016783
36 of 39	1.567917310164	1.580355598359	1.557889228130	-2.106116129220
37 of 39	0.933656343560	1.895185581689	1.207728814321	1.812678768594
38 of 39	1.235547911324	0.764147533127	1.164617713111	-2.685424321892
39 of 39	2.211522828184	1.232898636426	1.954991050377	0.242695680275
$k = 40$				
1 of 40	2.092647588509	1.679897717378	2.316504492167	-2.745096770811
2 of 40	1.435804797427	2.103833062151	3.083887280200	2.724306802391
3 of 40	2.052834071080	1.773353058552	0.817275459547	0.396525532979
4 of 40	1.436197952416	2.035388189078	0.139322666224	2.476216372756
5 of 40	2.049587573020	1.784012730752	0.796140970871	-1.451450360352
6 of 40	2.256306172304	1.013924052244	1.151846047388	-2.685462497104
7 of 40	1.842916551236	0.180623923545	1.717667096429	-0.594708152322
8 of 40	2.865293228452	3.107986704011	0.007946145795	-3.005720580163
9 of 40	1.532618298446	1.724452478457	2.146841871523	-0.541805560364
10 of 40	1.397570201228	2.221838635873	1.122756929122	-0.520331268542
11 of 40	1.474233167563	1.930747354864	1.914726098051	2.606757592306

Table A.3: Coordinates of 5-dimensional codes (continued).

pt #	$\widetilde{x}_1$ (radians)	$\widetilde{x}_2$	$\widetilde{x}_3$	$\widetilde{x}_4$
12 of 40	0.881561618460	2.144252789972	1.974094044683	-1.522331521749
13 of 40	1.045620839527	1.472592316368	0.803747265005	-1.451332892050
14 of 40	1.238595028977	0.750537892203	1.180578226166	1.619139452062
15 of 40	1.197922247891	0.954368788629	1.966138158025	0.242684612375
16 of 40	1.573675335160	1.561237035914	1.583703434627	1.035476524487
17 of 40	2.207936327514	1.246407086585	1.933863833066	-1.328913895563
18 of 40	1.906044743408	2.377445111099	1.976974935390	0.456168355559
19 of 40	0.930069818186	1.908694025420	1.186601596182	-2.898896967737
20 of 40	1.194828936621	0.966053962926	1.945057865170	-1.328812208241
21 of 40	1.048945065081	1.461694936211	0.825088161422	0.396495882779
22 of 40	1.705787856163	1.037759591439	0.057705373390	-0.417285851199
23 of 40	1.088758582510	1.368239595038	2.324317194043	-2.745067120611
24 of 40	1.705394701174	1.106204464512	3.002269987365	-0.665376280833
25 of 40	1.092005080570	1.357579922838	2.345451682719	1.690142293238
26 of 40	0.885286481285	2.127668601346	1.989746606202	0.456130156486
27 of 40	1.298676102354	2.960968730045	1.423925557160	2.546884501267
28 of 40	0.276299425138	0.033605949579	3.133646507871	0.135872073426
29 of 40	1.608974355144	1.417140175133	0.994750782067	2.599787093225
30 of 40	1.744022452362	0.919754017717	2.018835724468	2.621261385048
31 of 40	1.667359486027	1.210845298726	1.226866555539	-0.534835061284
32 of 40	2.260031035130	0.997339863618	1.167498608907	1.619261131841
33 of 40	2.095971814063	1.669000337222	2.337845388585	1.690259761539
34 of 40	1.902997624613	2.391054761386	1.961014427424	-1.522453201528
35 of 40	1.943670405699	2.187223864961	1.175454495565	-2.898908041215
36 of 40	1.567917318430	1.580355617676	1.557889218963	-2.106116129102
37 of 40	0.933656326076	1.895185567005	1.207728820523	1.812678758027
38 of 40	1.235547910182	0.764147542491	1.164617718199	-2.685424298031
39 of 40	2.211522835404	1.232898628170	1.954991057408	0.242695685853
40 of 40	1.946763716969	2.175538690664	1.196534788420	1.812780445349