Assignment for

Computer Science Theory for the Information Age

Day 3

BY ZEN HUANG

5120309027
2012 ACM class

June 14, 2013

Exercise 1. What is the surface area of a unit cube in d-dimensions?

Answer.

a d-dimensional cube has $2\,d$ faces($(d-1)$-dimensional cube), with each face of volume 1.

so

$$V_d = 2\,d \tag{1}$$

Exercise 2. Is the surface area of a unit cube concentrated to the equator, defined here as the hyperplane $\left\{\boldsymbol{x}\colon \sum_{i=1}^{d} x_i = \frac{d}{2}\right\}$, as is the case with the sphere?

Answer.

The answer is **true**.

Suppose we generate points randomly onto the surface of a d-dimensional cube. To prove the surface area is concentrated to the equator, all we need is to show that distance from random generated point to the equator is almost near zero.

Let p be the random generate point with coordinate $(x_1, x_2, \ldots, x_d)$.

Let X be the random variable of

$$X = \sum_{i=1}^{d} x_i. \tag{2}$$

Let l be the distance from point p to the equator

$$l = \frac{\left|\sum_{i=1}^{d}\left(x_i - \frac{1}{2}\right)\right|}{\sqrt{d}} = \left|\frac{X - \frac{d}{2}}{\sqrt{d}}\right| \tag{3}$$

Now we want to caculator value of $E(X)$ and $\sigma^2(X)$.

Notice that the d coordinates of p is not independent. Every time we generate a point onto the surface of d-cube, first we pick one dimesion k with probability $\frac{1}{d}$, draw $x_k = 0$ or $x_k = 1$ with equal probability, and then let the rest $d-1$ coordinates take values uniformly from [0,1].

So we have

$$\begin{aligned} E(X) &= \frac{1}{d}\sum_{k=1}^{d}\left(\frac{1}{2}\left(0+E\left(\sum_{i=1,i\neq k}^{d} x_i\right)\right)+\frac{1}{2}\left(1+E\left(\sum_{i=1,i\neq k}^{d} x_i\right)\right)\right) \\ &= \frac{1}{2}\sum_{k=1}^{d}\left(\sum_{i=1,i\neq k}^{d} E(x_i)+\sum_{i=1,i\neq k}^{d} E(x_i)+1\right) \\ &= \frac{d}{2} \end{aligned} \tag{4}$$

and

$$\begin{aligned} \sigma^2(X) &= E(X^2)-E^2(X) \\ &= \frac{1}{d}\sum_{k=1}^{d}\left(\frac{1}{2}E\left(\left(\sum_{i=1,i\neq k}^{d} x_i\right)^2\right)+\frac{1}{2}E\left(\left(1+\sum_{i=1,i\neq k}^{d} x_i\right)^2\right)\right)-\frac{d^2}{4} \\ &= (d-1)E(x_i^2)+2\,C_{d-1}^2E^2(x_i)+(d-1)E(x_i)+1-\frac{d^2}{4} \\ &= \frac{d-1}{12}+\frac{(d-1)(d-2)}{4}+\frac{d-1}{2}+1-\frac{d^2}{4} \\ &= \mathrm{O}(d) \end{aligned} \tag{5}$$

Now that we consider the Chebyshev's Inequality

$$P(|X-E(X)|\geqslant \epsilon)\leqslant \frac{\sigma^2(X)}{\epsilon^2} \tag{6}$$

and take $\epsilon = \frac{1}{d}$, we have

$$P\left(|X-E(X)|\geqslant \frac{1}{d}\right)\leqslant \frac{1}{\mathrm{O}(d)} \tag{7}$$

also for

$$P\left(l\geqslant \frac{1}{d\sqrt{d}}\right)\leqslant \frac{1}{\mathrm{O}(d)} \tag{8}$$

and that shows **As the dimension *d* gets higher, the randomly generated points are more likely to appear near the equator within a distance going to infinitesimal**, which leads to the answer proved.

