Sahil Bansal 77642095

Introduction
In this project, I will be comparing Monte Carlo (Probabilistic) and Cube Based Integration (Deterministic) methods to compute the volume of the hypersphere. I will be estimating the volume of a d-dimensional hypersphere of radius r=1 centered at the origin.

Part 1:

Monte Carlo Integration:
In this method, I am surrounding the hypersphere (with radius r=1 and centered on the origin) with the smallest hypercube that can enclose this hypersphere. So, this hypercube is centered at the origin with sides of length 2 and has a volume of 2^d. Now, I have selected N points at random inside the hypercube and counted how many of them lie inside the hypersphere. Then, the volume of the hypersphere is estimated as

$$(Points_{inside}/Points_{total}) * Volume_{hypercube}$$

To assure 4 digits of accuracy, I have used different values of N for each dimension but for a given dimension d, I have used a fixed number of N to calculate mean and standard deviation.

I have used 95% confidence interval. Therefore, to assure 4 digits of accuracy, assuming that n samples are drawn from Gaussian distribution, we need

$$z_{0.95} * error_{standard} <= 0.001$$

$z_{0.95} * \sigma/\sqrt{n} <= 0.001$, where σ is the standard deviation of n samples

Below are my results which are shown as an interval with an upper and lower bound and with 95% confidence, we can say that the true mean of the samples is between $\bar{x} - z_{0.95}\frac{\sigma}{\sqrt{n}}$ and $\bar{x} + z_{0.95}\frac{\sigma}{\sqrt{n}}$. In my results, n is 100 and $z_{0.95}$ is 1.96 and I have chosen N points to ensure that that the estimated value lies between $\bar{x} - 0.001$ and $\bar{x} + 0.001$ with 95% confidence.

dimension, d	Mean, $\bar{x}$	Standard Deviation, σ	Number of Points used, N	Estimated Value with 95% Confidence Interval
1	2	0	1000	2 ± 0
2	3.142	0.005	100000	3.142 ± 0.001
3	4.189	0.006	500000	4.189 ± 0.001
4	4.934	0.003	5000000	4.934 ± 0.001
5	5.264	0.004	8000000	5.264 ± 0.001

dimension, d	Mean, $\bar{x}$	Standard Deviation, σ	Number of Points used, N	Estimated Value with 95% Confidence Interval
6	5.168	0.004	20000000	5.168 ± 0.001
7	4.724	0.004	50000000	4.725 ± 0.001
8	4.059	0.004	80000000	4.059 ± 0.001
9	3.298	0.004	100000000	3.298 ± 0.001
10	2.550	0.005	80000000	2.550 ± 0.001
11	1.882	0.006	100000000	1.882 ± 0.001
12	1.335	0.007	100000000	1.335 ± 0.001
13	0.911	0.006	200000000	0.911 ± 0.001
14	0.600	0.004	500000000	0.600 ± 0.001
15	0.382	0.007	200000000	0.382 ± 0.001
16	0.235	0.005	500000000	0.235 ± 0.001
17	0.142	0.007	500000000	0.142 ± 0.001
18	0.083	0.007	500000000	0.083 ± 0.001
19	0.046	0.007	500000000	0.046 ± 0.001
20	0.026	0.006	500000000	0.026 ± 0.001
21	0.014	0.008	500000000	0.014 ± 0.002
22	0.008	0.008	500000000	0.008 ± 0.002

In higher dimensions, the probability that a point in a cube falls inside a ball is very minute. So, estimating hypersphere volumes by probabilistic methods start to fail in higher dimensions. For 40 dimensions, the probability that a point in a cube falls inside a ball is less than 10^{-20}, which is to say that it will never happen even on a teraflop computer.

In the above table, For d={1,2,..,9}, the N random numbers were selected from a uniformly distributed random numbers in the interval (-1,1) and then were checked for whether they lie inside the hypersphere or not and accordingly the volume was estimated to 4 digit precision. Till d=9, no variance reduction technique was used.

For d ≥10, the regular Monte Carlo method could be not used to produce results in an effective manner. So, to improve Monte Carlo simulation, I used various *Variance Reduction* techniques.

