Rose-Hulman Undergraduate Mathematics Journal

Volume 18
Issue 2

Article 5

A New Computationally Efficient Method for Spacing n Points on a Sphere

Jonathan Kogan
Columbia Grammar and Preparatory School, New York, jkogan18@cgps.org

Rose-
Hulman
Undergraduate
Mathematics
Journal

A NEW COMPUTATIONALLY EFFICIENT METHOD FOR SPACING n POINTS ON A SPHERE

Jonathan Kogan[a]

Volume 18, No. 2, Fall 2017

Sponsored by

Rose-Hulman Institute of Technology
Department of Mathematics
Terre Haute, IN 47803
mathjournal@rose-hulman.edu
scholar.rose-hulman.edu/rhumj

[a]email: jgkogan99@gmail.com

Rose-Hulman Undergraduate Mathematics Journal
Volume 18, No. 2, Fall 2017

A NEW COMPUTATIONALLY EFFICIENT METHOD FOR SPACING n POINTS ON A SPHERE

Jonathan Kogan

Abstract. The problem of equally spacing n points on a sphere is impossible in general, but there are methods that come close to spacing the points equally. The method introduced in this paper uses a spiral that was found using experimental evidence. The resulting spacings are close to theoretical bounds, and the method is computationally efficient for large numbers of points. The method's accuracy ranges from 70% to 86% of the upper bound as n changes.

Acknowledgements: Thank you to Todd Rowland and the Wolfram Mentorship program for supporting me through the research process. I would also like to thank my math teacher, Matthew Hoek, for advice in writing and editing the paper. Additionally, I would like to thank my English teacher, Jayne Connell, for help with editing. Lastly, I would like to thank the editor and referee for their detailed suggestions.

1 Introduction

The problem of spacing n points equally on a sphere is impossible to solve in general, but it is possible to develop methods that come close. Figure 1 illustrates the configurations with equally spaced points for $n = 2, 3, 4$, and 6. For $n = 2$, the configuration is 2 points on opposite poles; for $n = 3$, the configuration is an equilateral triangle; for $n = 4$, the configuration is a tetrahedron; for $n = 6$, the configuration is an octahedron. When $n = 5$, though, there is a configuration—the trigonal bipyramid—that is known to have its points spaced in the "best" possible configuration (that is, the sum of the distances between each pair of points is maximized), but the spacing is not perfect because adjacent points are not all separated by the same distance. In this case the two different distances between adjacent points are $\sqrt{2}$ and $\sqrt{3}$. Figure 2 illustrates this $n = 5$ configuration. For most larger n's, the "best" configurations are unknown [7].

Figure 1: 2, 3, 4, and 6 points equally spaced on a sphere

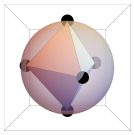

Figure 2: 5 points spaced as equally as possible on a sphere

In this paper we will discuss a new computationally efficient method for spacing n points on a sphere. Section 2 will address prior and similar formulations of the problem. Section 3 will discuss how we created our new method of spacing n points on a sphere. Section 4 will evaluate the accuracy of the new method relative to the Golden Spiral Method. And Section 5 will discuss applications of spacing n points equally on a sphere and topics for future research.

2 Background

This section will analyze similar and prior formulations of the problem of spacing n points, as equally as possible, on a sphere.

2.1 The Thomson Problem

The problem of spacing n points equally on a sphere is one that has intrigued scientists for over a century, starting with J.J. Thomson, the man who discovered the electron. When

Thomson was working on this problem, he thought that electrons were arranged in terms of rigid electron shells around a spherical atom and was trying to figure out how to space n repulsive points around a sphere. These solutions tend to have points spaced as equally as possible. Now we know that his theory of electrons was incorrect, but the Thomson Problem still remains relevant today [1].

2.2 Circle Packing of Spheres

Another similar problem is circle packing of spheres. Circle packing of spheres is the problem of spacing n circles on a sphere with no overlapping circles where the edges all touch one another. Many solutions have circles with center points that are the same as solutions to the problem of n points equally spaced on a sphere. This can be seen in Figure 3 [2].

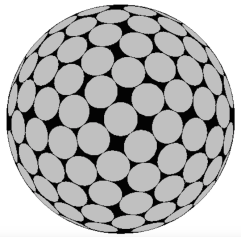

Figure 3: Circle packing of spheres

2.3 N Equally Spaced Points on a Circle

The problem's 1-dimensional counterpart is n equally spaced points on a circle. This can be seen in Figure 4 [4].

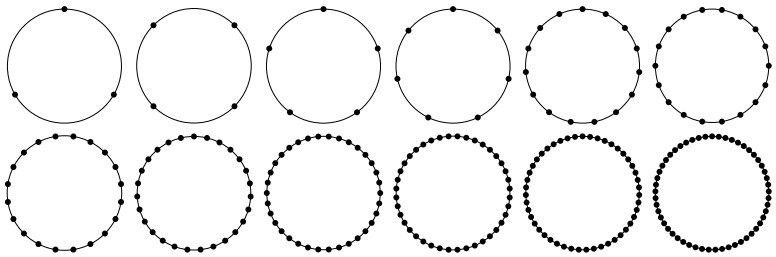

Figure 4: N equally spaced points on circles

The problem of placing n points on a circle so that the distance between consecutive points is the same can be solved by looping over $\{\cos(\theta), \sin(\theta)\}$ from 0 by $2\pi/n$ until the angle θ equals $2\pi - 2\pi/n$. One might think that we can easily carry this solution for the problem on a circle to the problem for a sphere, but unfortunately we cannot. This idea for spacing points along a curve, though, is central to the method in this paper.