Exercise 3. Generate 20 points uniformly at random on a 1,000-dimensional sphere of radius 100. Caculate the distance between each pair of points. Then project the data onto subspheres of dimension $k = 100, 50, 20, 5, 4, 3, 2, 1$ and caculate the sum of squared error between $\frac{\sqrt{k}}{\sqrt{d}}$ times the original distances and the new pair wise distances for each of the above values of k.

Answer.

In order to perform the generation, I wrote a cpp based programme, which will be shown below.

Algorithm 1

```
#include <cstdlib>
#include <cstdio>
#include <cmath>
#include "ctime"
#include "fstream"
#include "algorithm"
#include <iostream>

using namespace std;

ofstream file("output.txt");

const double radius = 100;
const int d = 1000;//origin dimension
const int testnum = 8;
const int k[ testnum ] = {100, 50, 20, 5, 4, 3, 2, 1};//set of dimension k
const int N = 20;//number of points generated

const double pi = 3.1415926535897932384626433;

double** coordinate;

double gaussian(double sigma){
    double flag = rand() % 2;
    flag -= 0.5;
    flag *= 2;
    return flag * sqrt( - 2 * sigma*sigma * log(( rand()+1.0 ) /
RAND_MAX));
}

void generate(int N, int d){
    coordinate = new double*[ N ];
    for (int i = 0; i < N; ++i)
    {
        double length = 0;
        coordinate[ i ] = new double[ d ];
        for (int j = 0; j < d; ++j)
        {
            coordinate[ i ][ j ] = gaussian( 1 );
            length += pow( coordinate[ i ][ j ], 2 );
        }
        length = sqrt( length );
        if(length != 0)
            for (int j = 0; j < d; ++j)
            {
                coordinate[ i ][ j ] *= radius / length;
            }
    }
}
```

```
double get_distance( double* x, double* y, int d ){
    double dis = 0;
    for (int p = 0; p < d; ++p)
    {
        dis += pow( x[ p ] - y[ p ], 2 );
    }
    return sqrt( dis );
}

void anylize_distance( double** p ,int d, int n, int &num, double &exp,
double &var ){
    file << "This is a "<< d <<" demensional space"<<endl;
    num = 0;
    exp = 0;
    var = 0;
    double* dis = new double[ n*n /2 ];
    for (int i = 0; i < n; ++i)
    {
        for (int j = i+1; j < n; ++j)
        {
            dis[ num ] = get_distance(p[i], p[j], d);
            exp += dis[num];
            ++num;
        }
    }
    exp /= num;
    sort(dis, dis+num);
    for(int i = 0; i < num; ++i){
        file << dis[i] << endl;
        var += pow( dis[i] - exp, 2 );
    }
    file << endl;
    var /= num;
    file << "expection=" << exp << endl;
    file << "var=" << var << endl;
    file << endl;
    delete[] dis;
}

void anylize_project( double** p, int k, int d, int n, double& var ){
    int num = 0;
    var = 0;
    for (int i = 0; i < n; ++i)
    {
        for (int j = i+1; j < n; ++j)
        {
            var += pow( get_distance(p[ i ], p[ j ], k) - get_distance(p[ i
], p[ j ], d) * sqrt( k ) / sqrt( d ), 2 );
            ++num;
        }
    }
    file << "The squared error between the original distances and the new
pair wise distances in "<< k <<" dimension projecting is "<< sqrt(var) <<
endl;
}
```

```
int main(int argc, char *argv[])
{
    srand(time(NULL));
    generate(N,d);
    int num;
    double exp,var;
    anylize_distance( coordinate, d, N, num, exp, var );

    for( int i = 0; i < testnum; ++i){
        anylize_distance( coordinate, k[ i ], N, num, exp, var );
    }

    for( int i = 0; i < testnum; ++i){
        anylize_project(coordinate, k[i], d, N, var);
    }
    return 0;
}
```

The output will be appedix to this document[1]. And I have the data anylized:

