A Novel Approach to the Spherical Codes Problem

Simanta Gautam and Dmitry Vaintrob

Massachusetts Institute of Technology

Abstract

A spherical code is a finite set of points on the surface of a sphere in n dimensional space. The spherical codes problem asks for the maximum number of points in a spherical code where the angle between any two points with respect to the center is at least α. A traditional approach to this problem is to define an energy function for a spherical code as the sum of the inverse distance power of every pair of points and to optimize for minimum energy. However, a method to globally minimize the energy function for any given parameters is unknown, especially for spherical codes in higher dimensions. In our work, we improve the optimization by imposing certain symmetry groups on the spherical codes, as most configurations are invariant under reflection groups of certain Euclidean lattices. Furthermore, we develop a novel algorithm that individually separates pairs of points by treating them as unit vectors and applying gradient flow on their dot products. Using this approach, we were able to reproduce configurations for many of the best known spherical codes found in literature and find new configurations for dimension 6.

Executive Summary

The subject of a famous dispute between Isaac Newton and David Gregory in 1694 was, in a slightly different context, the following question: what is the maximum number of points that can be on the surface of a sphere such that any two points are at least 60° apart? Newton correctly believed the answer was 12, whereas Gregory argued that it was 13. Although this dispute was finally resolved by Schutte and van der Waerden [1] in 1953, it engendered a larger, open-ended problem in discrete geometry known as the spherical codes problem.

A spherical code is simply a finite set of points on the surface of a sphere in n dimensional space. The spherical codes problem asks for the maximum number of points in a spherical code where the angle between any two points is at least α. Some applications of finding good spherical codes (i.e., configurations of well-distributed points on a sphere) include assembling colloidal clusters [2], improving spread-spectrum communications [3], studying microbiological structures of pollens and viruses [4], and constructing Grassmannian frames (important for wireless communication and multiple description coding [5], [6]).

The purpose of this work is to develop an improved computational tool that allows us to find useful spherical codes by optimally distributing m points on a sphere in n dimensions. The known approach to this problem is to define an energy function for a spherical code as the sum of the inverse distance of every pair of points and optimize for minimum energy using gradient descent. A configuration with minimum energy corresponds to a useful spherical code (i.e., a distributed configuration of points on the suface of a sphere). However, a method to globally minimize such a function is unknown, especially for spherical codes in higher dimensions. In our approach, we studied how imposing symmetry helps optimize for minimum energy and consequently find well-distributed configuration of points on a sphere. Along with comparing different symmetry groups useful for finding optimal spherical codes, we created a hybrid program with two distinct algorithms to get a new list of spherical codes in six-dimensional space.

1 Introduction

A spherical code in dimension n with a minimal angle α is a finite set X of unit vectors such that for any distinct vectors $x, y \in X$, we have $x \cdot y \leq \cos(\alpha)$. That is, the pairwise angle between any vectors x and y is at least α. The spherical codes problem asks for the maximum cardinality of a spherical code for any given dimension n and angle α.

An important case of the spherical codes problem is the kissing number problem, where $\alpha = \pi/3$. A kissing number is the cardinality of a spherical code where any two vectors $x, y \in X$ satisfy the condition that $x \cdot y \leq 1/2$. Thus, the kissing number problem asks for the maximum cardinality of such a spherical code. Equivalently, the kissing number problem asks for the maximum number of non-overlapping spheres that can touch another sphere of the same size in some n-dimensional space. This problem in dimension three created a dispute between Isaac Newton and David Gregory in 1694. Newton correctly believed that the maximum kissing number in dimension three was 12, that 12 non-overlapping spheres could touch another sphere of same size. Gregory, however, believed the maximum kissing number in this dimension was 13. This dispute was finally resolved by Schutte and van der Waerden [1] in 1953. Table 1 gives the currently known greatest kissing numbers in other selected dimensions.

