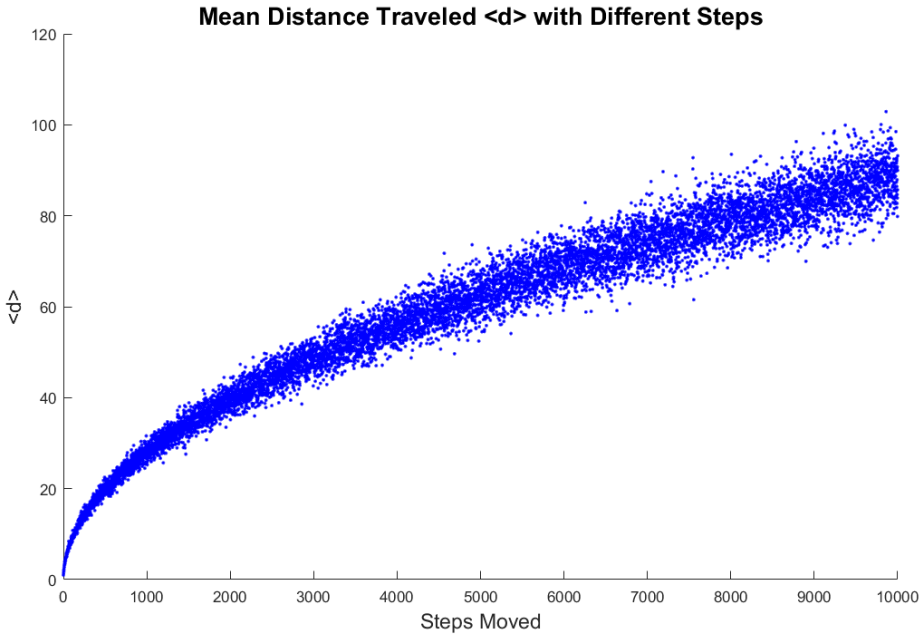
Problem Set 3 **R08244002 Shin-Rong, Tsai**

- **Random walk on a two-dimensional square lattice**

1. Results

Run code *random_walk.c*, then plot the result with MATLAB.



2. Discussion

At first, I thought that I can also simulate this result with a Pascal-triangle like plane, which it expands itself in four directions for a step. Then calculate the probability and the expectation value of the distance traveled (with Python36 in *layer.py* file). But it took the computer lots of time, and while I'm writing this, it has not done the job yet. It only finished from 1 to 15 steps moved, but we can see that these results match the one done by Monte Carlo method.

After looking up in wiki, I take another approach to analysis this. We can view this 2D lattice random walk as two 1D. Since the distance we are calculating matters only with the largest horizontal and vertical distances it travels, we only have to know the expectation value of the distance it travels in 1D case.

In 1D case, we want to know the expectation value of distance:

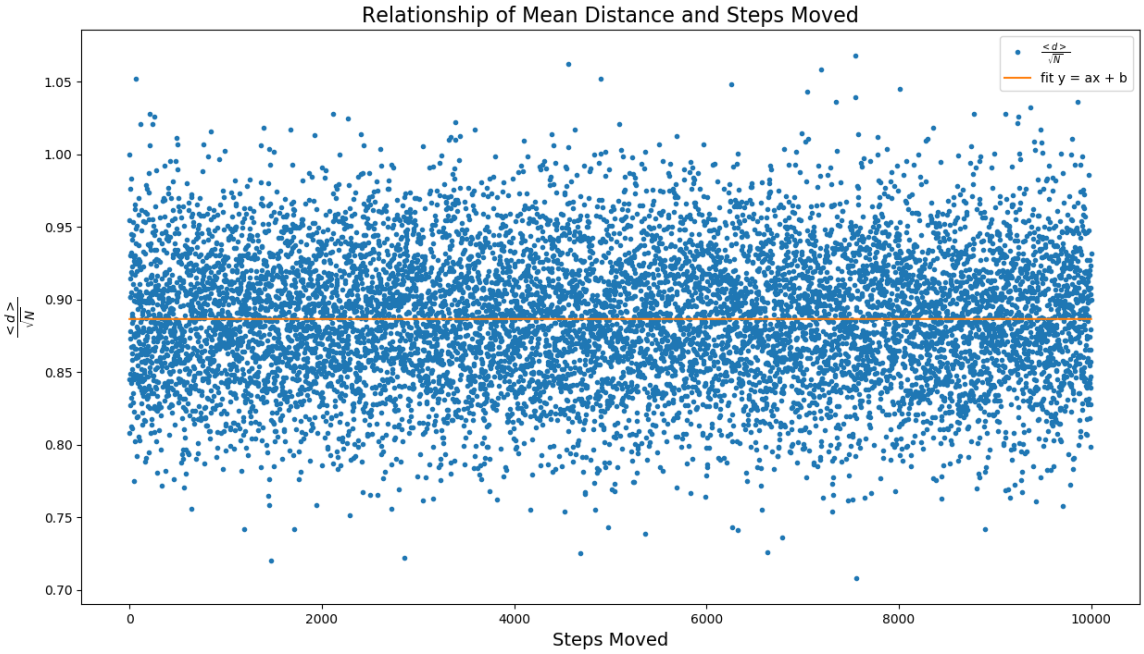
$S_n = \sum_{j=1}^{n} Z_j, \quad Z_j = \{+1, -1\}.$

$$E(S_n^2) = \sum_{i=1}^{n} E(Z_i^2) + 2 \cdot \sum_{1 \le i,j \le n} E(Z_i Z_j)$$

$$= n + 2 \cdot [\frac{1}{4} \cdot (+1)^2 + \frac{1}{2} \cdot (+1)(-1) + \frac{1}{4} \cdot (-1)^2] = n$$