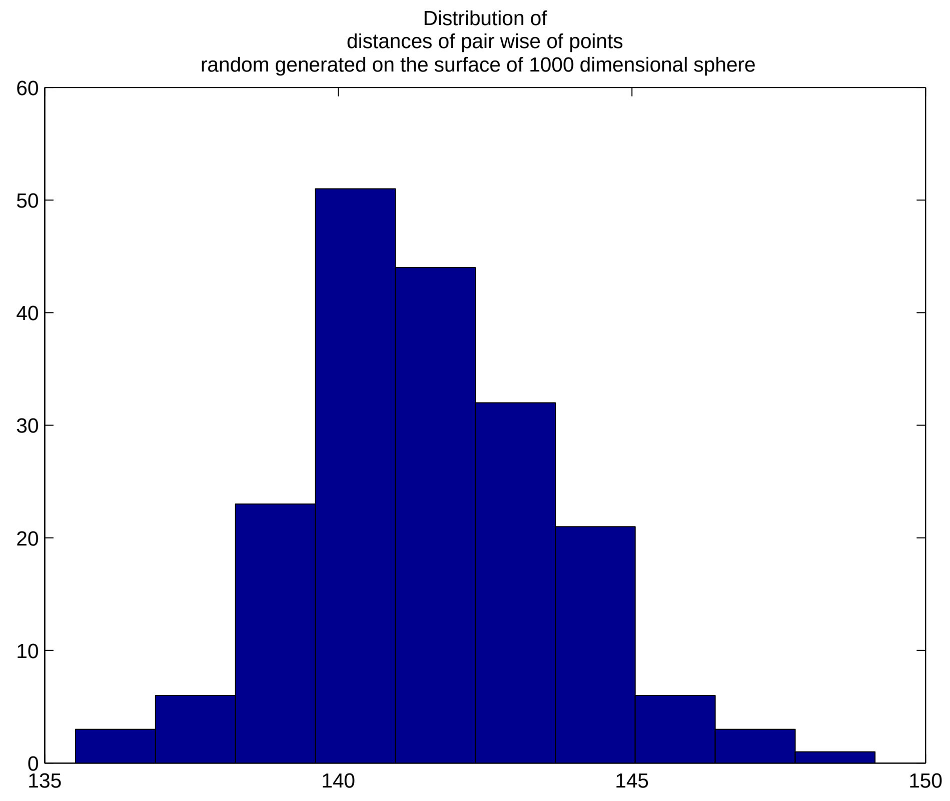

Figure 1.

The expectation for pair wise distances is 141.481, with $\sigma^2 = 4.66494$

1. see into output.txt.

