Test of a Random Number Generator Using Random Walks

Probability Distributions

After collecting the data for a different number of walks it is no possible to graph the resultant probability distributions for bits and compare them to the theoretical ones. The simulations were run with a number of walks that are powers of ten starting from 10^2 and up to 10^7. Below the results from the run with 10^7 walks are analyzed as they provide the most data points for the analysis.

Below are the graphs comparing theoretical probability distributions with the measured ones. There are 5 different distributions the graphs of which are shown below which correspond to the distributions of the 1st, 8th, 12th, 16th, and 64th bits. These 5 bits were picked to compare the behavior of different bits in this random number generator.

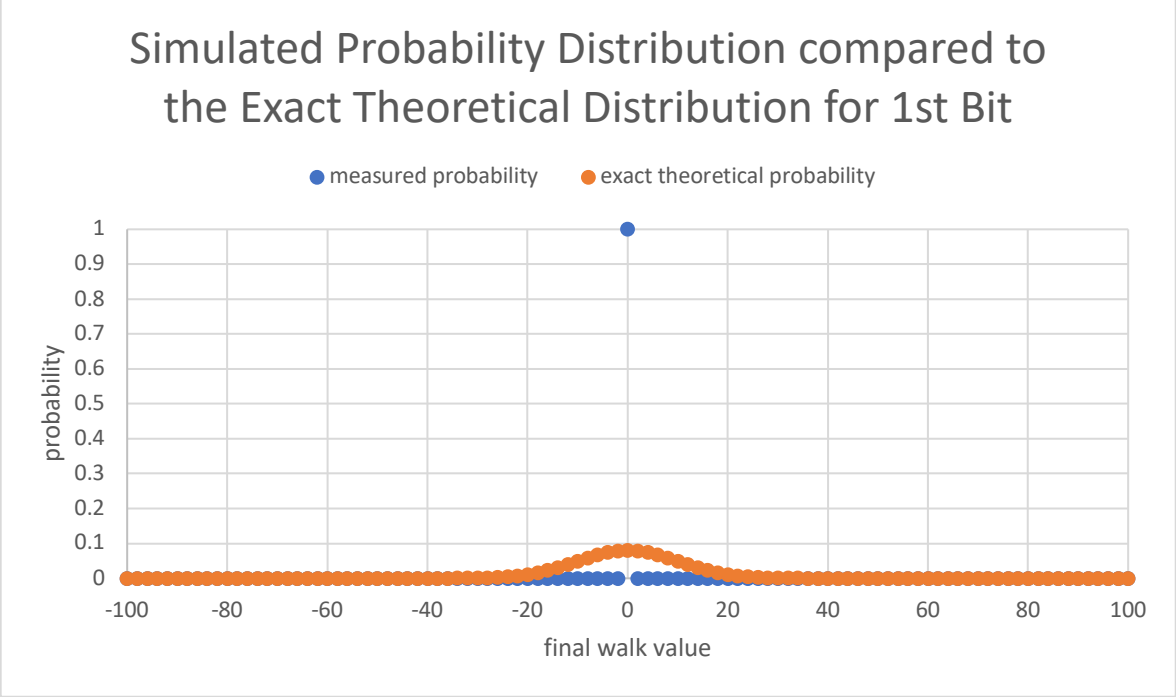

Figure 1: Simulated Probability Distribution compared to the Exact Theoretical Distribution for 1st Bit on a linear scale

Starting with the 1st bit, it is clear that the distribution does not follow the expected behavior. All the walks led to the very same result, causing the probability of the value 0 to be 100%, and the rest ones are 0%. This can be seen from the graph above.