2.4 Calculus Methods

There exist calculus-based methods like the ones used for the Thompson Problem. These calculus methods put points on a sphere and then continuously adjust them to make the correct configuration. So, all of the points continuously repel one another and adjust their location based on the location of the other points until they reach the best configuration. While these methods work quite well, they are also slow for large n since each of the points is slightly adjusting itself based on the location of all of the other points over and over again. For example, if there are 500 points, then each of those points continuously adjusts itself based on the location of the other 499 points. An interactive example can be found here: http://thomson.phy.syr.edu/thomsonapplet.php [1].

The next two methods we will discuss, the Icosahedron Interpolation Method and the Golden Spiral Method, are two fast non-calculus methods that have been created to space points equally on a sphere.

2.5 The Icosahedron Interpolation Method

The Icosahedron Interpolation Method is a method that works only for some n [9]. Points are placed within the triangular faces of an icosahedron and then are normalized to be on the sphere. Figure 5 shows what one of the twenty triangles looks like.

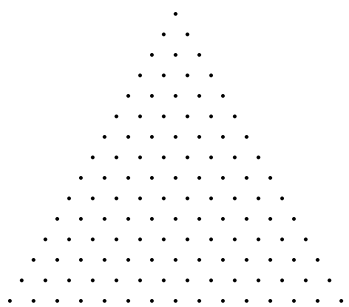

Figure 5: Triangle with hexagonally interpolated points

In Figure 6 the triangles are placed on the sphere and points are normalized.

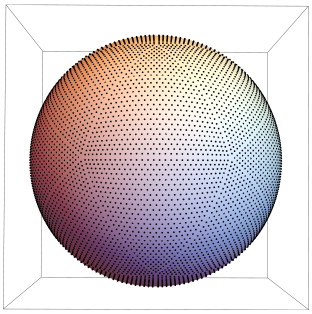

Figure 6: Icosahedron Interpolation Method

The Icosahedron Interpolation Method has two problems:

1. It works only for n's that satisfy the equation $20 * (j^2 + j)/2 = n$ with integer j's. In other words, it works only for n's that when divided by 20 are triangle numbers.

2. The other problem with this method is that the distances between the triangles are too large. Within the triangles, every point is the same distance from the next, but that is not true of the points separating the triangles because they are aligned in a square way. So the distance between two adjacent points is the same as within the triangles and is ideal, but the diagonal distances are the smallest distance times $\sqrt{2}$. Because so many regions have this extra length added in, it takes space that could be occupied by the well-spaced points and lowers the overall accuracy.

2.6 The Golden Spiral Method

The Golden Spiral Method is the most successful of the previously created methods [3]. This method plots the points around the golden spiral. In two dimensions, it works by spiraling points outward from the center. The points are rotated by the golden angle, $\pi(3 - \sqrt{5})$, and the distance the point lies from the center is in constant proportion to its area. For a sphere, the method is then extrapolated. In the (x, y) plane, the points are continually rotated by the golden angle, and then the angle of rotation along the z axis is proportional to the surface area of the sphere. Figure 7 shows the Golden Spiral Method in two and three dimensions.

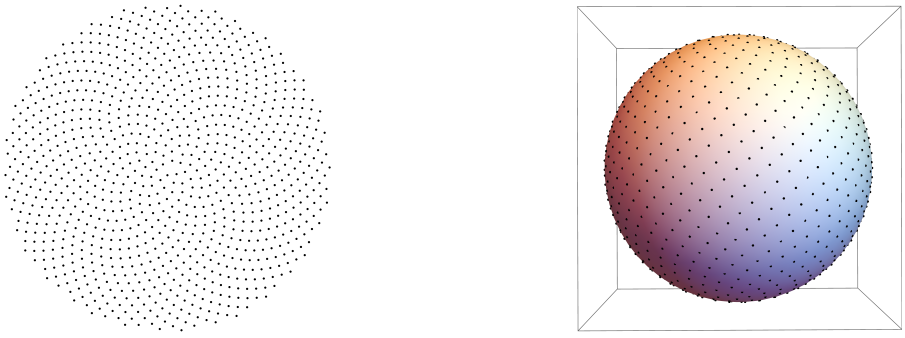

Figure 7: 2D and 3D Golden Spiral Method

As seen in Figure 7 in contrast to the Icosahedron Interpolation Method, the Golden Spiral Method appears to have every point spaced equally.

3 Method

This section will describe how we experimentally found our new method of spacing points, as equally as possible, on a sphere.

We first created a spiral around a sphere by defining a function in two dimensions and then extending it to three dimensions. The two-dimensional function looped over $\{s, \pi/2 * \text{signum}(s) * (1 - \sqrt{1 - |s|})\}$ with s going from -1 to 1 in steps of $2/(n-1)$. Note that signum is a function that returns the sign of its argument. It returns -1 if the argument is negative, $+1$ if the argument is positive, and 0 if the argument is 0. Here we will explain the function.

The $2/(n-1)$ makes the function actually generate n points. Because the iterator goes from -1 to 1 rather than 0 to 1, there are double the points plus the point when it is 0. So the 2 just doubles the increment and subtracts 1 to make $\{s, \pi/2 * \text{signum}(s) * (1 - \sqrt{1 - |s|})\}$ actually happen n times with a different s ranging from -1 to 1 making n distinct points.

The s in the x coordinate increases from -1 to 1 in steps of $2/(n-1)$.

The $\pi/2 * \text{signum}(s) * (1 - \sqrt{1 - |s|})$ in the y coordinate looks complicated, but is simpler than it seems. The factor, $(1 - \sqrt{1 - |s|})$, gives the curved shape to the function. The second part, signum(s), simply keeps the sign of s. In the previous part, we are taking the absolute value so the s loses its sign; signum(s) returns the sign to the number. The significance of this is that when the s is a negative number, without signum(s), the negative s values would appear positive; but with signum(s), the function retains the sign. The last part, the $\pi/2$, puts the function in terms of π and is not important to the two-dimensional model, but when the function is eventually put into three dimensions, the $\pi/2$ makes the points wrap entirely around the sphere. Figure 8 shows the function in two dimensions.