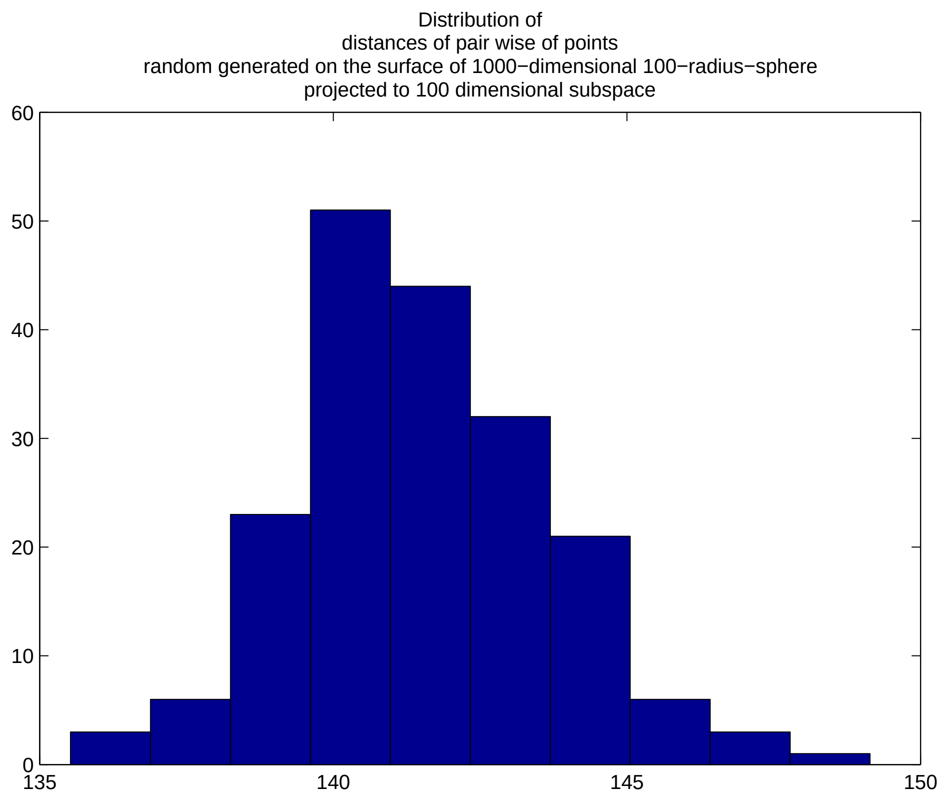

Figure 2.

The expectation for pair wise distances is 44.5058, with $\sigma^2 = 4.7675$

The squared error to the original distances is 28.0824

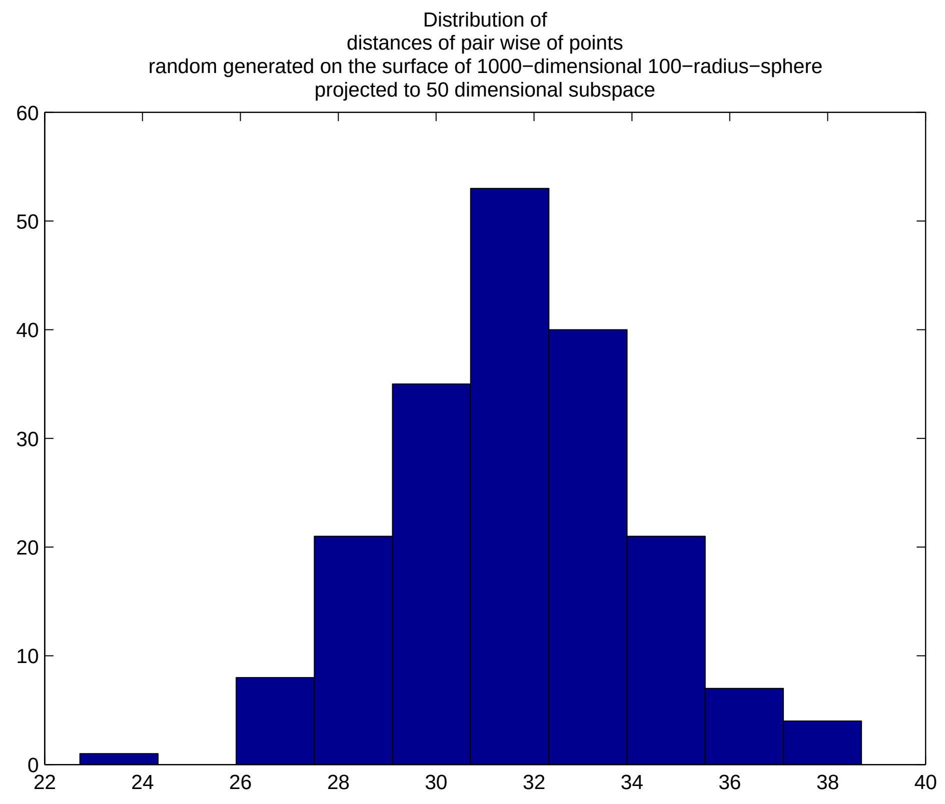

Figure 3.

