Numerical Integration - The Monte Carlo Simulation

B. C. Chap *

Winter Quarter 2021

Course: Computational Methods of Mathematical Physics
Instructor: Professor D. Ferenc
University of California, Davis
One Shields Ave., Davis, CA 95616, USA

Abstract

The Monte Carlo integration techniques differ from other common integration techniques such as Simpson's rules, rectangular, and trapezoidal summation in that it does not require evenly spaced evaluation points of the function. This is especially useful for multi-dimensional and select types of improper integrals [1]. Rather than utilizing predetermined points prior to integration, the Monte Carlo method takes a random sampling of points within an interval where all points have equal probability of occurring [1]. In this project, we will utilize the Monte Carlo integration technique to evaluate 1-D, 3-D, and improper integrals in the form of the normal probability function, the Yukawa charge distribution, and an inverse trigonometric identity.

1 Introduction

1.1 Monte Carlo Integration and Random Sampling Generation

The Monte Carlo technique relies on the generation of a random sampling of size N, where the randomness of our sample $x_0, x_1, ..., x_N$ have equal probability of occurring between the limits of integration $[a, b]$. Note that equal probability of these points is entails that these points are evenly distributed, thus are not Gaussian or Poissonian. While this project may evaluate a normal probability function, this is not to be confused with the random sampling generated for the evaluation points. The simplest integrals have limits $[a, b] = [0, 1]$. For random sample values where $[a, b]! = [0, 1]$ the following equation is a quick transformation, which makes it so that the numerical integration is carried out on the interval $[0, 1]$ [1]:

$$x \longrightarrow \frac{1}{b-a}(x-a) \tag{1}$$

For a sampling size with large N, the average distance between two neighboring points is:

$$h = \frac{1}{1-N} \underset{largeN}{\longrightarrow} \frac{1}{N} \tag{2}$$

Thus, the value of the integral becomes:

$$f_{[a,b]} = \frac{1}{b-a}\int_a^b f(x)dx \underset{N\to\infty}{\longrightarrow} \frac{1}{N}\sum_{i=1}^{N} f(x_i) \tag{3}$$

If we binned our random sampling, regardless of the value of x, the number of points that falls into one of the bins would be about the same no matter how many sub-intervals there are [1]. To ensure that

*bcchap@ucdavis.edu

the random sampling size N is large enough to satisfy (2), an iterative approach is employed. To evaluate the 1-D, 3-D, and improper integrals using the Monte Carlo Method, the random values within a certain range $a, b]$ are generated. We first begin by generating two random samples, both of initial size N. We then take the average of these two samples and compare them.If the difference between those averaged values are above a certain tolerance, the value of N is then doubled or tripled and the two random samples are regenerated with the new size N. This process is repeated until the maximum number of iterations occurs or they difference between the averages fall within the tolerance necessary. Once this tolerance is met, we then take the final two averaged values and their sufficiently large corresponding N and again average them, thus achieving sufficient accuracy. The following sections expand on this process to solve the integrals of the normal probability function, the Yukawa charge distribution, and an inverse trigonometric identity.

1.2 1-D Integrals - The Normal Probability Function

To examine the Monte Carlo integration technique for a 1-D integral, we will examine the normal probability function:

$$A(x) = \frac{1}{\sqrt{2\pi}} \int_{-x}^{x} e^{\frac{-t^2}{2}} dt \tag{4}$$

To simplify our work, we will examine the case of $x = 1$, and exploit the symmetry of the function so that we are able to utilize a uniformly distributed random generator with the interval $[0, 1]$, yielding:

$$A(x = 1) = \sqrt{\frac{2}{\pi}} \int_{0}^{1} e^{\frac{-t^2}{2}} dt \tag{5}$$

The expected value of A is the area underneath the normal distribution curve, and thus:

$$A(x = 1) = 2(\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{1} e^{\frac{-t^2}{2}} dt - \frac{1}{2}) = 0.6826895 \tag{6}$$

1.3 3-D Integrals - The Yukawa Charge Distribution

The Monte Carlo method is extremely useful for integrals with n independent variables, where the amount of computation is proportional to N^n, this is due to the fact that the mesh requires integration over all space, whereas the Monte Carlo method relies on newly generated random variables that represent the overall volume through each iteration [1].