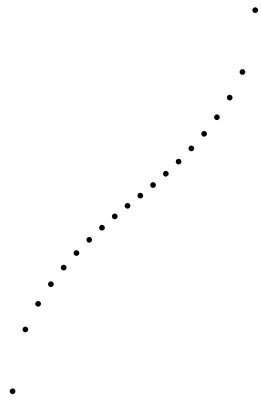

Figure 8: New Spiral Method in 2D

We then put the function into three dimensions using spherical coordinates. The two

angle inputs for this are the x and y from the two-dimensional coordinates in Figure 7. The function for spherical coordinates applied to every point is

$$\text{SphericalCoordinate}(x, y) = \{\cos(x)\cos(y), \sin(x)\cos(y), \sin(y)\}.$$

Applying the SphericalCoordinate function to our previous two-dimensional curve, $\text{SphericalCoordinate}(s, \pi/2 * \text{signum}(s) * (1 - \sqrt{1 - |s|}))$, with s again ranging from -1 to 1 in steps of $2\pi/(n-1)$, we obtain Figure 9.

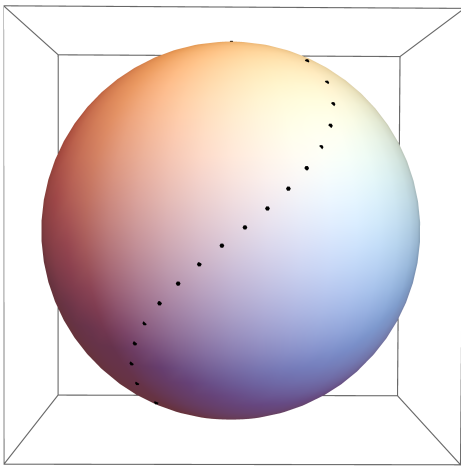

Figure 9: New Spiral Method in 3D

The spiral did not come close to equally spacing the points, but the points did spiral around a sphere from the bottom pole to the top pole, indicating that the spiral had potential. To change the configuration of these points, we manipulated the s variable through multiplication. To manipulate the s, we introduced a new variable x. The inputs to the SphericalCoordinate function changed accordingly:

$$\text{SphericalCoordinate}(s * x, \pi/2 * \text{signum}(s) * (1 - \sqrt{1 - |s|})).$$

Figure 10 shows $\{n, x\}$ pairs being tested. Some are better than others.

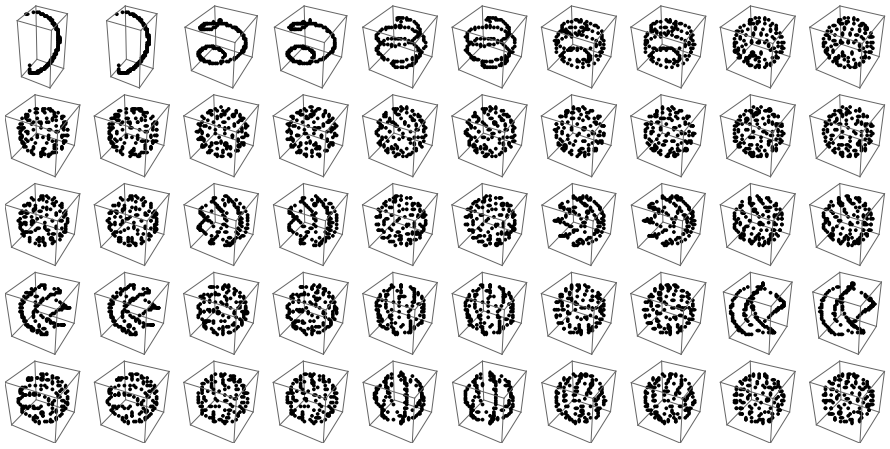

Figure 10: $\{n, x\}$ pairs plotted

When we changed the n and x values, the shape of the spiral changed too. Some $\{n, x\}$ pairs appeared to be approximately spherical providing insight that some $\{n, x\}$ pairs are better than others. This meant that we needed to find and analyze the superior $\{n, x\}$ pairs. Figure 11 shows those superior $\{n, x\}$ pairs, which were found by inspection:

```
{{107, 93}, {106, 93}, {105, 90}, {104, 90}, {103, 89}, {102, 88}, {101, 87},
 {100, 86}, {99, 85}, {98, 84}, {97, 83}, {96, 83}, {95, 82}, {94, 81},
 {93, 80}, {92, 79}, {91, 78}, {90, 77}, {89, 76}, {88, 75}, {87, 75}, {86, 74},
 {85, 73}, {84, 78}, {83, 77}, {82, 76}, {81, 75}, {80, 74}, {79, 73}, {78, 72},
 {77, 71}, {76, 70}, {75, 69}, {74, 68}, {73, 67}, {72, 66}, {71, 57}, {70, 56},
 {69, 55}, {68, 54}, {67, 54}, {66, 53}, {65, 52}, {64, 51}, {63, 50}, {62, 49},
 {61, 49}, {60, 48}, {59, 47}, {58, 46}, {57, 45}, {56, 45}, {55, 44}, {54, 43},
 {53, 42}, {52, 41}, {51, 41}, {50, 40}, {49, 39}, {48, 38}, {47, 37}, {46, 37},
 {45, 36}, {44, 35}, {43, 34}, {42, 32}, {41, 31}, {40, 30}, {39, 29}, {38, 27},
 {37, 24.5`}, {36, 23}, {35, 23}, {34, 22}, {33, 21}, {32, 20}, {31, 19}, {30, 35}}
```

Figure 11: "Superior" $\{n, x\}$ pairs

Figure 12 shows those same superior $\{n,x\}$ pairs plotted:

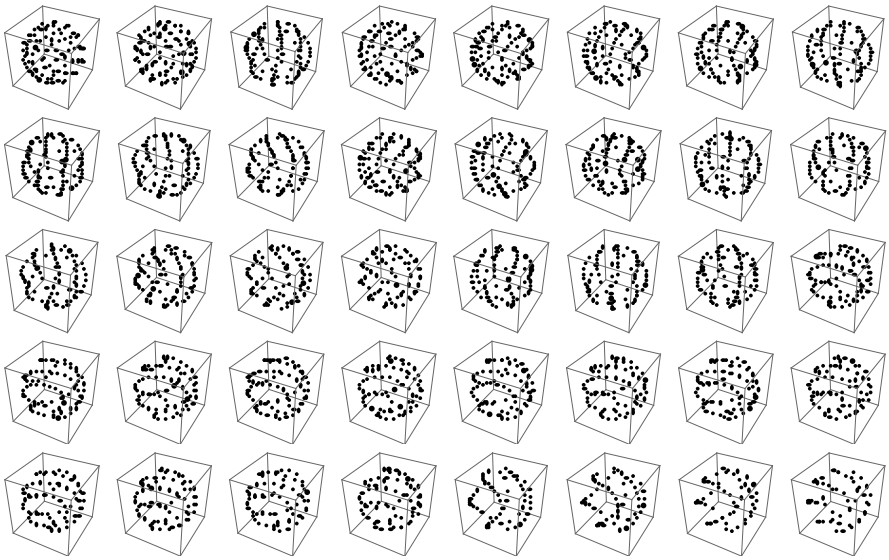

Figure 12: "Superior" $\{n, x\}$ pairs plotted

The configurations of the superior pairs showed promise and for the most part plotted points well. One main problem emerged though: there was clumping at the poles. We fixed the clumping by stopping the spirals right before they reached the poles. In the new function, the difference is that the iterator does not go from -1 to 1 anymore; it instead goes from $-1 + 1/(n-1)$ to $1 - 1/(n-1)$ in steps of $(2 - 2/(n-1))/(n-1)$.

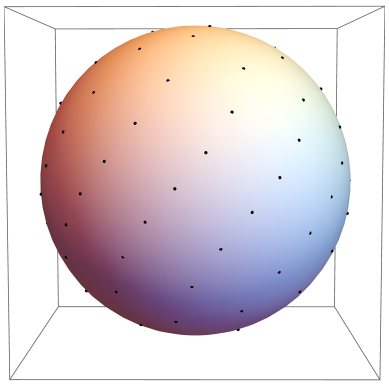

Figure 13: A "superior" pair as input for the new method

Having resolved the clumping problem, the new challenge was finding the optimal $\{n, x\}$ pairs and a function that would produce them optimally. To analyze the data, we plotted the $\{n, x\}$ pairs.

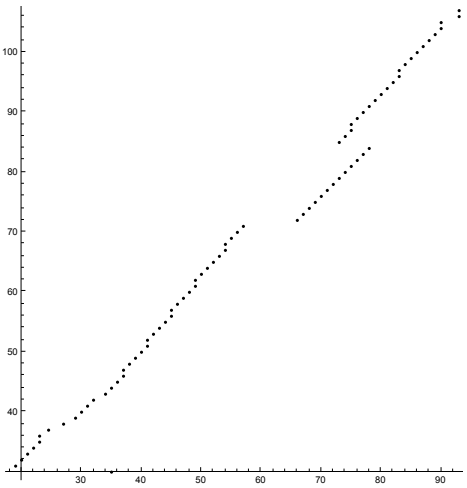

Figure 14: "Superior" pairs plotted

The graph showed promise because of its linear appearance. This meant that there was a high probability of there being a linear function that would provide an x for every n. To find this linear function analytically, we first created three new functions to analyze how well the points were spaced on a sphere: SmallestDistance, TheoreticalSmallestDistance, and NormalizedSmallestDistance.

SmallestDistance(pts) takes a list of points and identifies the smallest Euclidean distance between any two points on the sphere. In other words, it identifies the two points that are closest together and outputs the distance between the two.

The TheoreticalSmallestDistance(n) function returns the upper bound for the distance of n equally spaced points on a sphere. It is well known that the optimal spacing of points on the plane is the hexagonal grid. To find the upper bound for the distance of n points on a sphere, we imagine there is a planar hexagonal grid that covers the entire area of the sphere. Using this grid, we find this upper bound for all n values based on the size and the number of equilateral triangles.

Lastly, we defined the NormalizedSmallestDistance(pts) function as the SmallestDistance(pts)/ TheoreticalSmallestDistance($Length(pts)$); the function provides a relative accuracy for the points that are around the sphere.

Using these functions, we now could find optimal x values for n values. We used the SmallestDistance(pts) function to find the optimal x value for $n = 200$. We first used the function to find the smallest distance in all of the $\{n, x\}$ pairs for $n = 200$ with x going from 200 to 250 in steps of 0.1. We then took the $\{n, x\}$ pair with the largest smallest-distance, which also would have the best configuration. The pair with the best configuration when n was 200 was $\{200, 240.1\}$.