The expectation for pair wise distances is 31.6399, with $\sigma^2 = 6.33677$

The squared error to the original distances is 33.7084

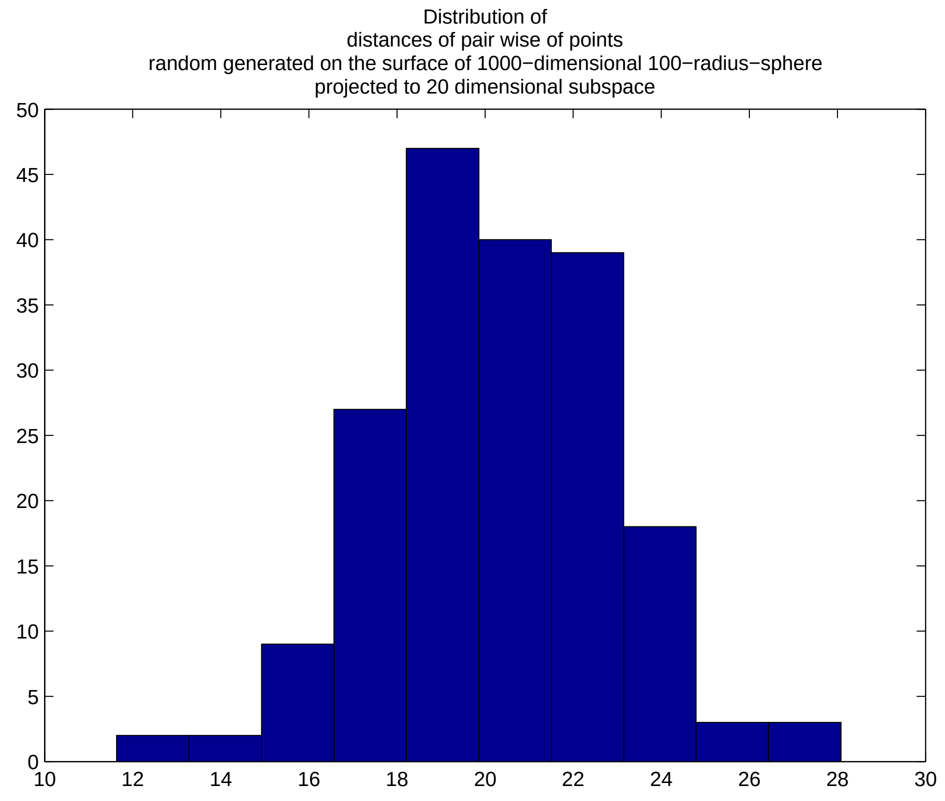

Figure 4.