Firstly, I tried the *Antithetic Sampling,* so for d=10, whenever I would choose a random point X, I would also check if (-X) lies inside the hypersphere or not. This helped in reducing the variance

and I was able to get results with the given accuracy with 80000000 points. But this technique, did not helped in achieving given accuracy for d>10 even after increasing the sample points.

Secondly, I tried *Stratified Sampling*, which means that the variance of f over a subinterval should be lower than the variance over the whole interval. This procedure prevents draws from clustering in a particular region of the interval as the procedure is force to visit each subinterval. Here, the idea is to choose $\lambda \in (0,1)$ and draw $N_a = \lambda N$ data points over $[0,\lambda]$ and $N_b = (1-\lambda)N$ over $[\lambda,1]$. I tried with λ values of 0.25 and 0.5 over the $[-1,1]$ interval. This technique also helped in reducing the number of points N, needed if tried individually for d=10, but in combination with Antithetic sampling didn't improve the efficiency. So, the number of sample points needed by this technique (with and without combination with Antithetic sampling) was same (80000000) and it also didn't reduce the variance for dimensions higher than 10.

Next, I tried the *Quasi-Monte Carlo* method. The quasi-Monte Carlo method is a method for numerical integration where points are chosen from low discrepancy sequences (quasi-random sequences) like the Halton sequence, the Sobol sequence, or the Faure sequence instead of pseudorandom numbers as in regular Monte Carlo method. The advantage of using the low discrepancy sequences is a faster rate of convergence which is close to $O(1/N)$ as opposed to $O(1/\sqrt{N})$ in Monte Carlo method. When I tried the Quasi-Monte Carlo method, I was able to get the estimated values much faster but since this method is deterministic as we are choosing well distributed points from the low discrepancy sequences, we only have the bound $\epsilon \leq V(f)D_N$ for error, and the error is hard to estimate. In order to analyze and estimate the variance, I used the next explained method.

Lastly, I tried the *Randomized quasi-Monte Carlo* for variance reduction. I chose N points $\{x_1, x_2, \ldots, x_N\}$ from the low discrepancy sequence, Sobol sequence (For higher dimensions, Sobol sequence perform much better than Halton and Faure sequence). Then I sampled random vector U from the uniformly distributed random number sequence and then for each x_j, I created

$$y_j = x_j + U(mod 1)$$

And used the sequence (y_j) instead of (x_j). Randomization allowed me to give an estimate of the variance while still using quasi-random sequences. Compared to standard Monte-Carlo, the variance and the computation speed are slightly better and I was able to estimate the results for d = {11,..,22} using this approach.

In summary, for this method, I have able to push d till 22.

Cube Based Integration:

In this method, I have divided each of the d dimensions into K segments, so that the hypercube is dissected into K^d small hypercubes. I checked each small hypercube of whether it is completely inside the hypersphere, wholly outside the hypersphere or else is intersected by the hypersphere. I estimated the volume of the hypersphere as

$$(Hypercubes_{inside} + \frac{Hypercubes_{intersected}}{2}) * (Volume_{Small-Hypercube})$$

To check whether a cube lies inside, outside or is intersected by the hypersphere, I checked the distance of the smaller hypercube's center from the hypersphere center. If the distance is less

than or equal to radius of hypersphere minus half the diagonal of the cube, then it is inside, if distance is greater than or equal to radius of hypersphere plus half the diagonal of the cube then it is outside else it is intersected. For each value d, I determined how large K should be by using an iterative method to calculate the volume and increasing the value of K till we achieve a 4-digit precision which was used as a stopping criteria.

For this method, I have been able to push d till 5. Here are the results for the same:

dimension	Actual Volume	Estimated Volume	K value
1	2	2	100
2	3.142	3.142	2400
3	4.189	4.189	600
4	4.935	4.935	360
5	5.264	5.264	370

