PH456 Lab 2 : Using Random Numbers

Sourabh Choudhary

April 19, 2022

The code is available on github via this link. All code is in lab-2.ipynb jupyter notebook.

1 Function to integrate

In the jupyter notebook the code for the implementation of monte carlo integration is given. The function uses uniform random numbers generated using numpy. Function returns the answer and the error of the given integral.

2 Calculating given definite integrals

Function written in task 1 was used to get solutions to the integrals. The same function was extended to 2D solve problem 2(d).

$$a)\int_0^1 2\ \mathrm{d}x = 2.0 \pm 7.105e-21 \qquad N = 50 \tag{1}$$

$$b)\int_0^1 -x\ \mathrm{d}x = -0.4997 \pm 3.335e-07 \qquad N = 50 \tag{2}$$

$$c)\int_{-2}^2 x^2\ \mathrm{d}x = 5.341 \pm 5.699e-06 \qquad N = 50 \tag{3}$$

$$d)\int_0^1\int_0^1 xy + xdxdy = 0.7486 \pm 9.902e-06 \qquad N = 500 \tag{4}$$

Figure 1 shows the solutions to all the equations above. It can be seen that for the linear equations the solutions converges within reliability within 50 counts, whereas for the non-linear case a lot more points are required, almost 10x in shown in the figure.

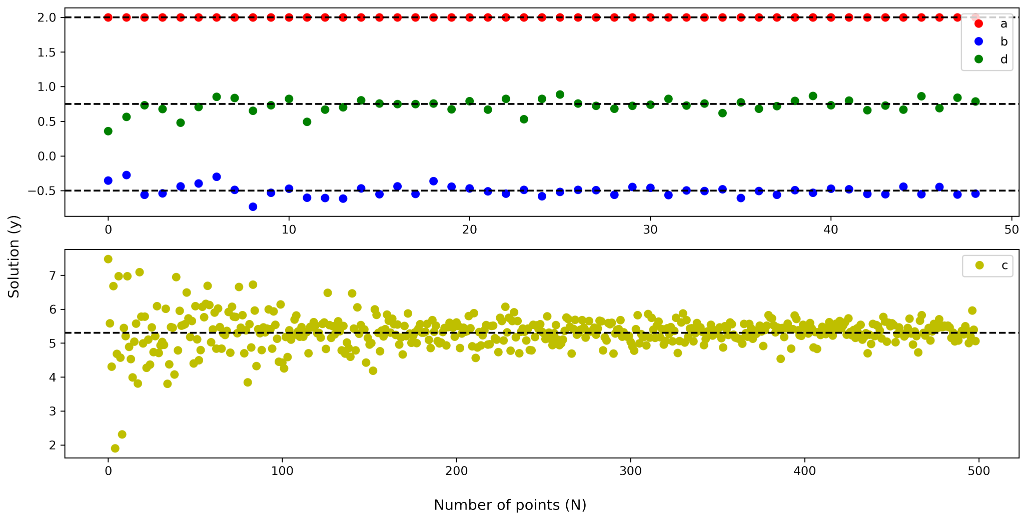

Figure 1: Solutions to Question 2, black lines are the analytical solutions and the dots are solutions for N number of steps on the monte-carlo integration function. Legend shows the which sub-question the solution points to.

3 Volume of an n-sphere

A different function with some similarities to the one in task 1 was written to calculate the volume of an n-sphere.

Radius	Dimensions	Volume	Error
2.0	3	33.44	
2.0	5	167.4	

4 9 D Integral

A jerry rigged attempt was made to calculate the 9D integral. It did not work, there definitely is an elegant and compact solution but I could'nt figure it out.

5 Importance sampling with Metropolis Algorithm

The Metropolis algorithm did not work as required. It was able to generate random numbers as per given probability distribution shown in the figure below. But when trying to use it to find the solution the answer was always wrong. There are bugs in the implementation and also a lack of understanding on my part on how to utilise the generated sample to effectively calculate the solutions. However, the implement does seem to work with simple linear functions like f (x) = 2x or f (x) = x +7.