The expectation for pair wise distances is 20.2423, with $\sigma^2 = 7.01114$

The squared error to the original distances is 36.5324

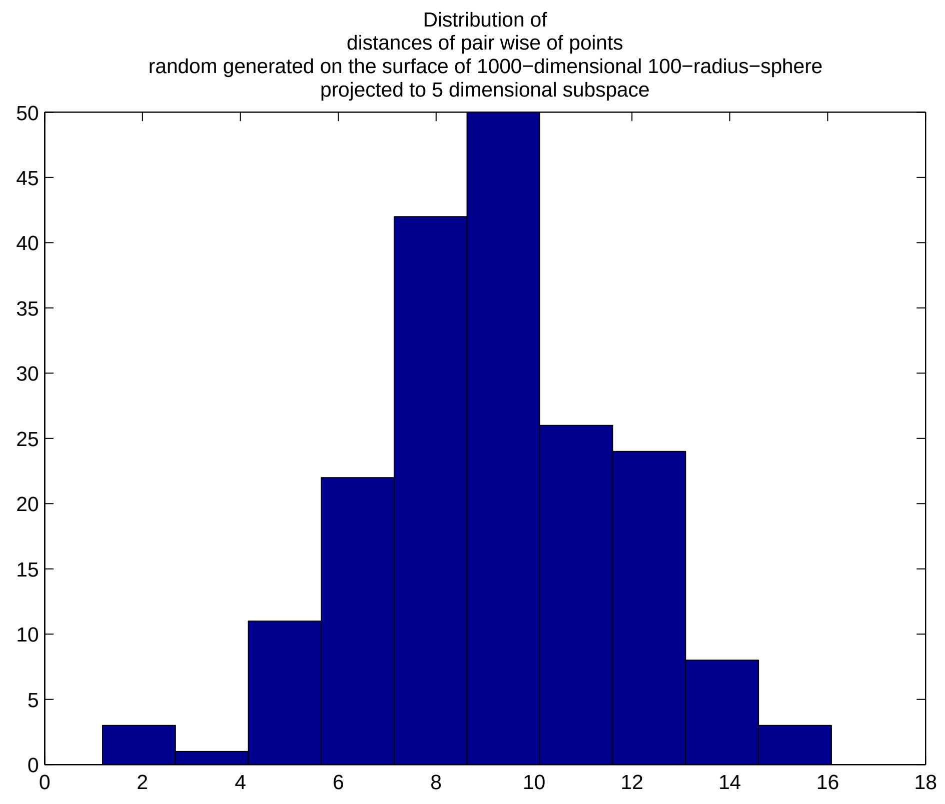

Figure 5.

The expectation for pair wise distances is 9.16856, with $\sigma^2 = 6.58486$

The squared error to the original distances is 37.1294

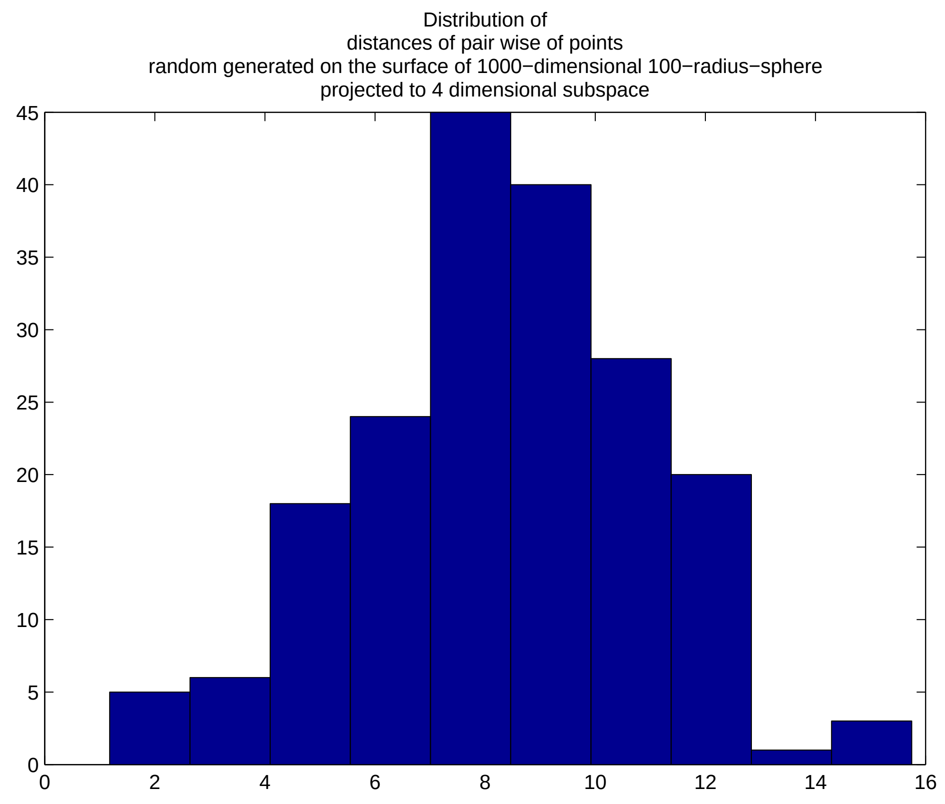

Figure 6.