Utilizing the same method, we take a 3-D integral and average the evaluated function over all space:

$$I = \int_{e}^{f} \int_{c}^{d} \int_{a}^{b} f(x)f(y)f(z)dxdydz \tag{7}$$

$$I = \sum_{i=1}^{N} f(x_i)f(y_i)f(z_i)\Delta v \tag{8}$$

where

$$\Delta v = \frac{(b-a)(d-c)(f-e)}{N} \tag{9}$$

The benefit of the Monte Carlo method is that there is no need to integrate over all space, a random sampling with a sufficiently large N of n the number of integration variables [1]. For smooth functions, high accuracy can be achieved even with low N values.

To explore the Monte Carlo method for a 3-D space, we examine the Yukawa charge potential confined to a cubic volume of length 2 on each side where the charge distribution $\rho(x, y, z)is$:

$$\rho(x,y,z) = \frac{e^{-r}}{8\pi} \tag{10}$$

where

$$r = \sqrt{x^2+y^2+z^2} \tag{11}$$

The total charge inside this volume then becomes:

$$Q = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} \frac{e^{-r}}{8\pi} dxdydz \tag{12}$$

Because of the symmetry of this function, we are able to use the transformation (1) for all variables so that the limits of integration for all space are $[0,1]$ so that the integral becomes:

$$Q = \int_{0}^{1}\int_{0}^{1}\int_{0}^{1} \frac{e^{-r}}{\pi} dxdydz \tag{13}$$

The expected result is $Q \approx \frac{.4}{\pi}$, and an accuracy of .0002 should be reached by $N = 8000$. The solution of the Yukawa charge distributions confined to a box begins with initial random sampling size $N = 100$, max iteration number $MAX\_IT = 15$, tolerance $TOLERANCE = 10^{-4}$, and initial seed $SEED = 1$.

We further explore the limit of this charge distribution by examining the way in which the charge drops off as a function of distance from the center of the cube. To do so, we approximate the charge distribution for all space by utilizing the following calculations [1]:

$$\begin{aligned} Q &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{e^{-r}}{8\pi} dxdydz \\ &= \int_{0}^{\infty}\int_{0}^{\pi}\int_{0}^{\pi} \frac{e^{-r}}{8\pi} sin\theta d\phi d\theta dr \\ &= \frac{1}{2}\int_{0}^{\infty} \frac{e^{-r}}{r^2} dr \\ &= 1 \end{aligned} \tag{14}$$

1.4 Improper Integrals

Improper integrals either have integration limits containing infinities or points of evaluation of the integral that yield infinities. One such example is the relation between the derivative of a commonly known trigonometric identity $arcsin(x)$. However, say we did not recognize this fact, for $x = 1$ we might still know that the integral still exists, and thus we have an integrable singularity [1]. The benefit of the Monte Carlo method in such case is that the occurrence of a singularity within the integration domain is quite small. This is due to the fact that all points within the randomly generated sample are equally probable. To account for this possibility however, one can simply discard the sample point and replace it with a newly generated random variable without sacrificing much accuracy. This way, even if the knowledge of the singularity is not apparent prior to the solving the improper integral, it is essentially accounted for and mostly ignored. Let us further examine the following function:

$$I(B) = \int_{0}^{B} \frac{dx}{\sqrt{1-x^2}} = arcsin(x) \tag{15}$$

The following section outlines the process for each of these integrals.

2 Procedure

The goal is to solve integrals utilizing the Monte Carlo method, generating random variables that are then used to evaluate the function until a given accuracy is reached. The following are the steps to complete this process, with notes on minor adjustments for each integral.

Language Used: Python 3

Argument List:

- SEED: Initial SEED value to initialize the random generator being used, set to 1 for all random number generators
- N: The initial sampling size
- TRU_ VAL: The accepted or expected values for each integral
- MAX_ IT: The max iteration value
- TOLERANCE: The threshold for the difference between two evaluated integrals from two separate random samples of the same size, i.e, the maximum error tolerated

  - Normal Probability Function: $[a, b] = [0, 1]$ and there is 1 independent integration variable x
  - Yukawa Charge Distribution: $[a, b] = [-1, 1]$ and there are 3 independent integration variables x, y, z, utilize (1) and take advantage of symmetry to transform the integral limits to $[0, 1]$
  - Yukawa Limit: $[a, b] = [-10, 10]$ and there are 3 independent integration variables x, y, z, do not forget to utilize (1)
  - Inverse Trigonometric Identity: $[a, b] = [0, 1]$ and there is 1 independent integration variable x, do not forget to utilize (1)

