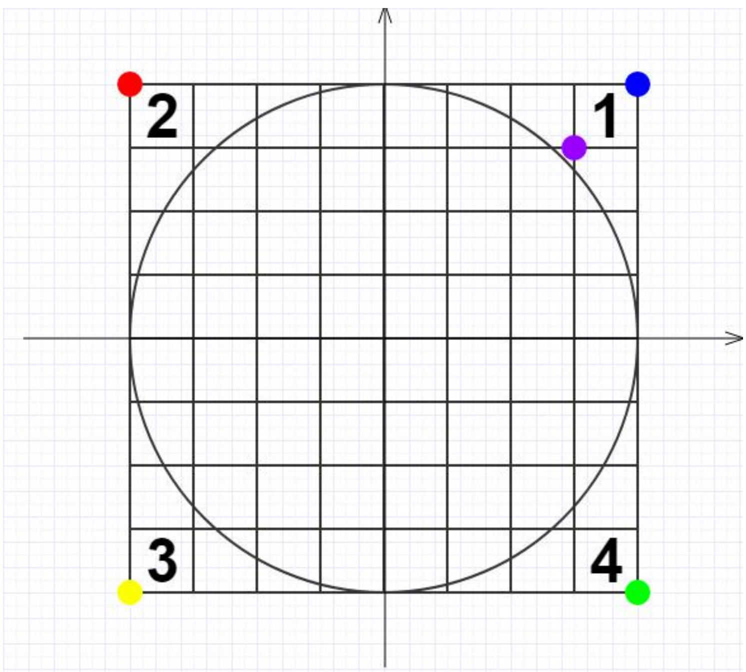
Project MC: Monte Carlo vs. Deterministic Volume Integration

Kuan Li 000995791

Introduction of Diagram

Monte Carlo Method: Based on different dimensions, my method is run one simulation many times until these simulations are satisfied 99% confidence and 4digits accuracy. One simulation means for example in 2 dimensions simulation, generate 1 million random points and check if the points are 1 distance from the origin and "return" the "calculate" which is 3.14xxx. So now we have many many simulations which means many 3.14xxx. Then based on these 3.14xxx, calculate their 99% confidence and 4digits accuracy. When satisfied the restriction, jump out of the loop. This is the easier and faster one than Cube Based and I run to 5 dimensions. But I think I can run to any dimensions, 5 dimension is just an example.

Cube Based Method: My method is creative I believe. My method is using the farthest and nearest vertex of each "small cubes" to judge the relationship between "small cubes" and "circle". For example, in 2 dimension, you could see the map blow. In first quadrant, the first small cube is named "1". Its farthest vertex from circle origin is the blue vertex. And blue point's distance from origin could tell us the relationships. When (the distance) > (r + small cube diagonal length), the small cube is out of the circle. When (the distance) < (r), the small cube is in the circle. But when the distance is between r and r+small cube diagonal length, we must need to calculate the nearest point which is the purple one to make sure the cube is cross the circle (because when the cube is not 45° from x coordinate, the restriction distance may less than r + small cube diagonal length, to make it more clear, I use the nearest point). The diagram in second, third, forth quadrant is similar to first quadrant.



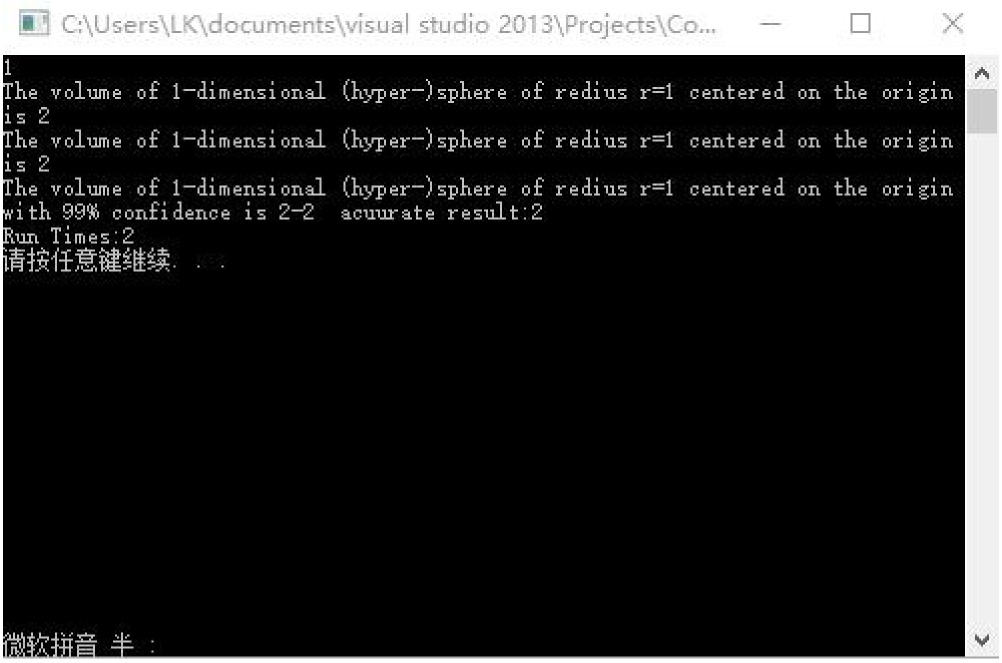
Above is my basic diagram and it is also applied to 3 dimension, 4 dimension or higher dimension.