The expectation for pair wise distances is 8.35724, with $\sigma^2 = 6.81835$

The squared error to the original distances is 36.7533

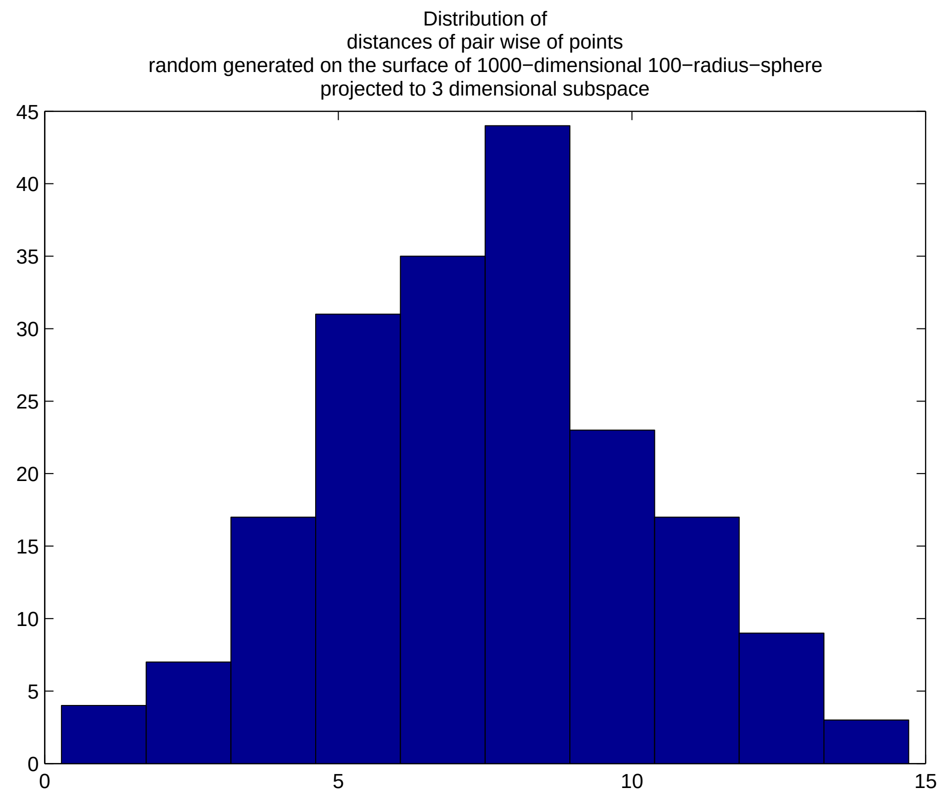

Figure 7.