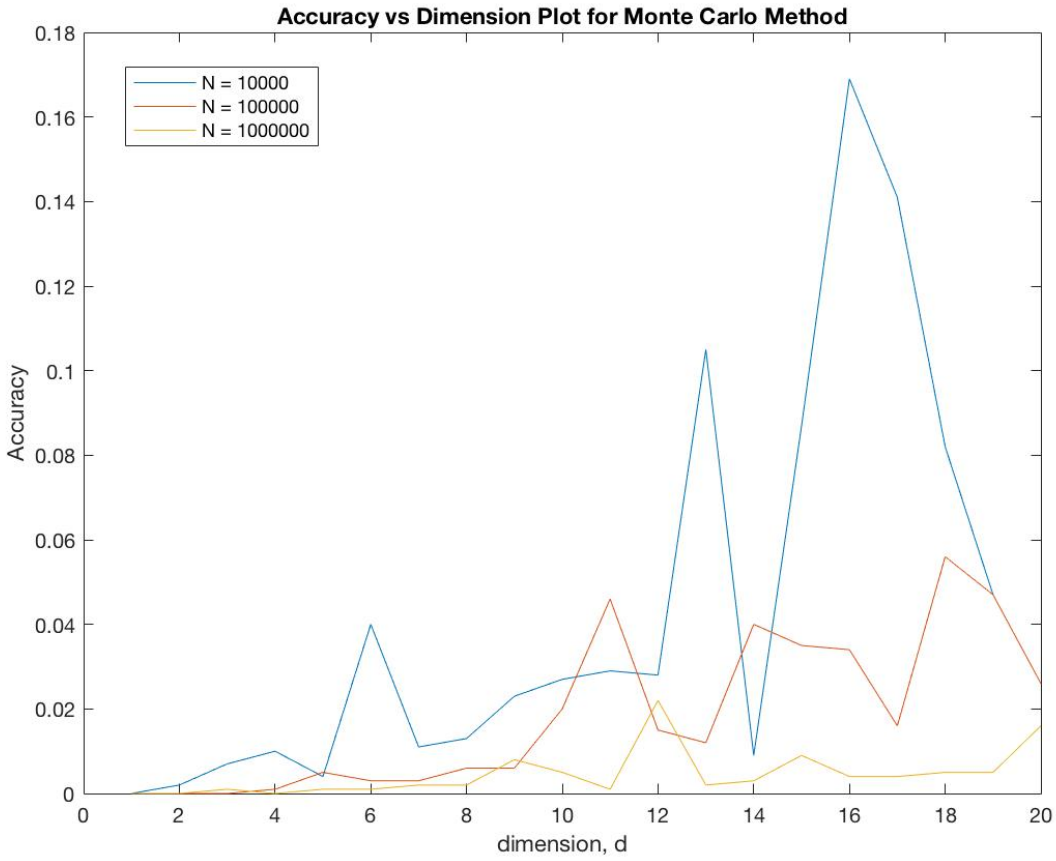
Comparing Efficiency: Run Times and Accuracies

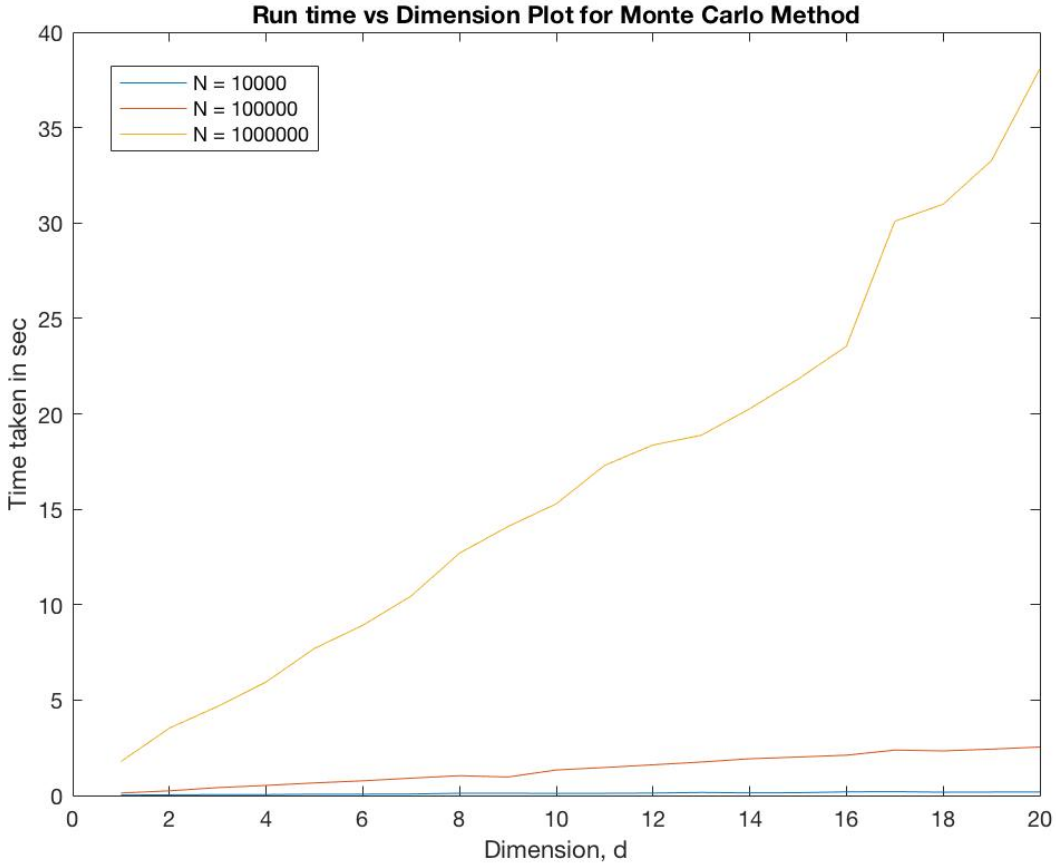
As we have seen above, we are able to estimate the volumes of the hypersphere for d=22 for Monte Carlo and for d=5 for Cube based Integration. In same amount of time, we are able to estimate the volumes more accurately in Monte Carlo than the Cube based method. Therefore, Monte Carlo method is more efficient as d becomes large given that we insist on 4 digits of precision in the answer.