In literature, attempts at improving spherical codes have primarily been done by finding methods to maximize α for a fixed dimension and number of unit vectors. Such methods have been both computational and theoretical, incorporating a myriad of global optimization techniques as well as analytic bounds and lattice theory [7]. Our approach to the spherical codes problem starts by defining an energy function as the sum of the inverse distance of every pair of unit vectors in the spherical code, as frequently done in literature. This energy function tells us the overall distribution of the vectors in the spherical code because as we optimally distribute the vectors, we would expect the energy value of the configuration

Dimension	Highest Kissing Number
1	**2**
2	**6**
3	**12**
4	**24**
5	40
6	72
7	126
8	**240**
12	756
16	4320
24	**196560**

Table 1: The currently known lower bounds for the kissing numbers in selected dimensions. Kissing numbers known to be the largest are bolded.

to decrease. Thus, the spherical code problem turns into a global optimization problem, where we try to globally minimize the energy function to find good spherical codes. Possible approaches for global optimization include a genetic algorithm, stochastic processes like the random walk, and gradient descent. In this study, we work with gradient descent as a technique to minimize energy. The problem with using gradient descent by itself as a global optimization method, as acknowledged by Nurmela [8], is that it tends to give local minima more often than global minima. To mitigate this problem, we propose and evaluate two potential solutions: combining the gradient descent algorithm with an *angle optimization* algorithm and limiting the starting configuration of vectors to certain symmetries (e.g., antipodal symmetry).

1.1 Tammes and Thomson Problem

Although we address the spherical codes problem, which asks for the maximum number of unit vectors for a fixed dimension and α, our attempt towards a solution actually involves answering the Tammes problem, named after a Dutch botanist who proposed the problem in 1930 as he studied pores on pollen grains. The Tammes problem asks for the maximum

α for a fixed dimension and number of unit vectors. Though this problem was proposed for unit vectors in three dimensional space, we study it in even higher dimensions because of its relation to spherical codes.

The Thomson problem, also related to the distribution of unit vectors in three dimensional space, asks for the minimum energy configuration of a fixed number of electrons on the surface of a sphere repelling each other with a force given by Coulomb's law. Because our energy function is similar to that given by Coulomb's law, in the case of three dimensional space, the unit vectors are analogous to electrons and the gradient flow models the physical movement of electrons repelling amongst each other. Thus, our methods in this work can also be used to address the Thomson problem.

1.2 Motivation

Improved understanding of good spherical codes in higher dimensions has a considerable impact in error correcting codes and digital communications [9]. Bachoc, Ben-Haim and Litsyn [5] showed that the Rayleigh flat-fading single-input multiple-output (SIMO) scenarios with 1 transmit antenna and n transmitted symbols with the channel unknown to the receiver correspond to configurations of points on the sphere in $\mathbb{R}^n$ with antipodal symmetry. Thus, considering symmetries of configurations, as done in our work, not only helps find better spherical codes, but also has practical applications in information transfer.

Furthermore, the problem of distributing a configuration of unit vectors also shows up in biological sciences, such as the analysis of protein structure [10] and the creation of a numerical method to evaluate the growth and development of solid tumors [11]. By proposing a new hybrid algorithm and ideas from group theory to an existing global optimization problem, we are create an efficient way to find good spherical codes that can be applied into these fields.

2 Preliminaries

We denote the unit sphere of the Euclidean space $\mathbb{R}^n$ as S^{n-1}, namely

$$S^{n-1} := \left\{ (x_1, ..., x_n) \in \mathbb{R}^n \;\middle|\; \sum_{i=1}^{n} x_i^2 = 1 \right\}. \tag{1}$$

The dot product of two unit vectors v and w given by $v \cdot w = \sum v_i w_i$ is a natural invariant that allows us to reconstruct the angle between v and w using the formula $\cos(\alpha) = v \cdot w$. Furthermore, the distance between v and w is defined by the dot product

$$||v - w|| = \sqrt{\sum_{i=1}^{n} (v_i - w_i)^2} = \sqrt{2 - 2(v \cdot w)}. \tag{2}$$