1. Random Number Generation:

2. Create two arrays for each independent variable n

3. Generate N random numbers within the interval $[a, b]$ for each array previously created

4. Integrand Evaluation:

5. Define the function within the integral, aka the integrand which utilizes the values stored within the random variable arrays

6. Create two arrays to fill with the evaluated integrand values, and another array to fill with the means of the two integrands evaluated at the end of every iteration

7. Use the random numbers generated for each independent variable to evaluate the integrand function twice, the first evaluation going into the first array created and the second going into the second array created

   - At the end of the iteration (when N evaluated integrand values have been put into the two integrand arrays, take the average of the two and compare them
   - If the difference between the two is greater than the THRESHOLD, put the average of the two evaluated integrands into the means array created for the end of every iteration, then double or triple N and continue iterating until all iterations occur OR until the THRESHOLD is met.

- If the THRESHOLD is met before the max number of iterations MAX_IT occurs, then BREAK the cycle

8. Return the mean of the two final evaluated integrands, their corresponding N value, and the percent error between the accepted integral solution and the Monte Carlo calculated solution

9. Visualizing Accuracy

10. Plot the accepted value for the evaluated integral

11. Plot the iteration number versus the Monte Carlo simulated values in the integrand means array

12. Examine how the Monte-Carlo simulation increases in accuracy as N increases

3 Discussion

The percent error and following figures for each integral are the most telling. Comparing the expected value to the Monte Carlo calculated value for each integral, each was evaluated at less than 1% error, thus proving that the Monte Carlo integration was correct. We expect to see increasing accuracy as the random sample size N increased, which was the case for every single integral. This is exemplified by the plots of iterations vs the Monte Carlo evaluated integrands fluctuating above and below the accepted value with decreasing amplitude with every iteration.

3.1 The Normal Probability Function

The following plot describes the convergence of the Monte Carlo Integration for (5) and following is its corresponding plot. For an initial value of $N = 40$, we expect by the that by 14th iteration, a difference of one part in 10^5 should be met [1]. The solution of the normal probability function then begins with initial random sampling size $N = 40$, max iteration number $MAX\_IT = 15$, tolerance $TOLERANCE = 10^{-5}$, and initial seed $SEED = 1$ does indeed accomplish this, resulting in a percent error of .002%

Table 1: Convergence of a Monte Carlo Integration for $A(x) = \sqrt{\frac{2}{\pi}} \int_0^1 e^{\frac{-t^2}{2}} dt$

Iteration	# of Points	A(x=1)
1	40	0.6867851098753224
2	80	0.6827223605833177
3	160	0.6782968721425169
4	320	0.6791897825728199
5	640	0.6839252189781416
6	1280	0.6826866717543368
7	2560	0.6839656769022655
8	5120	0.6820807566146405
9	10240	0.6824543820589903
10	20480	0.6823309840258357
11	40960	0.6826068869178903
12	81920	0.6826820310248631
13	163840	0.6826083559977807
14	327680	0.6827728215487106
15	655360	0.6826746800346912
Exact Value		0.6826895
Percent Error		0.002170820747751517 3%

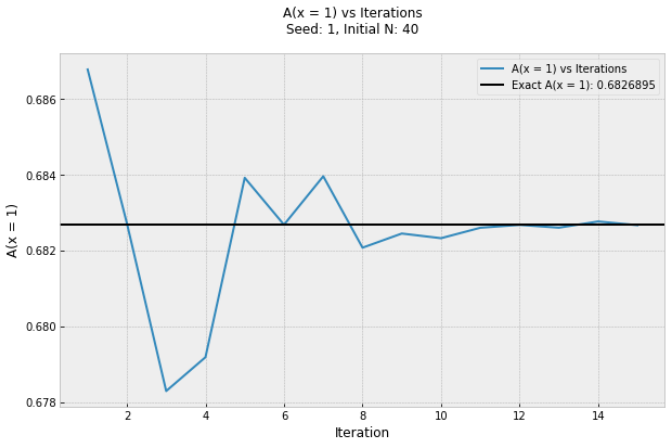

Figure 1: Monte Carlo Simulation to Solve the Normal Probability Function (5)