I write to 5 dimension. But based on this method, I could write to any dimensions I believe.

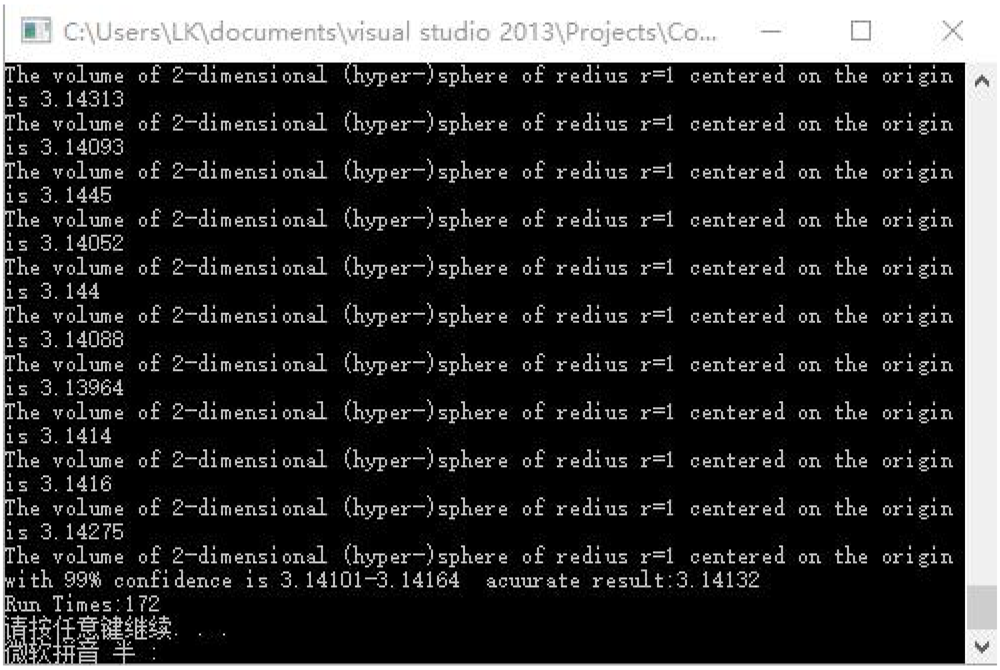
Run Test

Monte Carlo Method: I write this method based on C++. It is very simple and easy to understand. Most importantly, it runs very fast, so I didn't count the time. I directly show my result included confidence interval, accurate result which is the mean of samples and the run times to get 99% confidence and 4 digits accuracy:

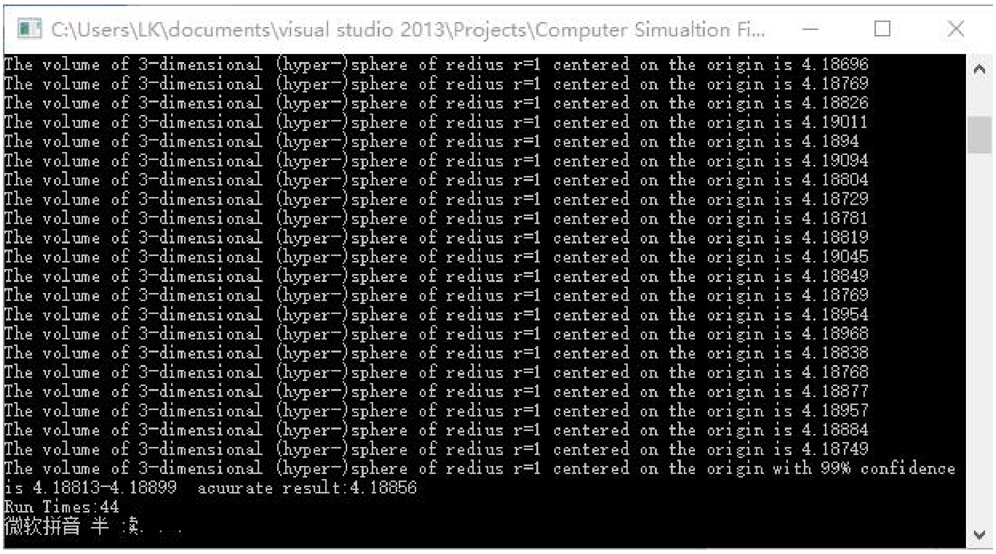
Dimension 1: Nd=100,000



Dimension 2: Nd=1,000,000

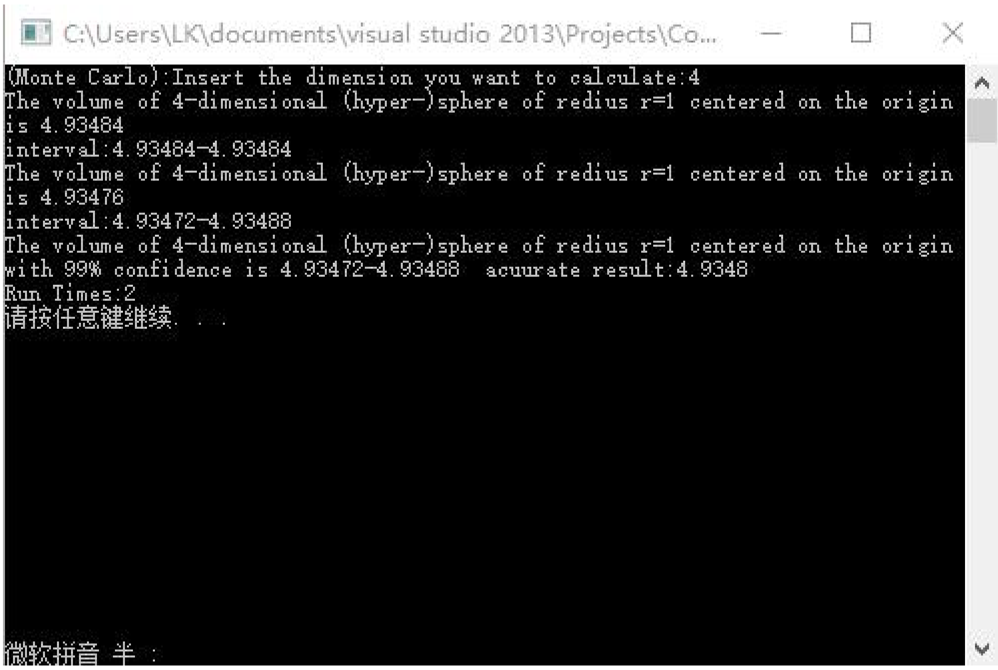


Dimension 3: Nd=10,000,000



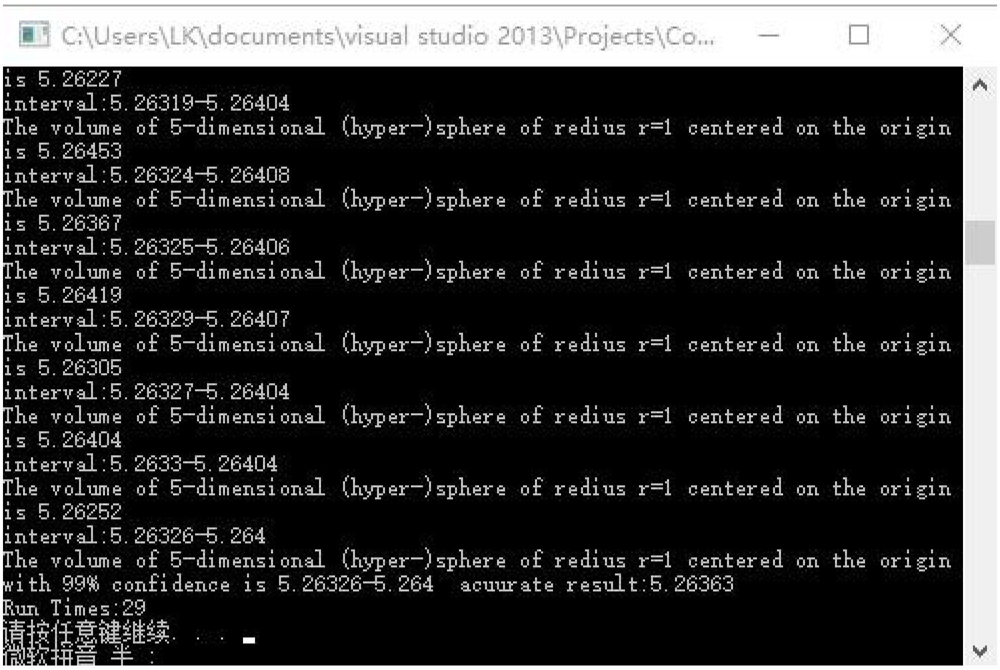
Dimension 4: Nd=400,000,000

I changed some code and could see intervals between each run. I find 400,000,000 is the magic number that could reach the accuracy in 2 times. When I choose 100,000,000, the number is always between 4.1933-4.1935 and run many many times to reach 4 digits accuracy.



Dimension 5: Nd=200,000,000

I run 29 times and I think it is fairly a good result. But the interval is 5.26326-5.264, the last interval is not exactly accurate maybe because the machine error. I don't why the machine think the interval have 4 digits accuracy and jump out of the loop. But I could guarantee my result is right because I see each intervals fluctuate smaller and smaller and reach that accuracy finally.



Cube Based Method:

I write a strict diagram that judge the in circle,out circle or cross circle in CubeBased3.0.cpp because Dr.Hayes required us to be strict and look it as a black-box test. But in CubeBasedTest.cpp, to quickly test my method, I use the symmetry of "circle". For example,in 2-dimention test, I just need to calculate 1/4 of the big square which separates by coordinate axis.In 3-dimention test, I just need to calculate 1/8. But it is just for test.

Although I write my code as simple as I can, the 4 digits accuracy for 3 dimension is still hard to close. The boundary of the accuracy is defined from (in circle cubes) to (in circle cubes + cross circle cubes). Only if the simple number is big enough can we calculate the accuracy to 4 digits. In 2-dimension, I use 20000*20000 to calculate accuracy to 4 digits. It's fast. But in 3-dimension, I think I need at least 40000*40000*40000 to close the accuracy. It is too huge number. More importantly, it is not rational if we don't count the cross circle cube into calculate. So I think we should use cross circle cube at least count it as half cube. So we will get an exact number and we can take that number into consideration to judge if we students did right or wrong. Or too many calculations like 40000*40000*40000 are just a waste of time I think.