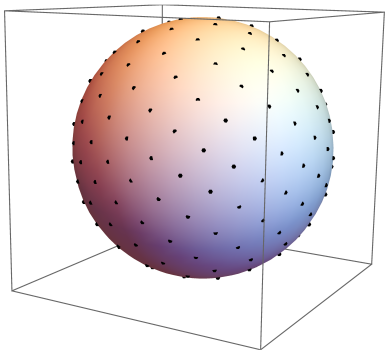

Figure 15: Optimal configuration when $n = 200$

Earlier we determined that the $\{n, x\}$ function was most likely linear with a slope a little over 1. We used that knowledge to find the slope of the line that passed through the point $\{200, 240.1\}$. We tried many different slopes from 0.4 to 1.6 by subtracting them from 240.1, and subtracting 1 from 200. The additional points we tried had the form $\{199, 240.1 - k\}$ where k is the trial slope. Then we took the two points and fit a linear function to them. Afterwards, we determined which $\{n, x\}$ function actually had the correct slope by using an input that was very far away from 200. We did this because all of the functions we tested would return the same optimal configuration for 200 points, but not necessarily for 3713 points. Figure 16 shows the functions plotted:

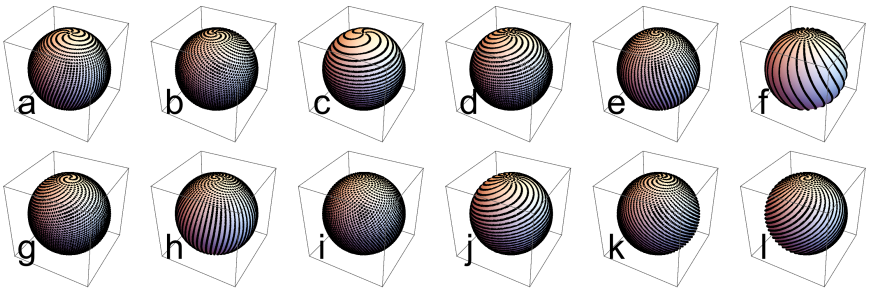

Figure 16: New method with different x functions plotted

After seeing the functions' outputs plotted, sphere i's function appeared to provide an optimal configuration of points because the points seem to be the most evenly spaced. To narrow down what the true function was, we repeated that same process for functions with more specific slopes. Eventually, we had twelve very accurate functions.

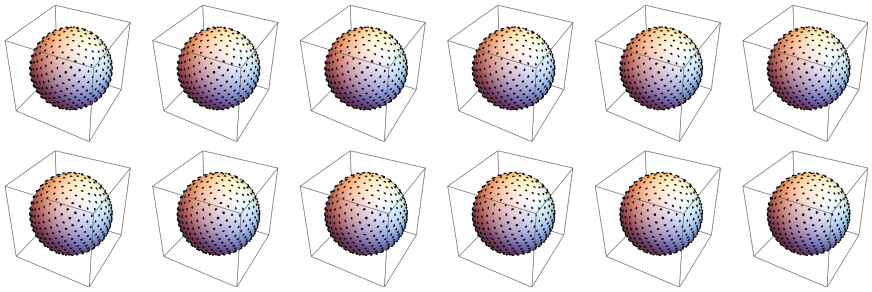

Figure 17: New method with different x functions plotted

To determine which function was best, we used the NormalizedSmallestDistance(pts) function to find the optimal function to provide an x for every n. In Figure 18, we plotted these ten accurate functions as they went from 100 to 2000 in steps of 50. We found that the function with the accuracy that stayed the highest most consistently was the best.

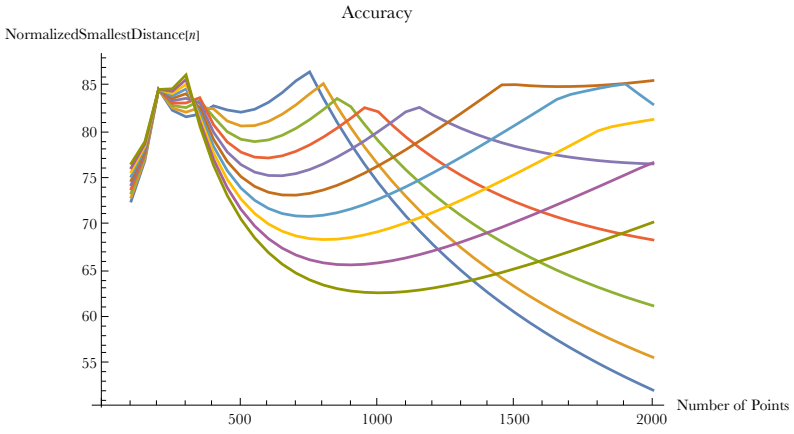

Figure 18: Accuracy plotted for different x functions

In Figure 18, the function appeared to be $x = 0.12 + 1.1999n$, but that was not the case because its next curve in the accuracy plot sank very low. Unfortunately, it would be too computationally intensive to provide a larger graph for all of the functions, but we were able to expand the graph so that it went to 16000 points for the function $x = 0.12 + 1.1999n$. It appears in Figure 19.

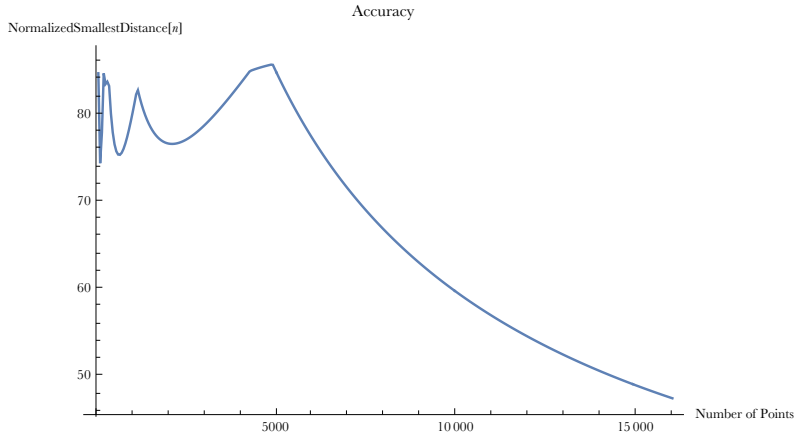

Figure 19: $0.12 + 1.1999n = x$ accuracy plotted

As seen in Figure 19, $x = 0.12 + 1.1999n$ has an accuracy that sinks, but other functions like $x = 0.1 + 1.2n$ and $x = 0.08 + 1.2001n$ have accuracies that stay high longer and more consistently. Even though time did not allow for collecting the data for a whole graph, we can infer that all of the functions would have their worst accuracies at approximately 16,000 points. Once again we relied on visual observation as seen in Figure 20.