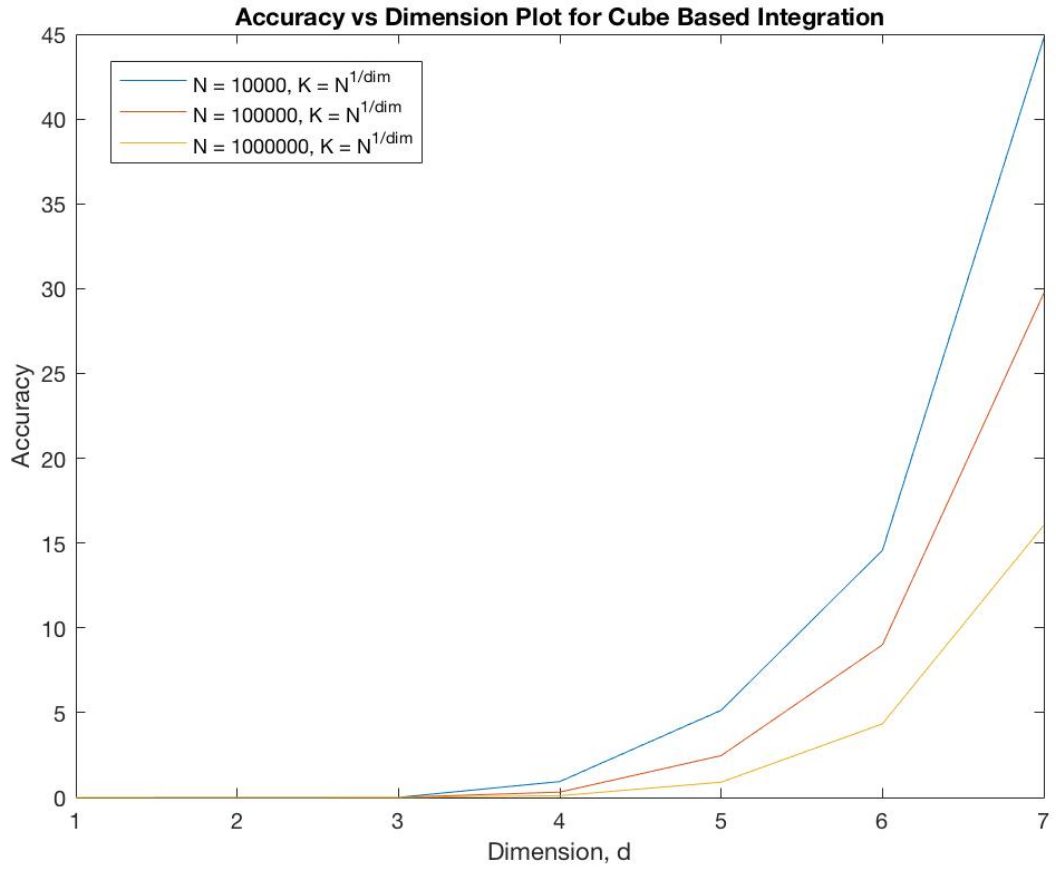
Plots:

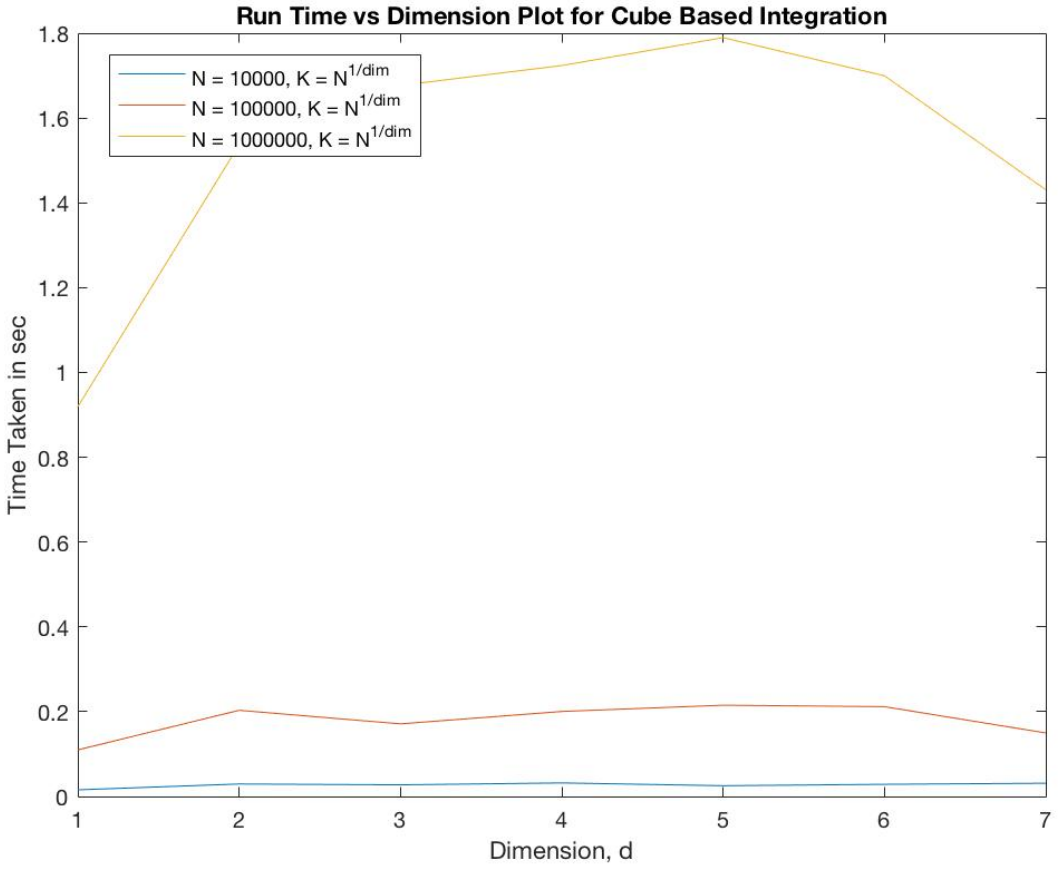
I have plotted the Accuracy vs Dimensions and Run Time vs Dimensions plot for N={10000, 100000, 1000000} for both the Monte Carlo and Cube Based Integration. In both the cases, the accuracy is calculated by taking the difference of estimated values from the actual values. For Cube based, I have taken $K = N^{1/dim}$ where K is the number of segments that a particular dimension is divided into.

When we compare the accuracies between Monte Carlo and Deterministic, we see that the accuracy of Monte Carlo is much better than that of Cube base since for cube based, when d>3, the accuracies gets really bad. But on the other hand, if we look at the run time given that $K = N^{1/dim}$, we see that Cube Based Integration has lower run times than that of Monte Carlo. This is due to the fact that time taken in Monte Carlo is due to 100*N where 100 is the number of runs, while for Cube Based, it is only N as we don't have multiple runs for cube based. So, If we had chosen $K = (R * N)^{1/dim}$, then we could have got similar run times for both the methods but the accuracy of Monte Carlo would have still been better than that of Cube based. The case when we have to provide an answer till 4 digit of precision, Monte Carlo's run time is much less than that of Cube based as the value needed for K is high to get that precision and the run time of Cube based is $O(K^{dim})$ and as dimension increases, the run time increases exponentially for it.