I believe my method is one of the most efficient way to simulate cube because in most times,I just need to judge one farthest point from the origin in one cube. And compared other people's run time, my method is also the fastest one. Here is the run test. Note: I run the CubeBasedTest.cpp which is a simplified method. So the normal run time is about *2**d.

Dimension 1: **k=1000**

```
C:\Users\LK\documents\visual studio 2013\Projects\Co...
(Cube Based) Insert the dimension: 1
(Cube Based) The volume of 1-dimensional (hyper-)sphere of redius r=1 centered o
n the origin is between 2-2 (2)
Run time: 0
请按任意键继续. . .
```

Dimension 2:

K=10000

```
C:\Users\LK\documents\visual studio 2013\Projects\Co...
(Cube Based) Insert the dimension: 2
(Cube Based) The volume of 2-dimensional (hyper-)sphere of redius r=1 centered o
n the origin is between 3.14079-3.14239 (3.14159)
Run time: 7.457
请按任意键继续. . .
```

K=15000

```
C:\Users\LK\documents\visual studio 2013\Projects\Co...
(Cube Based) Insert the dimension: 2
(Cube Based) The volume of 2-dimensional (hyper-)sphere of redius r=1 centered o
n the origin is between 3.14106-3.14212 (3.14159)
Run time: 16.987
请按任意键继续. . .
```

k=20000 This is my computer Alienware 17(2014): 29.229s

```
C:\Users\LK\documents\visual studio 2013\Projects\Co...
(Cube Based) Insert the dimension: 2
(Cube Based) The volume of 2-dimensional (hyper-)sphere of redius r=1 centered o
n the origin is between 3.14119-3.14199 (3.14159)
Run time: 29.229
请按任意键继续. . .
```

This is macbook Pro(2017): 2.929S What a big difference!So, I changed my computer.

```
(Cube Based) Insert the dimension: 2
(Cube Based) The volume of 2-dimensional (hyper-)sphere of
redius r=1 centered on the origin is between 3.14119-3.14199
(3.14159)
Run time: 2.92922
sh: Pause: command not found
Program ended with exit code: 0

All Output ◇    Filter
```

Dimension 3: **k=1000 Run time:1s Accuracy:1**

```
(Cube Based) Insert the dimension: 3
(Cube Based) The volume of 3-dimensional (hyper-)sphere of
redius r=1 centered on the origin is between 4.16991-4.20761
(4.18876)
Run time: 1.29375
sh: Pause: command not found
Program ended with exit code: 0

All Output                    Filter
```

k=2000 Run time:9s Accuracy:2

```
(Cube Based) Insert the dimension: 3
(Cube Based) The volume of 3-dimensional (hyper-)sphere of
redius r=1 centered on the origin is between 4.17936-4.19821
(4.18879)
Run time: 9.76522
sh: Pause: command not found
Program ended with exit code: 0

All Output                    Filter
```

K=4000 Run time:75s Accuracy:2

```
(Cube Based) Insert the dimension: 3
(Cube Based) The volume of 3-dimensional (hyper-)sphere of
redius r=1 centered on the origin is between 4.18721-4.19664
(4.19193)
Run time: 75.5053
sh: Pause: command not found
Program ended with exit code: 0

All Output                    Filter
```

K=8000 Run time:609.778s Accuracy:2(There may be a little accuracy mistake about higher because I changed a little code and I forgot to fix)

```
(Cube Based) Insert the dimension: 3
(Cube Based) The volume of 3-dimensional (hyper-)sphere of
redius r=1 centered on the origin is between 4.188-4.19343
(4.19072)
Run time: 609.778
sh: Pause: command not found
Program ended with exit code: 0
```

All Output ◊ Filter

K=15000 Run time:3972s Accuracy:2   It's pretty nearly!

```
(Cube Based) Insert the dimension: 3
(Cube Based) The volume of 3-dimensional (hyper-)sphere of
redius r=1 centered on the origin is between 4.18837-4.19088
(4.18963)
Run time: 3972.2
sh: Pause: command not found
Program ended with exit code: 0
```

All Output ◊ Filter

K=20000 Run time:9655s Accuracy:2   Pretty pretty near...

```
(Cube Based) Insert the dimension: 3
(Cube Based) The volume of 3-dimensional (hyper-)sphere of
redius r=1 centered on the origin is between 4.18848-4.19036
(4.18942)
Run time: 9655.7
sh: Pause: command not found
Program ended with exit code: 0
```

As we can see, the interval gets smaller and smaller. I didn't have too many time to run to the accuracy. But I think I could reach to 4 digits accuracy if I have time and better computers. Another interesting thing I found is that the lower boundary is pretty close to the accurate number, so we could know that with the increasing number of cubes, the "volume" could be represented by the cubes "in circle". And there are many waste cubes cross circle..

