cision of 10^{-5}. The fractional error was at 5.84427×10^{-5}, and this trend could be seen by the previous figures as most converge at a certain point very close to the true value before diverging.

Table 2: *Volumes of the n-dimensional hyper sphere trials and correlating fractional error*

Dimensions	*Estimated Volume*	*Theoretical Volume*	Fractional Error
2	3.15680	3.141592654	0.00484065
3	4.22640	4.188790204	0.00897868
4	5.03360	4.934802202	0.02002060
5	5.32800	5.263789015	0.01219860
6	5.06240	5.167712783	0.02037900
7	4.71040	4.724765972	0.00304057
8	3.66080	4.058712129	0.09803900
9	3.07200	3.298508904	0.06867010
10	1.84320	2.550164042	0.27722300

Table 2 refers to figure 5, as the number of dimensions are plotted against the volume. The fractional errors are larger for the similar n-term iterations for the higher dimensions. The higher fractional errors are due to the increased dimensions, therefore more n-trial iterations would be needed for a more precise volume estimation at larger dimensions.

5 Conclusion

In conclusion, the amount of N-trials allowed for more precise volume measurements of the n-dimensional hyper sphere and pond. Fractional error also decreased showing that there was a steady convergence to the actual volume value. These facts proved that taking a large amount of n-trial iterations allowed the Monte Carlo integration to be highly effective for these volume estimations.

References

[1] `http://www.phys.hawaii.edu/~gorham/p305/MonteCarlo1.html`

[2] `http://dlmf.nist.gov/5.19#iii`

[3] http://tohtml.com/cpp/

[4] http://www.wolframalpha.com/

[5] Landau, Rubin H., Manuel J. Paez, and Christian C. Bordeianu. (2007). Computational Physics: Problem Solving with Computation, 2nd edition. KGaA: Wiley-VCH.

[6] http://mathworld.wolfram.com/SpherePointPicking.html

6 Appendix

6.1 Pond Program for 3-Dimensions

```
#include <iostream>
#include <iomanip>   // this is required to use "setprecision" below
#include <fstream>
#define _USE_MATH_DEFINES
#include <cmath>
#include <cstdlib>

using namespace std;

#define PI  M_PI
//#define max 10000
#define seed 681111


int main(int argc, char *argv[])
{
int max = atoi(argv[1]);
   double x[max], y[max], z[max], r[max];  // allocate arrays
   double  hits=0.0, ntrials=0.0;
   ofstream outfile;
   outfile.open("pond3.dat");
   srand48(seed);                        // seed the number generator

      for (int i=0; i<max; i++){
         x[i] = (drand48()-0.5)*2.;      // x and y between
         y[i] = (drand48()-0.5)*2.;      // -1 and 1
         z[i] = (drand48()-0.5)*2.;

          r[i] = sqrt( pow(x[i],2) + pow(y[i],2) + pow(z[i],2));
         //r[i] = hypot(x[i],y[i]);        // distance from origin

 hits += r[i]<=1.0 ? 1.0 : 0.0;  // conditional expression
 ntrials += 1.0;
      }
   double volume = hits/ntrials*8.0;
   double vtrue = ((4./3.)*M_PI);
   cout << "Volume of inscribed circle= "<< setprecision(9) <<
    volume <<" vtrue= " <<vtrue <<" , fractional error= "<<
    fabs((volume-((4.0/3.0)*PI))/(PI*(4.0/3.0)))<<endl;
   for (int i=0; i<max; i++){ // write results into file, hits within circle
     if(r[i]<=1.0)outfile << x[i]<<" "<< y[i] <<" "<< z[i] <<endl;
   }
   outfile << "\n\n";
   for (int i=0; i<max; i++){ // write results into file, hits outside circle
     if(r[i]>1.0) outfile << x[i]<<" "<< y[i] <<" "<< z[i] <<endl;
   }
cerr << "data stored in pond3.dat" << endl;
```

```
   outfile.close();
}
```

6.2 Hyper sphere Program for n-Dimensions

```
#include <iostream>
#include <iomanip>   // required to use "setprecision" if needed
#include <fstream>
#define _USE_MATH_DEFINES
#include <cmath>

using namespace std;

#define PI  M_PI
#define max 1000
#define seed 681111

int main(int argc, char *argv[])
{
    ofstream outfile;
    outfile.open("hypen.dat");
    double hit,xi,R,Rsq,D,R_D,Vtot = 0.0,Vsphere = 0.0, Vtrue = 0.0;
    int n,i, NDIM, Ntrials;
srand48(1299811); // a large prime

if(argc<2){
cerr<< "usage: hypersphereMC [NDIM][NMAX]"<<endl;
exit(0);
}

NDIM = atoi(argv[1]); // number of dimensions
Ntrials = atoi(argv[2]); // number of trials

D= 2.0; // side of hypercube needed to contain hypersphere
R_D = 1.0; // radius of hypersphere
hit = 0.0; // the counter for events within sphere
n=0; // initialize the loop counter

    while(n<Ntrials){  // continue generating coordinates up to Ntrials
    Rsq = 0.0;  // this variable accumulates the square of each coordinate
    for(i=0;i<NDIM;i++){
            Rsq += pow((drand48()-0.5)*2.,2);
            //sum up the squares to get distance from origin
} // end of NDIM loop------------
        R = sqrt(Rsq);
        hit += R<=1.0 ? 1.0 : 0.0; //check if distance Rsq falls within R_D boundary,
        n++; // counter for Ntrials while loop
    }  //---------END OF WHILE LOOP-----------------
Vtot = D; // Vtot=D for 1-dimensional "hypercube" (a line)
 for(i=1;i<NDIM;i++){
        Vtot *= D;  //interative multiplication to get N-dim hypercube
```

```
    }
    //cout <<"Hits: "<< hit << " Ntrials: " <<Ntrials <<endl;

    Vsphere = ((hit)/(Ntrials))*Vtot;  //determine the hit ratio & estimated Volume
    Vtrue = pow(M_PI,((NDIM)/2.0))/tgamma(((NDIM)/2.0)+1.0)*pow(R_D,NDIM);
    //true volume from analytic formula

    for (int i=0; i<max; i++){ // write results into file, hits within circle
        if(Rsq<=1.0)outfile << i<<" "<< Vsphere <<" "<< Vtrue <<endl;
    }
    outfile << "\n\n";
    for (int i=0; i<max; i++){ // write results into file, hits outside circle
        if(Rsq>1.0) outfile << i<<" "<< Vsphere <<" "<< Vtrue <<endl;
    }
 cout<< NDIM <<" dimensions, Vsphere= "<< Vsphere <<" Vtrue= " <<Vtrue<< "  Vtot=
 "<<Vtot<< " fractional error= "<<abs(Vtrue-Vsphere)/Vtrue<<endl;
    cerr << "data stored in hypen.dat" << endl;
    outfile.close();
}
```