The expectation for pair wise distances is 7.43634, with $\sigma^2 = 7.46059$

The squared error to the original distances is 37.7744

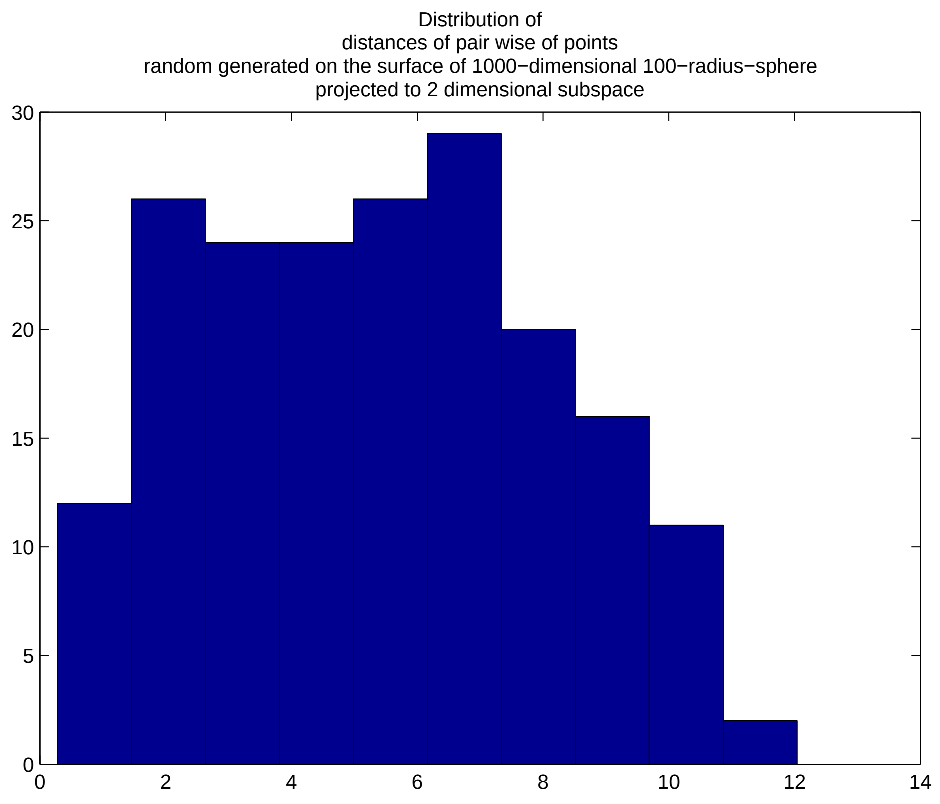

Figure 8.

The expectation for pair wise distances is 5.43386, with $\sigma^2 = 7.45803$

The squared error to the original distances is 39.6146

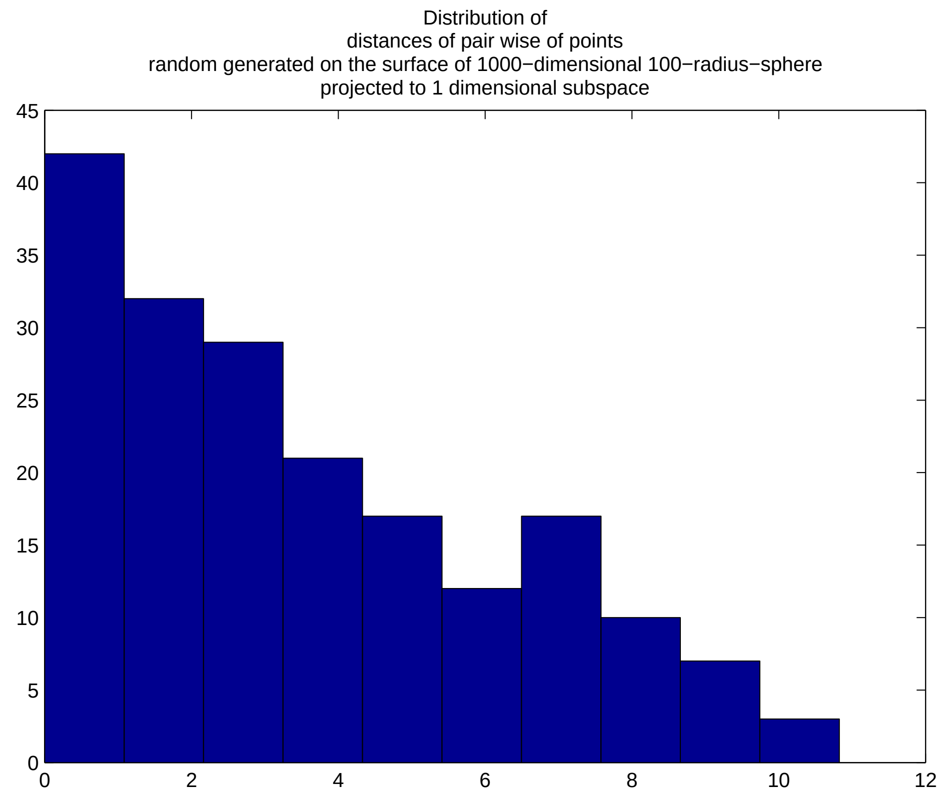

Figure 9.

The expectation for pair wise distances is 3.58003, with $\sigma^2 = 7.41404$

The squared error to the original distances is 39.5334