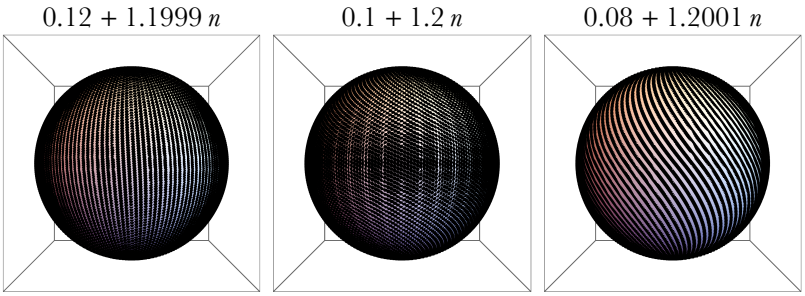

Figure 20: Best three functions at worst n

We found the experimentally determined function to be $x = 0.1 + 1.2n$. To confirm that the method was accurate for the general n-point case, we ran the function on many n's. Figure 21 shows a few of them.

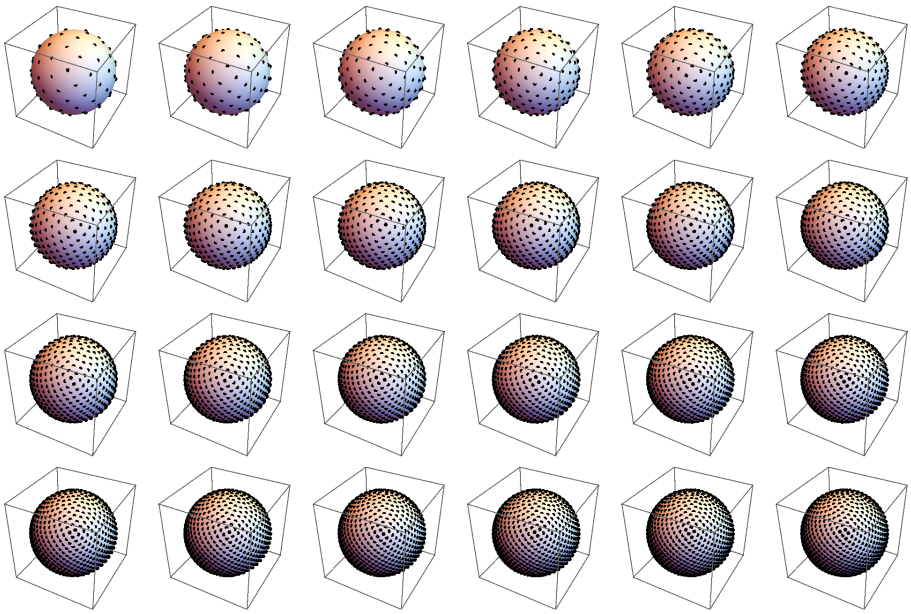

Figure 21: New Spiral Method run on many n's

The code for the new method can be found in the appendix.

4 Results and Analysis

This section further analyzes the success of the new method relative to the Golden Spiral Method.

As stated earlier, finding a general solution that can space any number of points equally on a sphere is impossible, but we can get close. We can measure the accuracy of the function by dividing the smallest distance between any two points on the sphere by the theoretical smallest distance for that number of points. Figure 22 shows the accuracies of the new method from 50 to 1200. Please note that we computed the accuracy at every 20th configuration because we had to make the code faster.

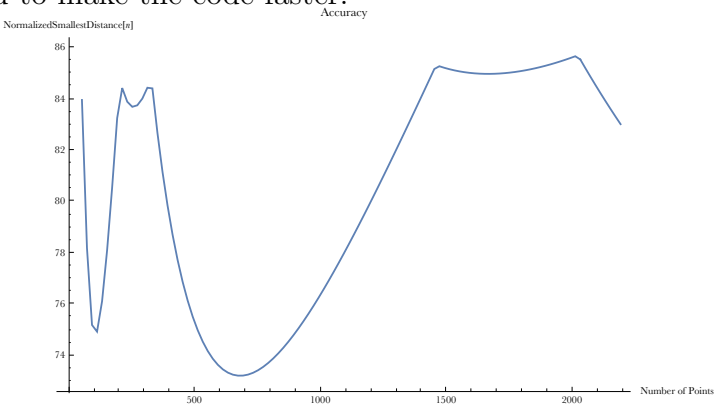

Figure 22: New Spiral Method's accuracy plotted from 50-1200

Figure 22 shows accuracies that fluctuate in a wavelike pattern. Figure 23 shows the same accuracies as Figure 22 but goes to higher n's because we only compute the NormalizedSmallestDistance of every 50th n, allowing the code to be faster.

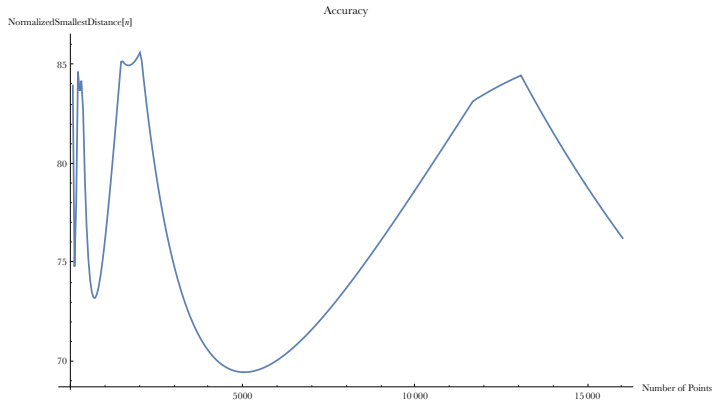

Figure 23: New Spiral Method's accuracy plotted from 50-16000

Figure 24 shows NormalizedSmallestDistance(n)'s for the Golden Spiral Method. The results are much more predictable; the curve seems to converge somewhere above 84% accuracy.