It follows from (2) that

$$||v - w|| = \sqrt{2 - 2\cos(\alpha)}. \tag{3}$$

2.1 Energy Function

We define an energy function of a configuration of vectors $v_1, v_2, ..., v_m$ on S^{n-1} with *constant of power* p given by the equation

$$E_p(v_1, v_2, ..., v_m) = \sum_{i<j}^{m} \frac{1}{||v_i - v_j||^p}. \tag{4}$$

We vary the constant of power to control the effect of the distance of pairs of vectors on E. For example, to flow a random configuration of vectors into a more stable configuration, we start out with a small value for p (e.g., $p = 1$) and increase p as the vectors become more separated. Our algorithm strives to find a configuration of these vectors that minimizes the energy function. To do this, the program calculates the gradient of each vector $v_i = (x_{1,i}, x_{2,i}, ..., x_{n,i})$

given by

$$\nabla E(v_i) = \nabla \sum_{j \neq i}^{m} \frac{1}{\|v_i - v_j\|} = \left(\sum_{j \neq i}^{m} \frac{-2p(x_{1,i} - x_{1,j})}{\|v_j - v_i\|^{p+1}}, ..., \sum_{j \neq i}^{m} \frac{-2p(x_{n,i} - x_{n,j})}{\|v_j - v_i\|^{p+1}} \right). \tag{5}$$

The gradient of a vector v_i in a configuration gives another vector $\nabla E(v_i)$ that points towards the local maximum of the energy function of this configuration. We scale $\nabla E(v_i)$ by some $\varepsilon \approx 10^{-3}$ and take $v_i \to v_i - \varepsilon \nabla E(v_i)$ to locally minimize the energy function. By iterating this process for all the vectors in a spherical code, we expect the vectors to flow to either a configuration that gives either the local or global minimum of the energy function.

Note that minimizing the energy function does not always maximize α. Finding global minimum for the energy function solves the Thomson problem, but for the spherical codes problem, we are more concerned with maximizing α. For example, in the case of 14 unit vectors in three dimensions, the configuration with the least known energy value has $\alpha \approx 52.866°$, whereas a configuration with $\alpha \approx 55.670°$ exists, but has a larger energy value. Thus, we describe a more direct approach at maximizing α in the following subsection.

2.2 Minimizing Dot Product

Applying a similar algorithm of using gradient descent to find optimal spherical codes, we look at minimizing dot products instead of minimizing the energy function. First, two vectors, $v = (x_1, x_2, ..., x_n)$ and $w = (x'_1, x'_2, ..., x'_n)$, with the smallest pairwise angle in the configuration are found. These two vectors are then treated as vector valued functions where $v : (x_1, ..., x_n, x'_1, ..., x'_n) \mapsto (x_1, ..., x_n)$ and $w : (x_1, ..., x_n, x'_1, ..., x'_n) \mapsto (x'_1, ..., x'_n)$. Now we take the gradient of the dot product of v and w given by

$$\nabla(v \cdot w) = (w, v) = (x'_1, x'_2, ..., x'_n, x_1, x_2, ..., x_n) \tag{6}$$

To minimize the dot product, we change v to $v - \varepsilon w$ and w to $w - \varepsilon v$. Iterating this flow for new vectors with smallest pairwise angle will stabilize the configuration for the maximum α within some interval of error dependent on ε, n, and m. This algorithm works well when the value of ε is reduced as the unit vectors become more distributed. Thus, using an appropiate monotonically decreasing function for ε that is dependent on the distribution of the vectors becomes an essential part of this algorithm.

2.3 Local and Global Minima

The immediate problem with both of these algorithms is that as we increase n and m, there is an exponential rise in the number of good local minima, which results in configurations that do not always have the minimum energy value. Although an optimization that results in a local minimum energy produces a good spherical code, our goal is to be able to optimize for global minimum energy and the global maximum α to get the *best* spherical codes. The primary strategies proposed, like imposing certain symmetries described in Section 4 and combining two algorithms, are meant to curtail the occurrence of a locally optimized spherical code. With these strategies, we are able to find many spherical codes that match the best known, cumulative records [12] and come up with new spherical codes in dimension six.

2.4 Group Action

Consider a group G acting on a set of vectors V. The symmetries we consider are given by linear actions of G on $\mathbb{R}^n$.