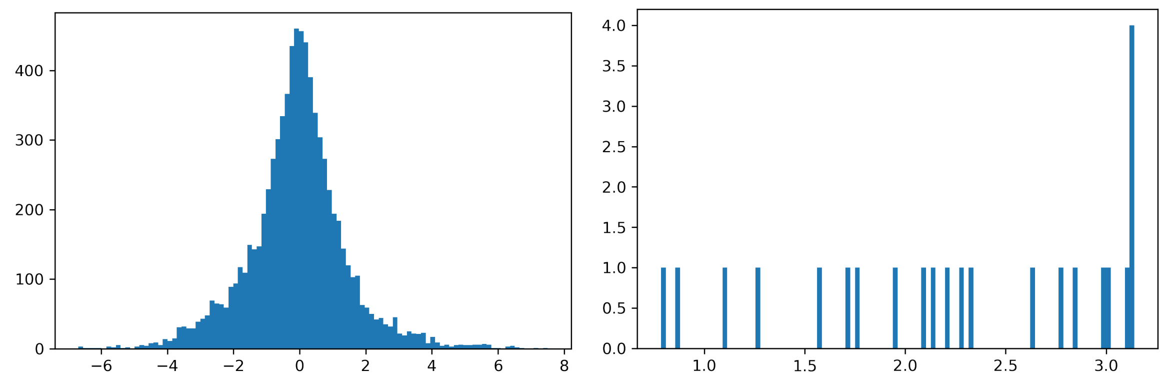

Figure 2: Probability distribution of the implementation. The first one for $e^{-|x|}$ looks alright. The other one, not so much. I used $0.1a, 0.1b$ for δ that worked alright for the $e^{-|x|}$ but not well for $sin(x)$.

6 Uniform sampling comparison with Metropolis Algorithm

Nothing to compare with cause the implementation does not work well with the functions in the question. The following plot shows the convergence of the uniform sampling method.

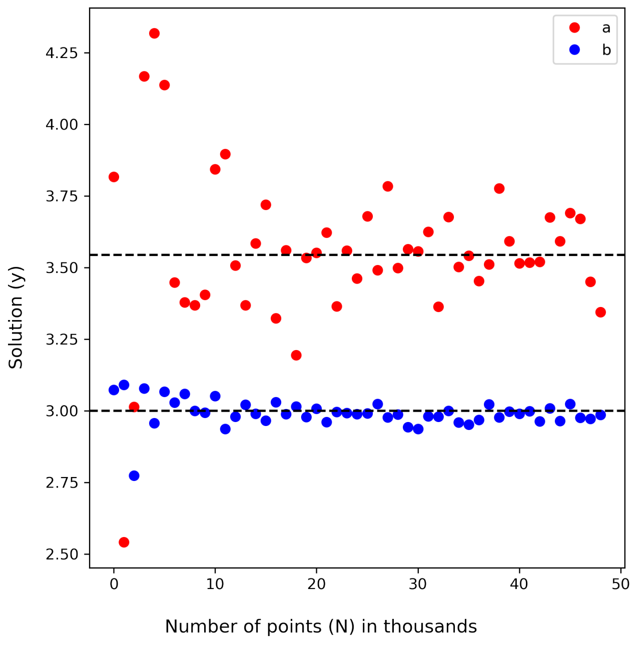

Figure 3: Figure on left shows the convergence of the functions
a) RED : $\int_{-10}^{+10} 2e^{-x^2}dx = 3.5360$,

$error = 2.2007e - 05 for N = 50000$

b) BLUE : $\int_0^\pi 1.5 sin(x) dx = 2.9995$

$error = 2.158e - 05 for N = 50000$

for number of counts N in thousands on x-axis. Function (a) does not converge well and would require a lot more points.