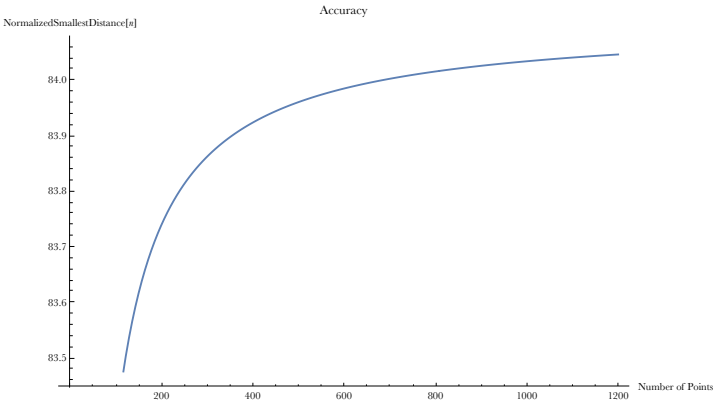

Figure 24: Golden Spiral Method's accuracy plotted

The two methods both achieve high accuracies, and it is difficult to distinguish between the two by simply looking at their configurations. Despite generating such similar configurations, though, the fluctuations in their accuracies as n increases are quite different. One's accuracy has a wavelike pattern; the other's accuracy appears to converge. Figures 25 and 26 show the two methods side by side for different n values. The four configurations in Figure 25 are from the new spiral method; The four configurations in Figure 26 are from the Golden Spiral Method.

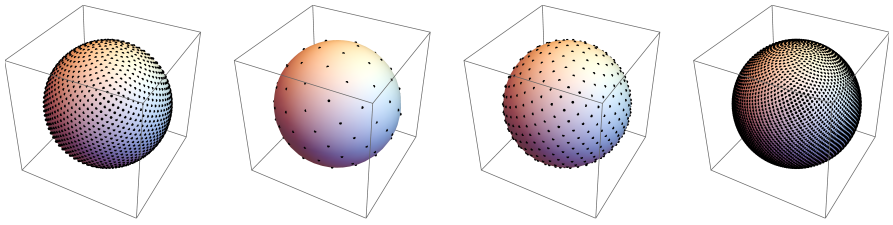

Figure 25: New Spiral Method

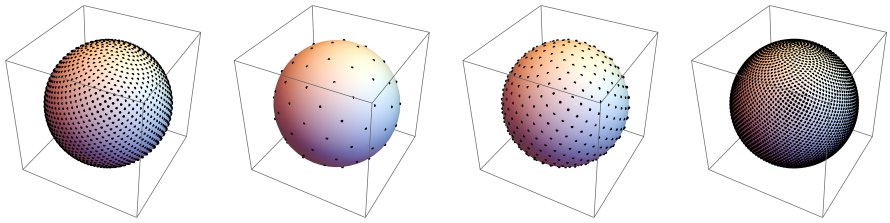

Figure 26: Golden Spiral Method

5 Conclusion

This section provides an overview of the methods in the paper and discusses applications of the method and topics for future research.

5.1 Overview and Applications

This paper discusses a new method of spacing n points as equally as possible on a sphere, quickly, and accurately, despite the fact that the problem is impossible to solve in general and most optimal configurations are unknown. The new method was designed to be efficient and falls into the regime of the faster methods, such as the Golden Spiral Method and the Icosahedron Interpolation Method. All three return points in similar amounts of time. As stated before, the calculus-based methods are slow because rather than being a function that takes the number of points as an input and then returns n points, they distribute n points randomly and then work to maximize the smallest distance by having the points continually repel until they reach it. The new method is another alternative to the slower calculus-based methods.

There exist many potential applications of n points spaced equally on a sphere. Some applications extrapolated from the Thomson Problem include multielectron bubbles in superfluid helium, virus morphology, protein s-layers, and coding theory. Spacing n points on a sphere is also equivalent to many other problems in biology, math, physics, and computer science, and can be applied to problems like structural chemistry, the design of multibeam laser implosion devices, and the optimum placement of communication satellites [1].

There are also many other interesting applications. Spacing points on a sphere can be used in designing satellites. In the Starshine 3 Student Satellite Project engineers had to distribute mirrors equally around a spherical satellite [3].

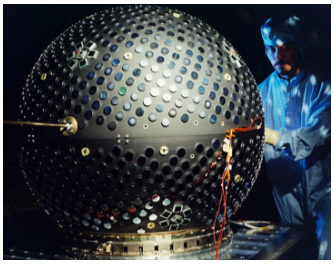

Figure 27: Starshine 3 Student Satellite

This method also can be used to model the famous Handshake Problem (if there are n people in a room, how many handshakes occur?) in three dimensions. Figure 28 shows every possible handshake among the people.

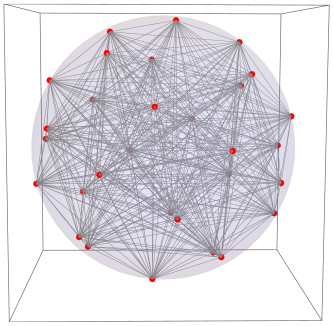

Figure 28: Handshake Problem shown on a sphere

In addition to those applications, the function can be used to make beautiful art, as seen in Figure 29, by using the function recursively on itself. First, the function was run and the equally spaced points were generated. Spheres were placed at the location of each point that made it so there were no visible openings in the structure. The process was then repeated, but this time, rather than putting spheres at the location of each point, we placed the structures from the previous iteration at the location of each point creating a new structure. Then, one more time the process was repeated, but using the newest structure.This process generated Figure 29.