Definition 2.1. *A representation of a group G is a group homomorphism $\rho : G \mapsto GL(n)$, where $GL(n)$ is the group of invertible $n \times n$ matrices.*

For our purposes, we can identify G with its image and think of elements of G as matrices.

We will study actions which take the unit sphere to itself, and in particular preserve distance and angle.

Definition 2.2. *The orbit of $v \in V$ is the set of vectors to which v can be moved by the elements of G. This is denoted as $Gv = \{g \circ v | g \in G\}$.*

In a symmetric configuration, keeping track of one representative of every orbit allows us to keep track of the entire configuration of vectors, which reduces the number of computations.

3 Algorithm

In this section, we describe our programs and compare the results obtained with and without antipodal symmetry in the energy optimization and angle optimization algorithms. Furthermore, we look at the *hybrid* algorithm, which first applies the energy optimization algorithm to a configuration of random vectors to get a configuration with minimal energy. Then it applies the angle optimization algorithm to this configuration to minimize α.

One of our global variables used is d: the cardinality of an orbit of a given vector. This value is also equal to the order of group G. Thus, because we want a group G with order d acting on m vectors, we randomly generate only m/d vectors $v_1, v_2, ...v_{\frac{m}{d}}$ and multiply all of the elements $I, M_1, M_2, ...M_{d-1}$ of a representation of G by the randomly generated vectors to get m vectors.

3.1 Energy Optimization Without Symmetry

The program accepts parameters n, m, ε, and p. From this, it randomly generates a set V of m vectors on S^{n-1}. After a random configuration of vectors is generated, the program iterates the following steps until a stable configuration is found (default number of iterations is set to 10^5). First, the gradient $\nabla E(v_i)$ is calculated for each vector. After all the gradients are calculated, the program flows each vector v_i to $v_i - \varepsilon \nabla E(v_i)$. Then it prints out the energy value and α of this configuration.

This program is very effective in dimensions two and three, but there is a fundamental problem in higher dimensions. This program strives to find the global minimum of E, but tends to get *stuck* in configurations yielding local minima of E.

For example, we run this program 100 times with the following inputs: $n = 4$, $m = 24$, $\varepsilon = 0.01$, and $p = 2$ to get the following results:

Occurrences	Energy	α
4	167.000	$60°$
96	167.048	$55.36937517592919°$

Table 2: Results generated from running the program 100 times.

From Table 2, we see that the program correctly finds the kissing number configuration for dimension four approximately 4% of the time. In the other 96 cases, the programs reaches some local minimum. Thus, our algorithm, without constraints from symmetry, is only effective in finding the configurations for the Thomson problem in dimension three. In higher dimensions, we need to impose some symmetry so that we limit the number of local minima of each vector.

3.2 Antipodal Symmetry

We change the program described in Subsection 3.1 by randomly generating a set V of $m/2$ vectors and for every generated vector $v \in V$, we add $-v$ to V. Note that the orbit of each vector v has two elements, namely v and $-v$. Thus, we only need to compute the gradients of $m/2$ vectors because $\nabla E(v) = r \Rightarrow E(-v) = -r$ for all $r \in \mathbb{R}$.

Therefore, adding antipodal symmetry not only allows for faster running time due to fewer computations, but also reduces the number of local minima of each vector. Running the same simulation as in Subsection 3.1, we get the following result.

Occurrences	Energy	α
100	167.000	60°
0	167.048	55.36937517592919°

Table 3: Result from running the program 100 times for inputs $n = 4$, $m = 24$, $\varepsilon = 0.01$, and $p = 2$.

By adding antipodal symmetry to the configuration of vectors, we produced the optimal kissing number configuration for S^3 more quickly and accurately than the program without antipodal symmetry. Furthermore, we were also able to find the kissing number configurations and the energy values in Table 4. Motivated by this result, we studied the effect of other symmetries on optimizing different spherical codes. In section 4, we compare the results of imposing other symmetries on the optimzation with that of imposing antipodal symmetry.

Dimension	Energy	α
2	8.75	60°
3	39.00	63.4349488°
4	167.00	60°
6	1509.00	60°
7	4593.75	60°
8	16550.00	60°

Table 4: E and α for final configurations generated by this program, $p = 2$.