3.2 Yukawa Charge Distribution Confined to a Box and for All Space

The Yukawa Charge Distribution confined to a cubic box provided an opportunity to utilize the transformation (1) to change the limits of integration from $[-1, 1]$ to $[0, 1]$ for all variables. Integration over all space must account for the volume being examined, thus an addition of the average volume (9) is utilized. The expected result for the 8 cubic volume is $Q \approx \frac{.4}{\pi}$, and an accuracy of .0002 should be reached by $N = 8000$, which was indeed met by the 8000th iteration. The solution of the Yukawa charge distributions confined to a box begins with initial random sampling size $N = 100$, max iteration number $MAX\_IT = 15$, tolerance $TOLERANCE = 10^{-4}$, and initial seed $SEED = 1$ had a percent error of .008%. Due to the Yukawa Charge Distribution being centralized, we also examine the limit as the volume confining the charge approaches infinity, encompassing all space. Because the charge is so centralized, utilizing a 20^3 volume is adequate to examine this. The expected value of Q for all space is $Q = 1$. We also expect to get an accuracy of 10^{-3} for samplings on the order of $N = 10^3$ [1]. The Monte Carlo evaluated integral was within .35% of this and accomplishes attaining the accuracy within the expect N values.

Table 2: Convergence of a Monte Carlo Integration for $Q(x, y, z) = \int_0^1 \int_0^1 \int_0^1 \frac{e^{-r}}{\pi} dxdydz$

Iteration	# of Points	Q([0,1], [0,1], [0,1])
1	100	0.1263639160918505
2	200	0.12589156596932652
3	400	0.12643873661588578
4	800	0.12743254061418705
5	1600	0.1268403167841383
6	3200	0.12634416142020907
7	6400	0.1265129269934237
8	12800	0.1266639508485763
9	25600	0.12690012406546822
10	51200	0.1266848646619518
11	102400	0.1267595509791484
12	204800	0.1267647249570894
13	409600	0.12683386266518165
14	819200	0.12674698714584542
Exact Value		.126758
Percent Error		0.00868809397007296 %

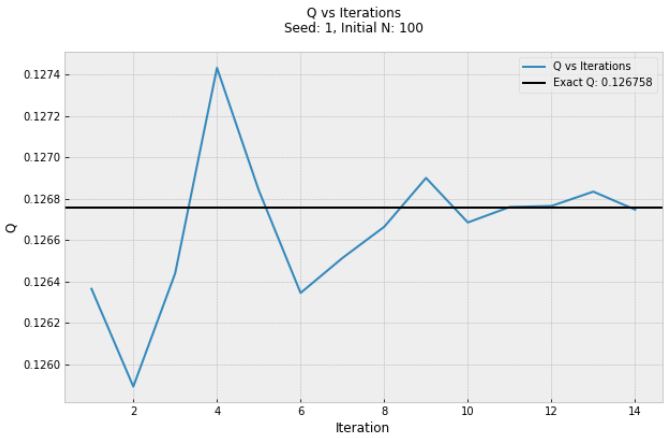

Figure 2: Monte Carlo Simulation to Solve the Yukawa Charge Distribution Confined to a Box (13)

Table 3: Convergence of a Monte Carlo Integration for the limit of $Q(x, y, z)$ for all space

Iteration	# of Points	Q([-10,10], [-10,10], [-10,10])
1	100	0.6051339228470939
2	200	1.5445371409874094
3	400	0.8547673577855208
4	800	1.1244138483741108
5	1600	0.9933791155478723
6	3200	1.0688533585083329
7	6400	0.9804624056240279
8	12800	0.9722592335220719
9	25600	1.0150845774197923
10	51200	0.9997663719378971
11	102400	1.0063490998962905
12	204800	1.000720520236226
13	409600	1.0008267779517654
14	819200	0.996504184636933
Exact Value		1
Percent Error		0.349581536306697 %

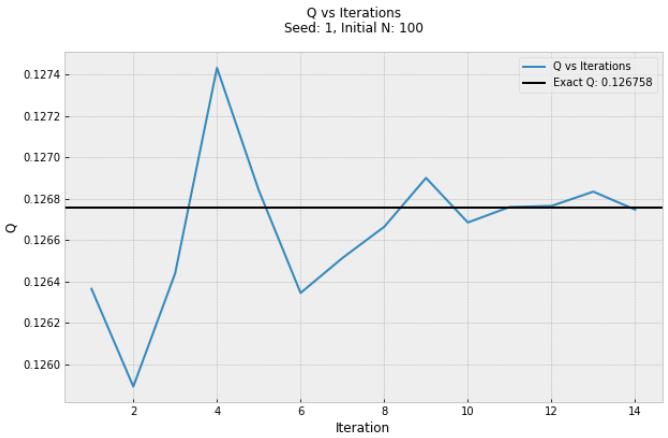

Figure 3: Monte Carlo Simulation to Solve the Yukawa Charge Distribution for All Space 14