Dimension 4:   k=400 Run time:10.8223s accuracy:1

```
(Cube Based) Insert the dimension: 4
(Cube Based) The volume of 4-dimensional (hyper-)sphere of
redius r=1 centered on the origin is between 4.85149-4.95093
(4.90121)
Run time: 10.8223
sh: Pause: command not found
Program ended with exit code: 0
```

All Output Filter

k=1000 Run time:699.042s accuracy:2

```
(Cube Based) Insert the dimension: 4
(Cube Based) The volume of 4-dimensional (hyper-)sphere of
redius r=1 centered on the origin is between 4.90136-4.94099
(4.92118)
Run time: 699.042
sh: Pause: command not found
Program ended with exit code: 0
```

All Output Filter

Dimension 5: **k=200 Run time:132.512s accuracy:1**

```
(Cube Based) Insert the dimension: 5
(Cube Based) The volume of 5-dimensional (hyper-)sphere of
redius r=1 centered on the origin is between 5.02206-5.36472
(5.19339)
Run time: 132.512
sh: Pause: command not found
Program ended with exit code: 0
```

All Output Filter

Bugs

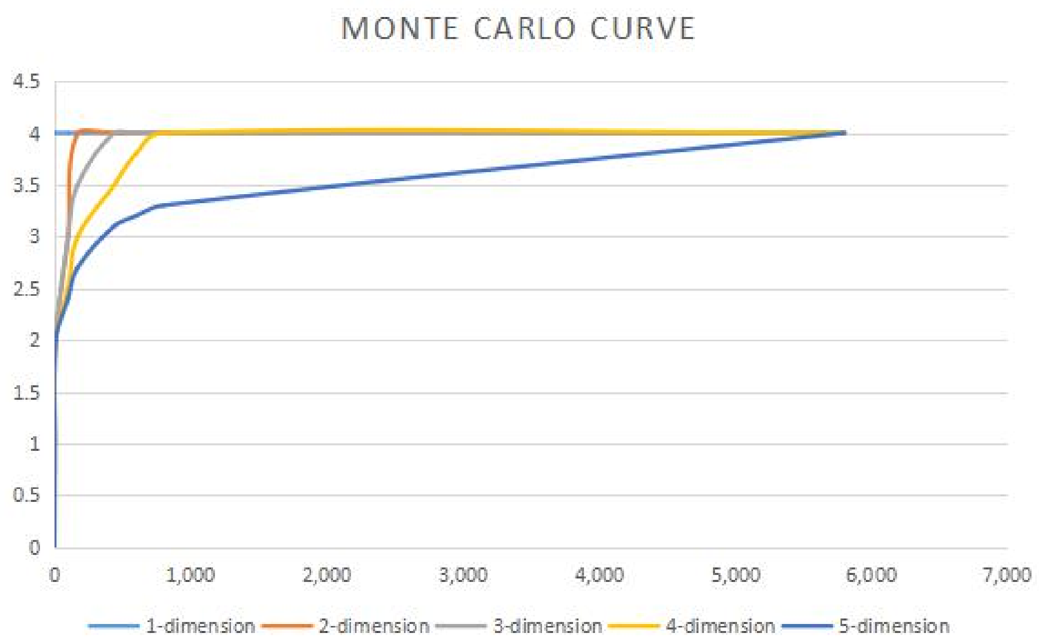
Monte Carlo: No bugs, perfect.

Cube Based: There is a little bug about the coordinate, when I check cube coordinates, I found the last coordinate is above the normal coordinate, but there is only one cube is abnormal, so I think it could be ignored. Beside this, I didn't find any bugs.

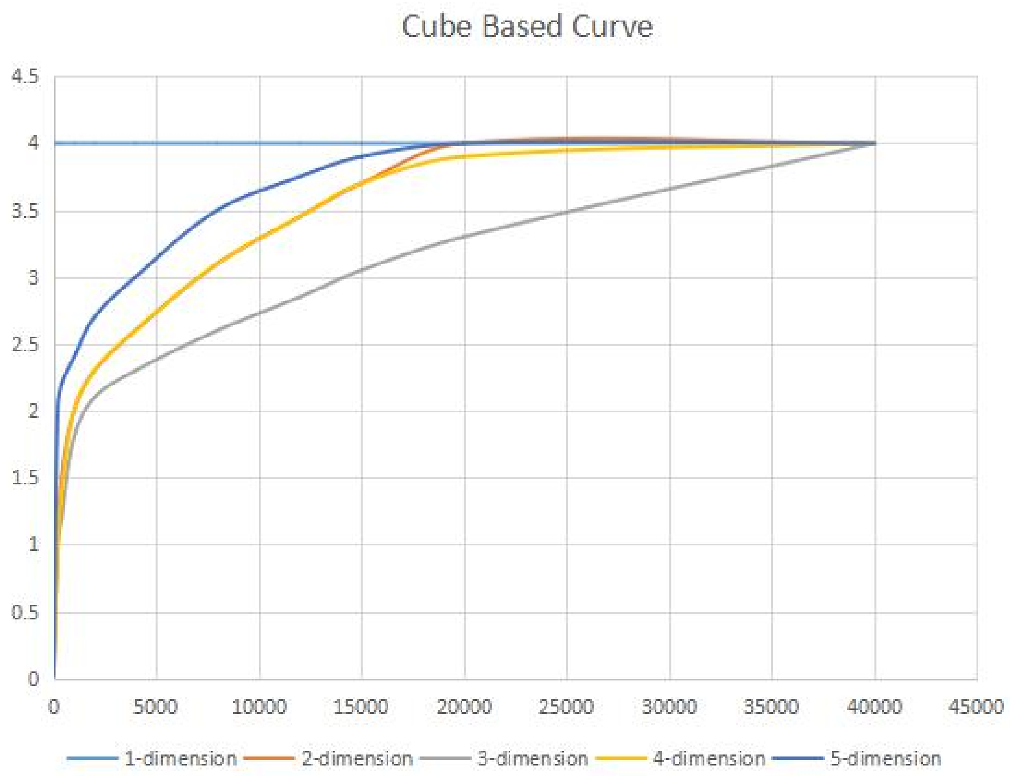
But I fixed many bugs through I run my programs. The most memorable bug is when my cube number is above 2,100,000,000, my program doesn't work. But later I found the max length of int is about 2,100,000,000, so I used long long instead and the program is perfect run!