3.3 Angle Optimization and Hybrid Optimization

Using the gradient of the dot product of two vector valued functions to maximize α was better than the energy optimization algorithm in that it seemed to have less local extremes. The run time and accuracy, however, were significantly worse with this program. In the energy optimization algorithm, ε does not affect the final configuration as much as it does in the angle optimization algorithm. In fact, as m increase, ε must be decreased to get a smooth flow. If ε is too small, however, then the run-time of the program increases. Thus, we vary ε with the monotonic function like $e^{-\frac{r^{3/4}}{500}}$ where r is the current number of iterations.

The hybrid optimization program that optimizes for maximal α by using the alpha optimization algorithm right after the energy optimization algorithm performed best in that there was less occurrence of getting stuck in the local extremes and the running time was reasonable given the accuracy. This program was able to match most of the currently known cumulative records on spherical codes in dimensions three, four, and five. Appendix A has the pseudocode.

4 Other Symmetry Groups

Motivated by the results produced by the addition of just antipodal symmetry on the configurations of vectors, we studied the effect of other symmetry groups acting on the configurations. We focused specifically on configuration of vectors in dimension 16. Given a group G with order d, the program was adjusted so that it began by generating a set X of m/d random vectors. Recall that d is also the cardinality of the orbit of each vector. The program then finds a representation Y of G, where $Y := \{M_1, M_2, M_3, ..., M_d\}$. To generate all the vectors, the program multiplies all the elements of Y by the vectors in V. Similar to the antipodal symmetry program, this program calculates the gradient of exactly one of the vectors v per orbit and applies the representation of G to v such that the gradient of another vector $M_i v$ in the orbit of v is given by $M_i \nabla E(v)$.

4.1 Rotational Symmetry, and Sparse Subgroups

After first imposing antipodal symmetry on the configuration of vectors, we look at the results of imposing rotational symmetry for improved optimization. Specifically, we impose rotation by 60 degrees such that for every randomly generated vector, v, there are six corresponding vectors $\{v, v_{60}, v_{120}, v_{180}, v_{240}, v_{300}\}$ in its orbit, where v_i denotes rotation of v by i degrees. Running this program in dimensions up to 24, enough data was gathered to compare the results with antipodal symmetry. The rotational symmetry was able to perform better than antipodal symmetry only for two different number of vectors in dimension six and one in dimension 12. Thus, we learned that imposing more than antipodal symmetry (which corresponds to $\{v, v_{180}\}$) results in worse performace.

Next, we compared the results obtained from antipodal symmetry with the results of some symmetry groups that we call *sparse*. A subgroup is called sparse if the coodinate representation for the group forms a sparse lattice. The motivation to try sparse subgroups arose from the existing theoretical work on the spherical codes problem and lattice theory. An example of a sparse subgroup we tried was the *Checkerboard Lattice* in dimension four. This is a subgroup of $(\mathbb{Z}/5\mathbb{Z})^2$ is $\{(0,0),(1,2),(2,4),(3,1),(4,3)\}$. This symmetry, along with a few other sparse subgroups that we have tried, have not given notably better results than antipodal symmetry. Because every spherical code is different, even within the same dimension, there is a still a potential for a symmetry to be better for a specific spherical code. Thus, we are still conjecturing and testing new symmetries that seem promising, but have not improved our initial finding that imposing antipodal symmetry with hybrid optimization algorithm is effective in finding better spherical codes. The best spherical codes for vectors in six dimensions found by our program is given in Table 6.

4.2 Spherical Codes in S^{15}

In dimension eight, the kissing number problem is solved by the E_8 lattice, defined by

$$E_8 = \left\{ x_i \in \mathbb{Z} \cup (\mathbb{Z} + \frac{1}{2})^8 : \sum x_i \equiv 0 \pmod 2 \right\}. \tag{7}$$

Number of Vectors	α
72	60°
74	58.07521958°
76	57.61988357°
78	57.438259759°
80	56.399346513°
82	56.3340096038°
84	56.0106264680°
86	55.6994474052°
88	55.2136149122°

Table 5: Results of optimal spherical codes in dimension 6.