3.3 Improper Integral, arcsin(B) Identity

It has already been established that integral limits $[a, b] = [0, 1]$ are the most simple to solve, however, for values of $b! = 1$, we must remember to account for the average value of x using (1). Compared to the previous methods for the 1-D and 3-D integrals, we wish to sum the values of the integrand not at the edge of the bins, but at the midpoints, thus in the middle of each sub-interval [1]. To account for this summing of midpoints, we increase the iterative process from doubling to tripling. Because the sampling size becomes so large, the initial sample size is set to $N = 50$, max iteration number $MAX\_IT = 10$, tolerance $TOLERANCE = 10^{-5}$, and initial seed $SEED = 1$. This method was tested for a variety of upper limits $B = .25, .5, .75, \& \ 1$ and compared to the actual value of $arcsin(B)$. The range of percent errors for all B values was $.004 - .05$, this high accuracy despite low iterations is most likely due to the fact that the value of N was tripled rather than double to ensure (2) was satisfied.

Table 4: Convergence of a Monte Carlo Integration for $I(.25) = \int_0^{.25} \frac{dx}{\sqrt{1-x^2}}$

Iteration	# of Points	$arcsin(.25)$
1	50	0.25275938420641353
2	150	0.2525703445196261
3	450	0.25281885852882924
4	1350	0.25266295490964463
5	4050	0.2526751601428157
6	12150	0.2526880583618363
Exact Value		0.25268025514207865
Percent Error		0.0030881794674624606%

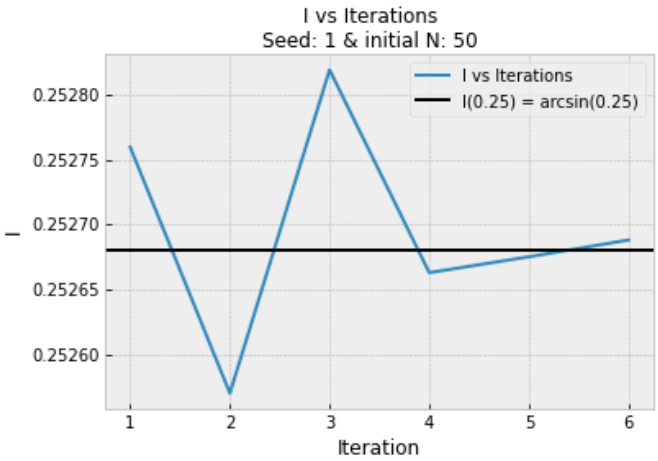

Figure 4: Monte Carlo Simulation to Solve $I(.25) = \int_0^{.25} \frac{dx}{\sqrt{1-x^2}}$ (15)

Table 5: Convergence of a Monte Carlo Integration for $I(.50) = \int_0^{.5} \frac{dx}{\sqrt{1-x^2}}$

Iteration	# of Points	$arcsin(.5)$
1	50	0.5242536945505532
2	150	0.5226700831066066
3	450	0.5248248149822831
4	1350	0.523471487853627
5	4050	0.5235621201166236
6	12150	0.5236759245561
Exact Value		0.5235987755982989
Percent Error		0.014734365586117348%

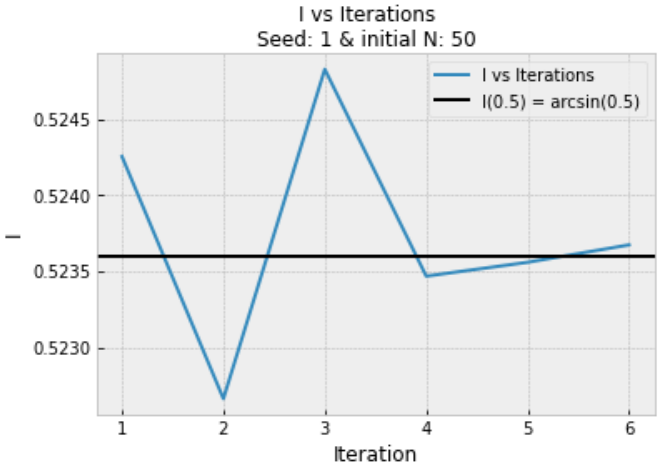

Figure 5: Monte Carlo Simulation to Solve $I(.5) = \int_0^{.5} \frac{dx}{\sqrt{1-x^2}}$ (15)