Part1

Based on the accuracy and run time, the more efficient method is Monte Carlo method as d becomes large on 4 digits of precision. On 4 digits of precision, I could push Monte Carlo to 5 dimensions or more and Cube Based to about 3 dimensions. My time is not enough to run I believe it is just depended on the computer. In 3-dimension Cube Based, I could run to K=15000 to pretty close to 4 digits accuracy. In higher dimension, I could run to 2 digits in my computer fast.



Here is the Monte Carlo accuracy plot and run times as a function of d. The x-coordinate unit is million samples, y-coordinate is accuracy numbers. Because accuracy is specific numbers, so the curves may not look so smooth. 1-dimension is light blue line; 2-dimension is orange line; 3-dimension is gray line; 4-dimension is yellow line; 5-dimension is dark blue; Because I only calculate several points, so the plot may not very smooth. But you can see the tendency, as the dimension increases, the curve is changed “rationally”.



Unlike the Monte Carlo, the cube based plot is pretty interesting. The x-coordinate is k, y-coordinate is accuracy numbers. 1-dimension is light blue; 3-dimension is gray line; 5 dimension is dark blue line; 2-dimension and 4-dimension are nearly overlaps which is the yellow line. Here is some amazing things I found:

Why there is a rebound tendency for higher dimensions? I think it's a little magic but rational because in higher dimensions, small k can still reach trillion trillion samples and the more samples there is, the more accuracy the calculate is. So, small k could also reach 4 digits accuracy in higher dimensions because there is much more cubes! And once the sample numbers are reached a super big boundary, the accuracy could be guaranteed which means the cross circle cubes could be ignored and the requirement k value will decrease while the total number of cubes will still increase.

We can see in the plot, 1,2,3-diemnsions(light blue, yellow, gray line) are arranged orderly, but once sample numbers reach the boundary, there will be a rebound tendency like 4,5-dimensions which are yellow and dark blue line. And my test data is
confirmed to this tendency. When k=1000, 3-dimension could only make sure 1 accuracy but 4-dimension is 2 accuracy.Amazing!

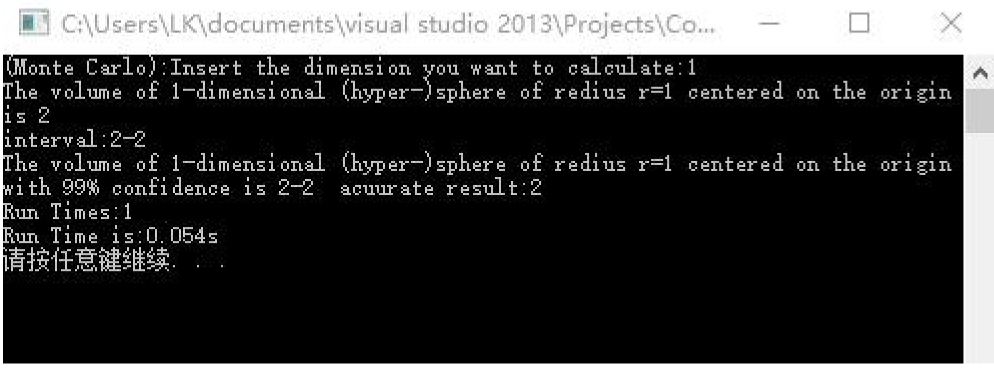
Here is another question: How do things change if we demand 8 digits of precision with 99.9% confidence? I think this is very simple. It will need more samples and better computer computers to 8 digits of precision. I tried to run to 8 digits of precision, fix double to long double, but it took so long time to run even in the 2-dimension Monte Carlo. So this demand a super fast computer and at least 1000 times of time.

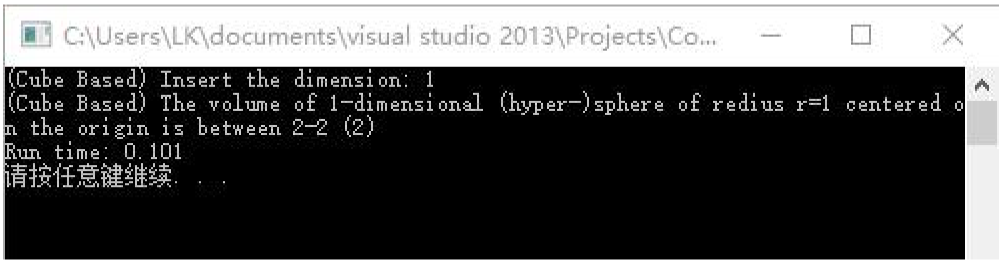
Part2

To compare the CPU cost, I didn't run test version. I run a normal version of cube based which is go over all the cubes and "circles". Here is my result: (Note:For Cube based, the value in the bracket is the accurate number which is plus 1/2 cross circle cubes; For Monte Carlo, you can ignore the confidence interval, just focus on the distance and run time)