We conjecture that an abelian group corresponding to the E_8 lattice can be incorporated in one of our optimization algorithms to find improved spherical codes in S^{15}. An example of a group that corresponds to the E_8 lattice is the subgroup H of $(\mathbb{Z}/4\mathbb{Z})^8$ satisfying the following conditions for any $(x_1, x_2, ...x_8) \in H$

1. $\sum x_i \equiv 0 \pmod 4$
2. $x_i + x_{i+1} \equiv 0 \pmod 2 \ \forall\ i \in (1, 2, ..., 7)$

The representation of H is a set of 16×16 matrices that have cardinality 256. An 8-tuple is mapped to a 16×16 matrix by putting eight 2×2 matrices along the diagonal where each number in the 8-tuple corresponds to the 2×2 matrix defined by the mappings below:

$$0 \mapsto \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad 1 \mapsto \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad 2 \mapsto \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}, \quad 3 \mapsto \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

If we randomly generate 17 vectors and find all the other points in the orbit of those 17 vectors using the representation of H, then we have $17 \times 256 = 4342$ vectors in our configuration. We have not fully tested this particular configuration of vectors using one of our programs due to long run-time, but we conjecture that either this, or more likely another abelian group acting on a configuration of vectors can improve the lower bound of the kissing number problem in S^{15}.

5 Conclusion

As a computational approach to the spherical codes problem, we began by defining an energy function that allowed us to treat each vector as a point charge. From this perspective, every configuration of vectors on S^{n-1} has some energy-value and our goal was to minimize the energy value so that we maximize the minimum pairwise angle in the configuration. Using our energy minimization algorithm with antipodal symmetry, we were able to globally minimize the energy and thus maximize the angle for many cases up to dimension eight. To strictly maximize the minimum pairwise angle in the configuration, we created another algorithm that took into account the gradient of the dot product of two vector valued functions. With this algorithm, we were able to decrease the number of good local extremes at a cost of less accuracy and slower run-time. We then created a hybrid algorithm that incorporates both the energy minimization algorithm and the angle maximization algorithm.

With this hybrid algorithm, we were able to match currently known cumulative results for many cases of the spherical codes problem along with creating a list of new spherical codes in dimension six found in literature. Furthermore, we conjectured that incorporating a sparse abelian subgroup of $(\mathbb{Z}/k\mathbb{Z})^{\frac{n}{2}}$ for $k \in \mathbb{N}$, acting by rotating pairs of coordinates with our program may not only help with improving spherical codes in dimension 16, but also in other dimensions.

6 Further Work

The problem of determining the *best* spherical codes for any dimension n and cardinality m is still open in mathematics. With our hybrid algorithm and implementation of certain symmetry groups, we improve on known computational tools to determine good spherical codes by offering a new approach with good run time that matches and improves existing results found in literature. Our approach could be further improved by considering the following:

1. A classification of other symmetry groups that, when implemented with our program, would produce good spherical codes in higher dimensions would be interesting and feasible with more experimental data. In small dimensions, most of the spherical codes from our program match best known spherical codes, but in higher dimensions, it may be possible to further improve certain known spherical codes by implementing the right symmetry group with our program.

2. In our work, we use gradient descent as an approach to minimize energy. There are, however, other approaches of minimization like the genetic algorithm and stochastic processes that could be used with the symmetry groups we proposed. It would be interesting to see the results with these other approaches of optimization.

3. A more detailed study of specific global and local minima per dimension, as we did with 24 vectors in four dimensional space, would be important because this knowledge would potentially help with global optimization.

4. As with many optimization problems, there is a great potential of finding applications of good spherical codes in various fields. In fact, the discoveries of applications into fields like quantum physics are fairly recent. Thus, a good study would be to find connections between distributing unit vectors (in any dimension) to another field like microbiology, where spherical codes help understand structures of certain viruses.