Accuracy vs Dimension Plot for Monte Carlo Method
N = 10000
N = 100000
N = 1000000
Accuracy
dimension, d
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
2
4
6
8
10
12
14
16
18
20


Run time vs Dimension Plot for Monte Carlo Method
N = 10000
N = 100000
N = 1000000
Time taken in sec
Dimension, d
40
35
30
25
20
15
10
5
0
0
2
4
6
8
10
12
14
16
18
20


Accuracy vs Dimension Plot for Cube Based Integration
N = 10000, K = N^1/dim
N = 100000, K = N^1/dim
N = 1000000, K = N^1/dim
Accuracy
Dimension, d
0
5
10
15
20
25
30
35
40
45
1
2
3
4
5
6
7


Run Time vs Dimension Plot for Cube Based Integration
N = 10000, K = N^1/dim
N = 100000, K = N^1/dim
N = 1000000, K = N^1/dim
Time Taken in sec
Dimension, d
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
1
2
3
4
5
6
7

Deterministic methods are better than Monte Carlo where the deterministic method is practical. But in out case, to estimate the volume of the hypersphere with the given accuracy of 4 digit and for higher dimensions, deterministic method is impractical. We use Monte Carlo because of the curse of dimensionality. The curse is that the work to solve a problem in many dimensions by the cube based method grows exponentially here with the dimensions making it impractical to generate results in a day's time of a computer.

If we demand 8 digits of precision, then in Monte Carlo method we will require much more sample points and much more runs to achieve this precision. For Cube based, we need to split the cube into much smaller parts to achieve this precision and would require much larger value of K. For higher dimension, to get 8 digits precision, Monte Carlo method will be more efficient than cube based method.

Part 2:

In this part, I am using a fixed value of N (10^6) for all dimensions and estimating the volumes.
For this part, I am not using variance reduction techniques in Monte Carlo as I used in Part 1 to compare a regular Monte Carlo with the Cube Based Integration.
For cube based method, I have assumed that the volume of the hypersphere is the sum of those h's that are inside the hypersphere plus half of those that are intersected by its boundary. One clever way than taking half of the intersected h's is to find a fraction of the part inside using the Monte Carlo method or by using a deterministic method where I could choose deterministic points inside the smaller hypercube and then check how many lie inside the intersection. This fraction could indicate the fraction of intersection inside the hypersphere and can be used to provide more accurate results for Cube based integration. But the downside of this approach will be the increase in the number of computations needed (Since for each intersected cube we will be running another method to find the better approximation on the fraction part of the cube which is inside).

Based on the part 1 and also as evident from the following results, Monte Carlo method is giving more accurate results.

Dimension , d	Actual Volume of Hypersphere	Monte Carlo Estimated Volume of Hypersphere with 95% Confidence Interval	Cube Based Estimated Volume of Hypersphere
1	2.000	2.000 ± 0	2.000
2	3.142	3.141 ± 0	3.141
3	4.189	4.190 ± 0.001	4.191
4	4.935	4.935 ± 0.002	5.041
5	5.264	5.265 ± 0.002	6.166
6	5.168	5.168 ± 0.004	9.511
7	4.725	4.722 ± 0.005	20.766

Dimension , d	Actual Volume of Hypersphere	Monte Carlo Estimated Volume of Hypersphere with 95% Confidence Interval	Cube Based Estimated Volume of Hypersphere
8	4.059	4.058 ± 0.007	35.76
9	3.299	3.301 ± 0.008	79.86
10	2.550	2.553 ± 0.010	319
11	1.884	1.873 ± 0.012	512
12	1.335	1.334 ± 0.012	1937.5
13	0.911	0.894 ± 0.017	3527.8
14	0.599	0.601 ± 0.018	7329.1
15	0.382	0.375 ± 0.023	12956
16	0.235	0.237 ± 0.025	32768
17	0.141	0.147 ± 0.029	65536
18	0.082	0.089 ± 0.028	1.3107E+05
19	0.047	0.052 ± 0.031	2.6214E+05
20	0.026	0.042 ± 0.040	5.2429E+05