Table 6: Convergence of a Monte Carlo Integration for $I(.75) = \int_0^{.75} \frac{dx}{\sqrt{1-x^2}}$

Iteration	# of Points	$arcsin(.75)$
1	50	0.850330294201912
2	150	0.844639486483283
3	450	0.8531888624347559
4	1350	0.847800872769642
5	4050	0.8480177539708462
6	12150	0.848469421586566
7	36450	0.8481508244087932
8	109350	0.8481321728275252
9	328050	0.8481376497484342
Exact Value		0.848062078981481
Percent Error		0.048032168302376124%

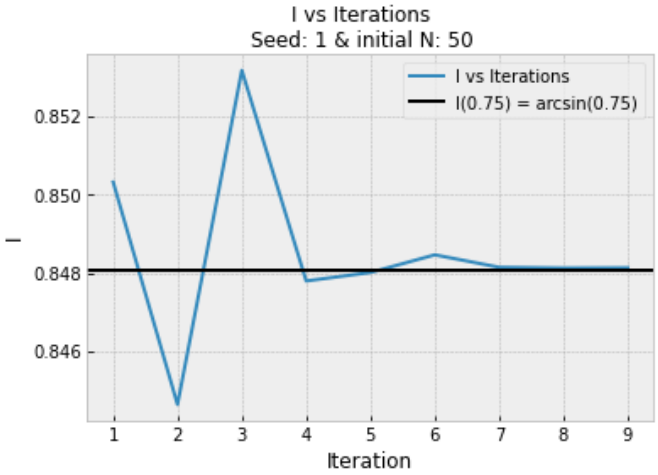

Figure 6: Monte Carlo Simulation to Solve $I(.75) = \int_0^{.75} \frac{dx}{\sqrt{1-x^2}}$ (15)

Table 7: Convergence of a Monte Carlo Integration for $I(1.0) = \int_0^{1.0} \frac{dx}{\sqrt{1-x^2}}$

Iteration	# of Points	$arcsin(1)$
1	50	1.5144419885812104
2	150	1.487487485037593
3	450	1.5624148435307175
4	1350	1.5586924436128342
5	4050	1.565610194153601
6	12150	1.585565347110705
7	36450	1.5767232830749167
8	109350	1.5674020444728443
9	328050	1.5691059985724778
10	984150	1.5711995209038556
Exact Value		1.5707963267948966
Percent Error		0.02566813418654663%

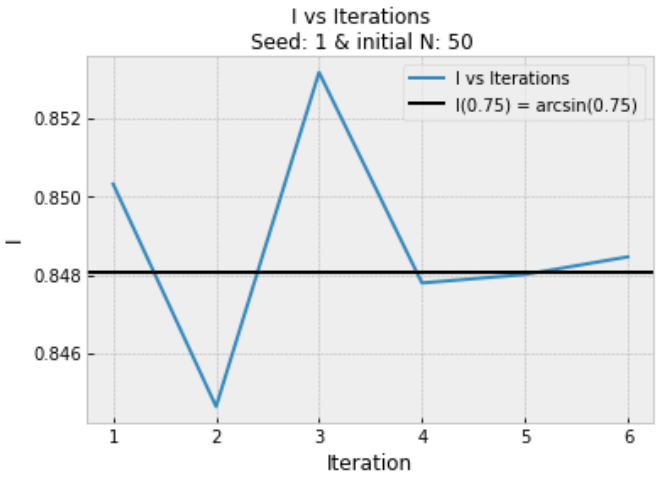

Figure 7: Monte Carlo Simulation to Solve $I(1.0) = \int_0^{1.0} \frac{dx}{\sqrt{1-x^2}}$ (15)

4 Conclusion

In this project, we used the Monte Carlo method to evaluate three types of integrals, 1-D, 3-D, and Improper. The Monte Carlo method proved to be extremely useful, allowing us to determine the accuracy and precision of the iterative approach. All integrals were evaluated with a percent error of less than 1%. This proves the usefulness of this method. For future projects, another useful plot would be N Number of Points (the second column of the tables) vs the value of the integrand following each iteration. This would showcase the way in which accuracy increases with sample size, and a relationship could be derived. Another option would be to plot N against the percent error, which would also provide a similar overview. Overall, the Monte Carlo method proves to be a method especially useful for 3-D and Improper Integrals with Integrabale Singularities.

References

[1] S.S.M Wong. *Computational Methods in Physics and Engineering.*