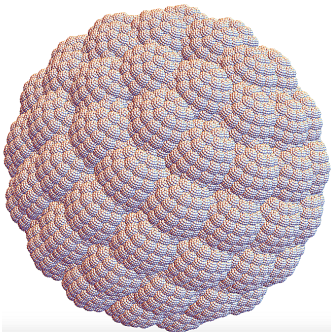

Figure 29: New Spiral Method used recursively

One other fun thing that can be done with the function is to use it for 3D printing. Figure 30 shows a 3D printed sphere with equally spaced points on it (Kogan).

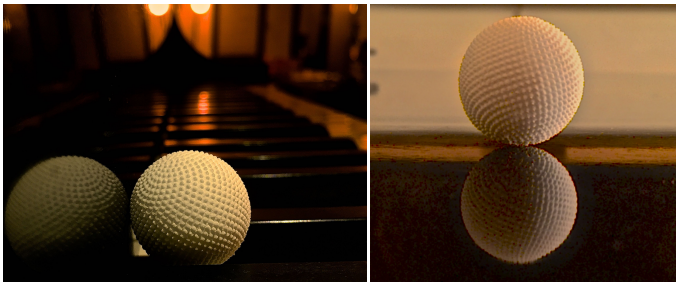

Figure 30: Pictures of New Spiral Method printed

The method can also be applied in developing brain scanning devices because they require a large number of points spaced equally on a hemisphere. The research discussed in this paper has also led to the creation of the Mathematica function SpherePoints[n], which was added in the Mathematica 11.1 update [10]. This Mathematica function enables anybody who wants points spaced equally on a sphere to generate them quickly and easily [8].

5.2 Future Research

Much research remains to be done concerning the problem of spacing n points equally on a sphere. One such topic is the accuracy pattern that this new function has. Methods like the Golden Spiral Method have much more predictable accuracies, so it would be interesting to research why the new method has this wavelike pattern.

Another future research topic involves finding additional methods of solving this problem. There are already a few, which have varying levels of accuracy, yet they all appear to be accurate to the eye. Finding more of these methods may lead to a deeper understanding of the problem.

With everything said, the method is a fast and accurate way of approximating a solution to an unsolvable task that has intrigued scientists for over a century.

References

[1] Bowick, Mark, Cris Cecka, Luca Giomi, Alan Middleton, and Kevin Zielnicki. *Thomson Problem - Points on a Sphere.*. Syracuse University, ND. http://thomson.phy.syr.edu/

[2] Buddenhagen, James. Kottwitz, D.A. *Packing Equal Circles on a Sphere.* Budden Books. N.A.

[3] Carlson, Christopher. *How I Made Wine Glasses from Sunflowers.*. Wolfram Research. N.D. http://blog.wolfram.com/2011/07/28/how-i-made-wine-glasses-from-sunflowers/

[4] *CirclePoints.* Wolfram Language Documentation. Wolfram Research. N.D. https://reference.wolfram.com/language/ref/CirclePoints.html

[5] Gonzalez, Alvaro. *Measurement of Areas on a Sphere Using Fibonacci and Latitude longitude Lattices.* Universidad De Zaragoza, 23 Decenber 2009. http://arxiv.org/pdf/0912.4540.pdf

[6] Perez-Garrido, A. *Global minimum for Thomson's problem of charges on a sphere.* National Center for Biotechnology Information. U.S. National Library of Medicine, 1 Apr. 2005. http://www.ncbi.nlm.nih.gov/pubmed/15903830/

[7] Schwartz, Richard Evan. *The 5-Electron case of Thomson's problem.* Experimental Mathematics 22.2 (2013): 157-186.

[8] *SpherePoints.* Wolfram Language Documentation. Wolfram Research. N.D. http://reference.wolfram.com/language/ref/SpherePoints.html

[9] Wang, Ning, and Jin-Luen Lee. *Geometric properties of the icosahedral-hexagonal grid on the two-sphere.* SIAM Journal on Scientific Computing 33.5 (2011): 2536-2559.

[10] Wolfram, Stephen. *The R&D Pipeline Continues: Version 11.1.* Wolfram Blog. 16 March 2017. http://blog.stephenwolfram.com/2017/03/the-rd-pipeline-continues-launching-version-11-1/

A Code For The Method

A.1 Mathematica Code

```
SphericalCoordinate[x_, y_] :=
 {Cos[x] Cos[y], Sin[x] Cos[y], Sin[y]};

NX[n_,x_] :=
Table[
 SphericalCoordinate[
  s * x,
  Pi / 2 * Sign[s] (1 - Sqrt[1 - Abs[s]])
  ],
 {s, -1 + 1 /(n - 1), 1 - 1 / (n - 1),
 (2 - 2 / (n - 1)) / (n - 1)}
 ];

GeneratePoints[n_] := NX[n, 0.1 + 1.2 n];
```

A.2 Python Code

```
import math
def spherical_coordinate(x, y):
    return [math.cos(x) * math.cos(y),
    math.sin(x) * math.cos(y), math.sin(y)]

def NX(n, x):
    pts=[]
    start = (-1. + 1. / (n - 1.))
    increment = (2. - 2. / (n - 1.)) / (n - 1.)
    for j in xrange(0, n):
        s = start + j * increment
        pts.append(
        spherical_coordinate(
        s * x, math.pi / 2. *
        math.copysign(1, s) *
        (1. - math.sqrt(1. - abs(s)))
        ))
    return pts

def generate_points(n):
    return NX(n, 0.1 + 1.2 * n)
```