In the above table, for the cube based integration we start to see huge values of volume of hypersphere for dimensions greater than 6, since when for a fixed N = 10^6, we calculate K as the nearest integer of ($N^{1/d}$), we start to get very small K, and when K is really small, lets say 2 for dimensions greater than or equal to 16, then there are just few small hypercubes and based on if they intersect and using the approximation that we will take half the volume of the intersected cubes, we get the results as below where the volume of hypersphere is near the volume of the hypercube. So, for a fixed N of 10^6, cube based method fails terribly for dimensions greater than 5.

Dimension, D	N	K	Volume of Whole Hypercube (2^d)	Estimated Volume of Hypersphere
6	1000000	10	64	9.511
7	1000000	7	128	20.766
8	1000000	6	256	35.76
9	1000000	5	512	79.86
10	1000000	4	1024	319
11	1000000	4	2048	512

Dimension, D	N	K	Volume of Whole Hypercube (2^d)	Estimated Volume of Hypersphere
12	1000000	3	4096	1937.5
13	1000000	3	8192	3527.8
14	1000000	3	16384	7329.1
15	1000000	3	32768	12956
16	1000000	2	65536	32768
17	1000000	2	1.3107E+05	65536
18	1000000	2	2.6214E+05	1.3107E+05
19	1000000	2	5.2429E+05	2.6214E+05
20	1000000	2	1.0486E+06	5.2429E+05

Part 3:
I will be using Monte Carlo method with d=2 to compute π as accurately as possible if I am given only a specific number of samples N and I don't know the value of N in advance. I came to this conclusion by trying out Monte Carlo and Cube based method for different dimensions. The pattern, I observed is that for the same value of N, Monte Carlo is giving more accurate results than Cube based. In each method as the dimension increases, the estimates get less accurate. In Monte Carlo, dimension d=2 is giving the most consistently accurate estimates for the value of π. Therefore, to estimate the value of π, I will be using Monte Carlo method with dimension 2.

N=1000

dimension, d	Actual Value of π	Estimated Value of π by Monte Carlo	Estimated Value of π by Cube Based
2	3.142	3.143 ± 0.003	3.102
3	3.142	3.143 ± 0.006	3.216
4	3.142	3.136 ± 0.005	3.745
5	3.142	3.138 ± 0.007	5.391
6	3.142	3.137 ± 0.007	5.772

Sahil Bansal 77642095

N=10000

dimension, d	Actual Value of π	Estimated Value of π by Monte Carlo	Estimated Value of π by Cube Based
2	3.142	3.141 ± 0.001	3.134
3	3.142	3.140 ± 0.002	3.146
4	3.142	3.142 ± 0.002	3.428
5	3.142	3.140 ± 0.002	4.412
6	3.142	3.139 ± 0.002	4.911

N=100000

dimension, d	Actual Value of π	Estimated Value of π by Monte Carlo	Estimated Value of π by Cube Based
2	3.142	3.142 ± 0.0003	3.141
3	3.142	3.142 ± 0.001	3.147
4	3.142	3.141 ± 0.001	3.241
5	3.142	3.142 ± 0.001	3.801
6	3.142	3.141 ± 0.001	4.397

N=1000000 (Note, this is the last N value that I tries as it is given in the problem statement that we are not allowed to use more than 1 million sample points)

dimension, d	Actual Value of π	Estimated Value of π by Monte Carlo	Estimated Value of π by Cube Based
2	3.142	3.142 ± 0.0001	3.141
3	3.142	3.142 ± 0.0002	3.143
4	3.142	3.142 ± 0.0001	3.175
5	3.142	3.142 ± 0.0002	3.400
6	3.142	3.142 ± 0.0002	3.850

Tighter Bounding Shape than Hypercube

Cosmotopes or any regular polytope with more sides than the hypercube could provide a tighter bounding shape than a Hypercube. In 2-D we know that as the number of sides of the polygon increases, the shape starts to become more and more like a circle and bounds the circle more tightly. Similarly in 3-D, Dodecahedron would provide a more tighter bounding shape than a hypercube.