$E(|S_n|) \propto \sqrt{n}.$

So after running *fit.py*, I get this result:

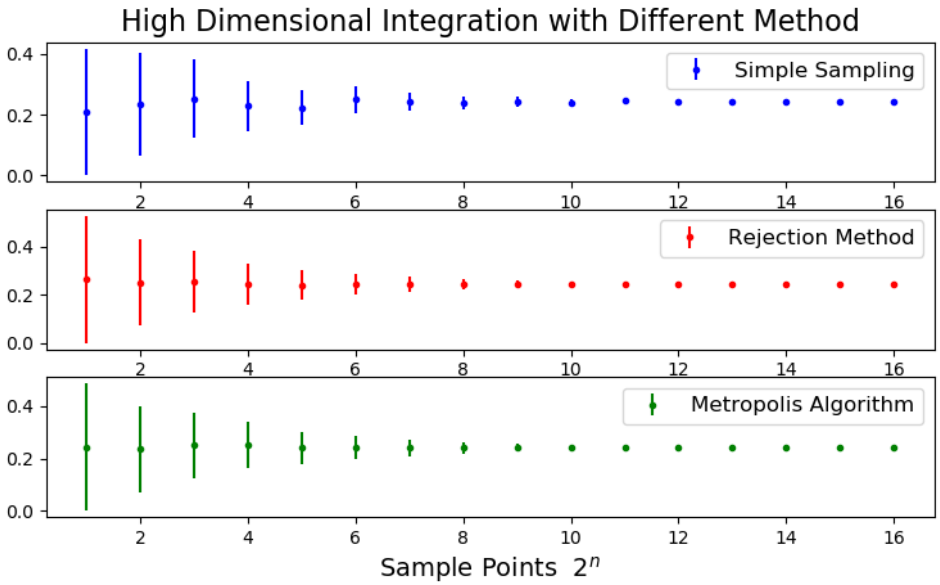


Where $a = 1.04810971 \times 10^{-9}$ and $b = 8.86492221 \times 10^{-1}$. It is a horizontal line as expected.

- **Monte Carlo integration in 10 dimension**

  1. Results

  Run *monte_carlo_integration.c*, and plot with *plot_integration_result.py* with Python36, we get:



2. Discussion

From the figure, we can see that both Rejection Method and Metropolis Algorithm converge faster. And we can check the central limit theorem by dividing each of their error with $\sqrt{2}$, then we will approximately get the error of the next row below.

	N	SS	SSerror	RM	RMerror	MA	MAerror
0	2	0.208967	0.209095	0.263295	0.263296	0.243014	0.243020
1	4	0.234802	0.169186	0.250793	0.177382	0.235138	0.166544
2	8	0.252875	0.129217	0.253354	0.127334	0.249706	0.124958
3	16	0.229098	0.081990	0.241937	0.085734	0.250688	0.088792
4	32	0.222360	0.056322	0.240989	0.060326	0.240124	0.060147
5	64	0.249713	0.044697	0.242020	0.042872	0.241591	0.042781
6	128	0.241960	0.030595	0.243201	0.030469	0.239887	0.030026
7	256	0.238907	0.021344	0.245512	0.021744	0.240745	0.021313
8	512	0.242480	0.015360	0.245783	0.015400	0.242942	0.015208
9	1024	0.239744	0.010732	0.244418	0.010826	0.243305	0.010776
10	2048	0.245065	0.007769	0.245468	0.007688	0.242152	0.007581
11	4096	0.242973	0.005448	0.244560	0.005416	0.242551	0.005370
12	8192	0.243104	0.003855	0.244962	0.003836	0.242930	0.003803
13	16384	0.243387	0.002728	0.244616	0.002708	0.243115	0.002691
14	32768	0.242666	0.001923	0.244655	0.001915	0.242912	0.001901
15	65536	0.242781	0.001361	0.244692	0.001355	0.242691	0.001343

I think the hardest part of this problem set is finding the weight function for 10-dimensional integral. Since in one variable function, the weight function (g(x)) and f(x) looks like this, it is likely that we may assume the weight function w(x) will be $w(x) = 2 - m \cdot \sum_{i=1}^{10} x_i$ with constant m acts as slope.
Because Rejection method and Metropolis Algorithm are both importance sampling, and the first one is irrelevant to w(x) and the second one utilize it when generating random number. And since Rejection method will somewhat reflect the distribution of f(x), maybe we can adjust m till both method converges or the difference reach the smallest at $N = 2^{16}$. This is just my guess, please correct me if my idea is wrong.

So eventually I get $w(x) = 2 - \frac{1}{5}\sum_{i=1}^{10} x_i$. And the integral I might be 0.243 I guess.