Dimension 1: N=1,000,000 K=1,000,000

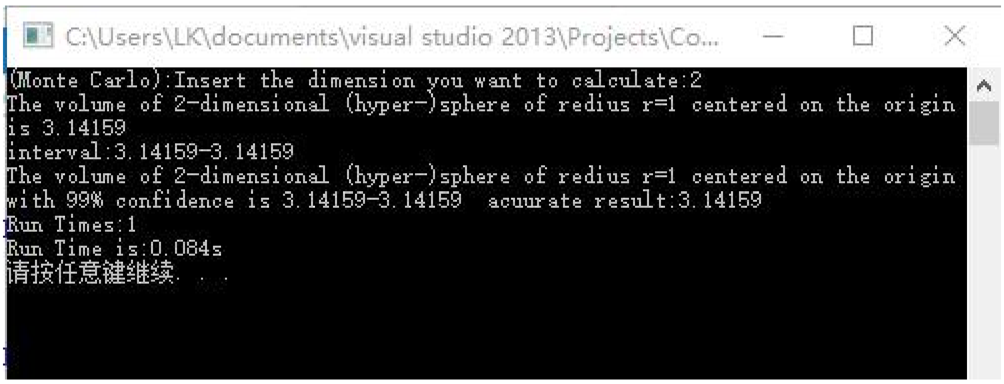
In this dimension, the result is nearly the same.

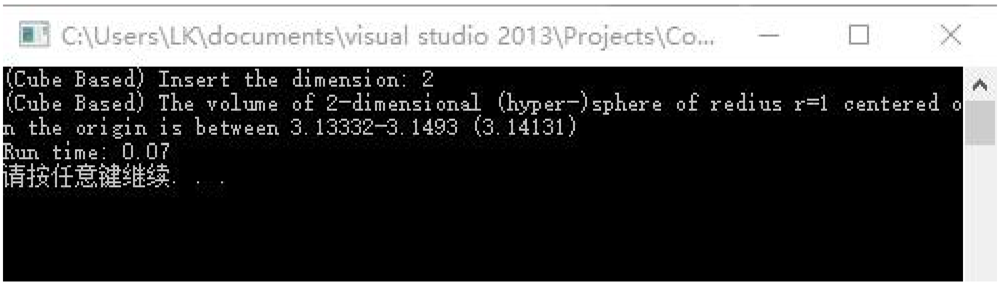




Dimension 2: N=1,000,000 K=1,000

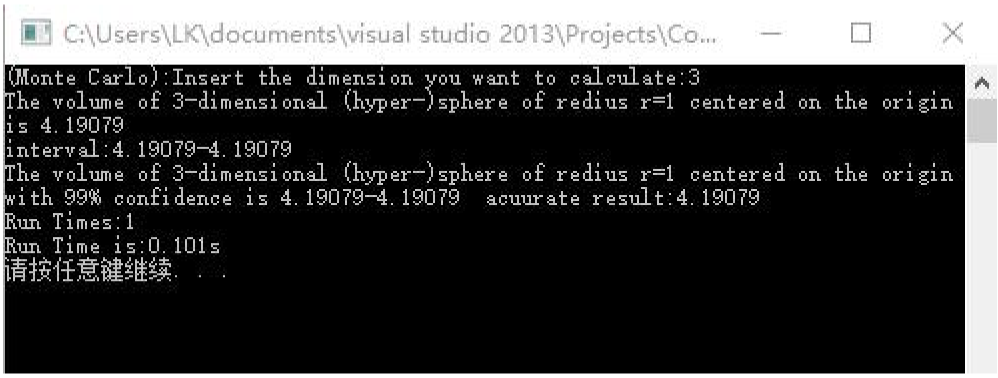
In this dimension, the time is about the same but the accuracy is different. Monte Carlo is 3.14159 which is 6 digits accuracy. Cube Based is 3.14131 which is 4 digits accuracy. They both get at least 4 digits accuracy. Great!