7 Acknowledgements

I would like to extend my gratitude to the Center for Excellence in Education, Research Science Institute, and Massachusetts Institute of Technology for providing the opportunity and place to work on this research project. I would also like to thank Dr. Abhinav Kumar for the research idea, Dr. Tanya Khovanova for helpful suggestions, and my mentor Dmitry Vaintrob for his guidance throughout this research. Furthermore, I want to thank my tutor Dr. John Rickert for his editing and advice.

References

[1] K. Schutte and B. van der Waerden. Das problem der dreizehn kugeln. *Math. Ann.*, 125, 1953.

[2] C. Phillips, E. Jankowski, M. Marval, and S. Glotzer. Self assembled clusters of spheres related to spherical codes. preprinted arXiv:1201.5131v1, 2011.

[3] Z. Utkovski and J. Lindner. On the construction of non-coherent space time codes from high-dimensional spherical codes. *Spread Spectrum Techniques and Applications Conference*, pages 327–331, 2006.

[4] D. Weaire and T. Aste. *The Pursuit of Perfect Packing.* Taylor & Francis Group, 2008.

[5] C. Bachoc, Y. Ben-Haim, and S. Litsyn. Bounds for codes in products of spaces, grassmann and stiefel manifolds. *IEEE Transactions Information Theory*, 54:1024–1035, 2008.

[6] J. B. Sandrine Vialle. Performance of optimal codes on gaussian and rayleigh fasing channels: a geometrical approach. *in Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing*, 1999.

[7] J. Conway and N. Sloane. *Sphere Packings, Lattices, and Groups.* Springer-Verlag, 1998.

[8] K. J. Nurmela. Constructing spherical codes by global optimization methods. Research Report A32, Helsinki University of Technology, Department of Computer Science and Engineering, Digital Systems Laboratory, Espoo, Finland, 1995.

[9] T. M. Thompson. *From Error-Correcting Codes Through Sphere Packings to Simple Groups.* Math. Assoc. Amer., 1984.

[10] R. Morris, R. Najmanovich, and A. Kahraman. Real spherical harmonic expansion coefficients as 3d shape descriptors for protein binding pocket and ligand comparisons. *Bioinformatics*, 21:2347–2355, 2005.

[11] M. Chaplain, M. Ganesh, and I. Graham. Spatio-temporal pattern formation on spherical surfaces: numerical simulation and application to solid tumour growth. *Journal of Mathematical Biology*, 42:387–423, 2001.

[12] N. J. A. Sloane. Spherical codes: Nice arrangements of points on a sphere in various dimensions. Available at `http://neilsloane.com/packings/index.html#I` (2012/07/30).

A One of the Algorithms Created

Alg. 1 Hybrid Algorithm

```
Input: dimension $n$, number of vectors $m$, group $G$, constant $\varepsilon$
Output: good spherical code, minumum angle $\alpha$, Energy $E$
foreach $0 < i < m/d$ do
    Randomly choose $v_i \in \mathbb{R}^n$ with $\|v_i\| = 1$
    add $v_i$ to setOfOrbits[]
end
foreach $g \in G$ do
    foreach $v \in$ setOfOrbits[] do
        add $g \circ v$ to setOfVectors[]
    end
end
while Configuration is unhappy do
    foreach $v \in$ setOfOrbits[] do
        $v \mapsto \frac{v - \varepsilon \nabla E(v)}{\|v - \varepsilon \nabla E(v)\|}$
    end
    foreach $g \in G$ do
        foreach $v \in$ setOfOrbits[] do
            add $g \circ v$ to setOfVectors[]
        end
    end
    if $\Delta E / \Delta r \approx 0$ then
        Configuration is happy
    end
end
Change $\varepsilon$ to 0.01
while $b \leq 100,000$ do
    foreach $v_i, v_j \in$ setOfVector[] do
        if $v_i \neq v_j$ and $v_i \cdot v_j <$ leastdotprod then
            vector1 = $v_i$
            vector2 = $v_j$
            leastdotprod = $v_i \cdot v_j$
        end
    end
    $vector1 \mapsto vector1 - \varepsilon vector2$
    $vector2 \mapsto vector2 - \varepsilon vector1$
    Reduce $\varepsilon$ accordingly

end
Print spherical code, $\alpha$, and $E$
```