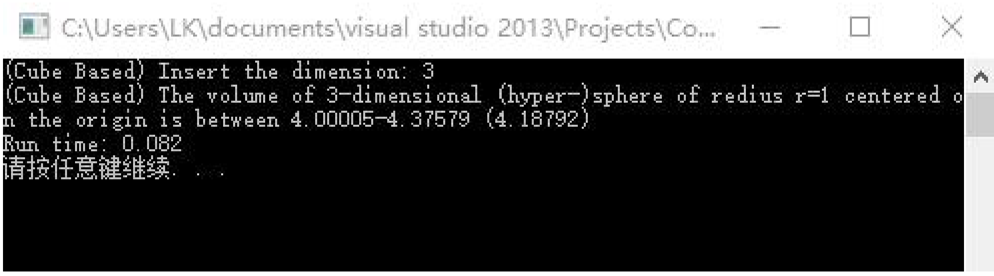




Dimension 3: N=1,000,000 K=100

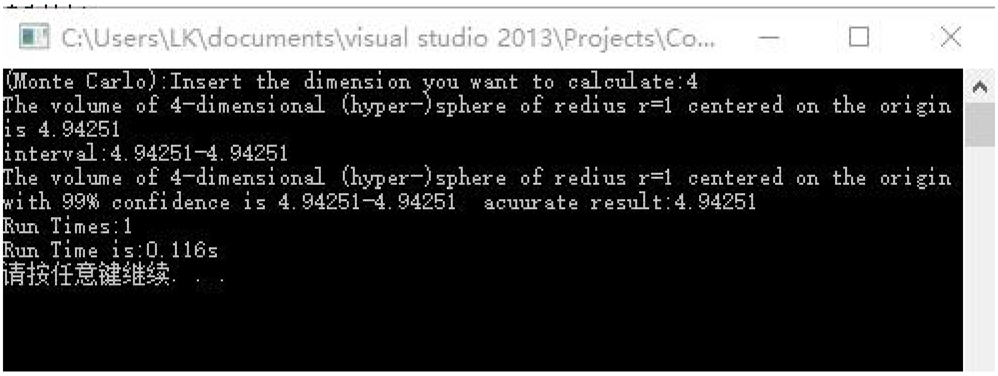
In this dimension, the time is also about the same. But in accuracy, this time Cube Based algorithm which is 4.18792 with 3 digits accuracy won the Monte Carlo 4.19079 with 2 digits accuracy! Interesting!

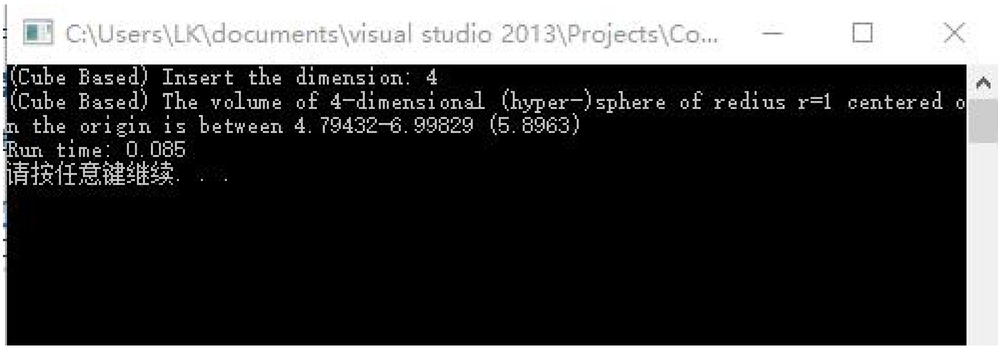




Dimension 4: N=1,000,000 K=32

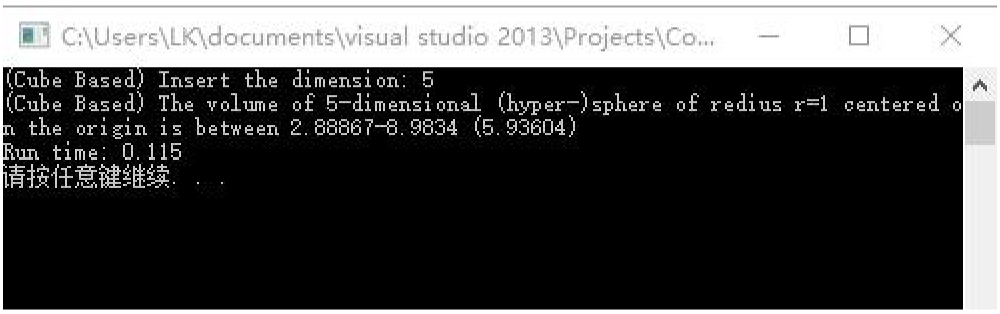
From this dimension, Monte Carlo Method overwhelmed Cube Based Method totally. Monte Carlo Method is 2 digits accuracy while Cube Based is 0. Because although the one million sounds big, when “sqrt” and “sqrt”, the k is only 32. This is why I think in higher dimension there is a rebound tendency in the curve plot.





Dimension 5: N=1,000,000 K=16

In this dimension, The precision of Monte Carlo Method is 3 while the cube based method is 1. And there run time is also similar.



In conclude, Monte Carlo again, gives the more accurate answer. Although in 3-dimension, cube won but in general, Monte Carlo is more accurate, especially in higher dimensions. And in tun time, cube based is a little bit faster maybe because I didn't delete the confidence interval in Monte Carlo method.

Part 3

Based on my test data and experience, If I was given a constant N, I think I should use 2-dimension and Monte Carlo method to compute for π. And in Monte Carlo Method, it could use variance reduction techniques to increase the accuracy. Cube Based couldn't be used a variance reduction because the boundaries are strict.

As Dr,Hayes said, we can use (uniform distribution) and (1-uniform distribution) to reduce variance. For example, if we have N uniform distributions, we can use another sample which is N (1-uniform distributions). So we can increase the sample numbers and reduce variance of mean of (uniform distribution numbers + (1-uniform distribution numbers))/2. This algorithm is easy to realize I believe.

Feeling

I really like this challenging but interesting class. Dr.Hayes is strict but really erudite and I admired him a lot. I could also feel a strong academic atmosphere in this class. This kind of feeling is what I am searching for. I love it.

By the way, I am an international student and this is my first quarter in UCI. In assignment 1, I didn't know to submit write-up, so I just got 60. This time, I took seriously about this write-up and I hope my final score could cover the shortage of my first assignment.