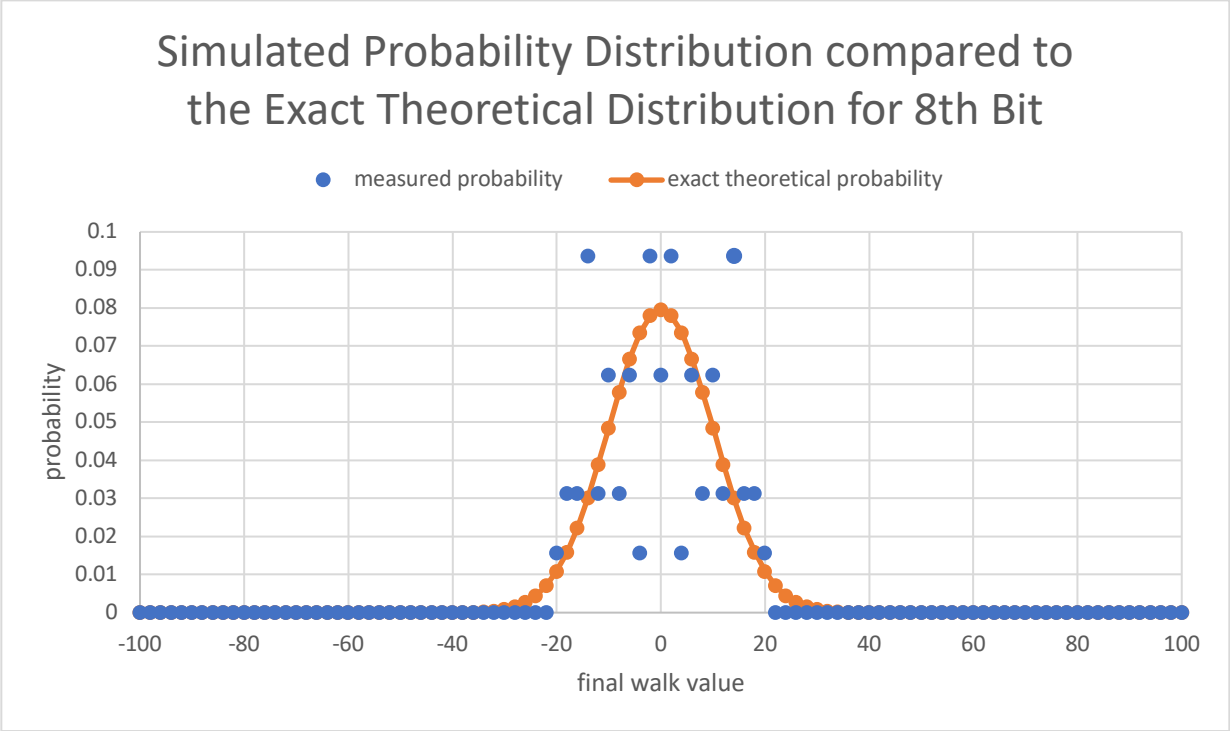

Figure 2: Simulated Probability Distribution compared to the Exact Theoretical Distribution for 8th Bit on a linear scale

Looking at the graph for the 8th bit above it is clear that there is some progress towards moving closer to the theoretical distribution. Even though there is still a small, fixed number of possible outcomes, there are now many more of those. However, those still form a pattern different from the one formed by the theoretical distribution with a number of unnatural peaks.

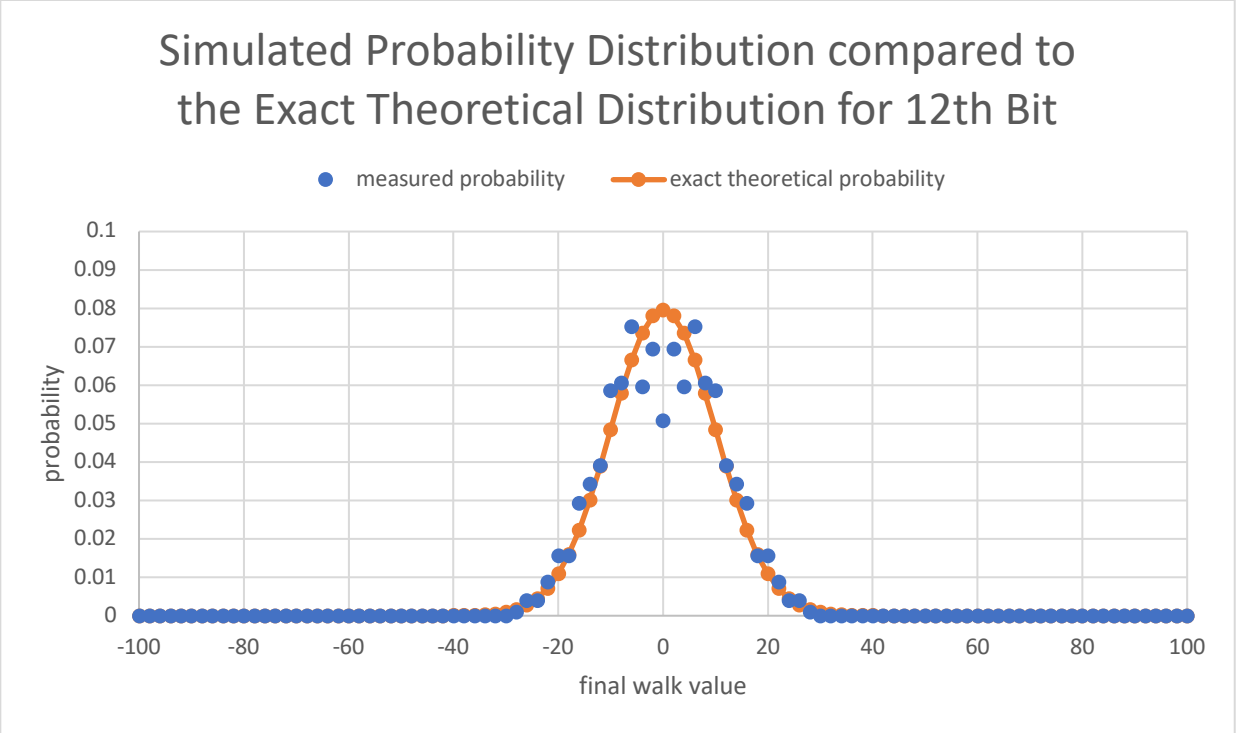

Figure 3: Simulated Probability Distribution compared to the Exact Theoretical Distribution for 12th Bit on a linear scale

Consequently, the graph above yields that increase the bit number moves the measured distribution closer to the expected one. Most of the points are now close to the theoretical distribution, except for the one corresponding to the values in the middle.

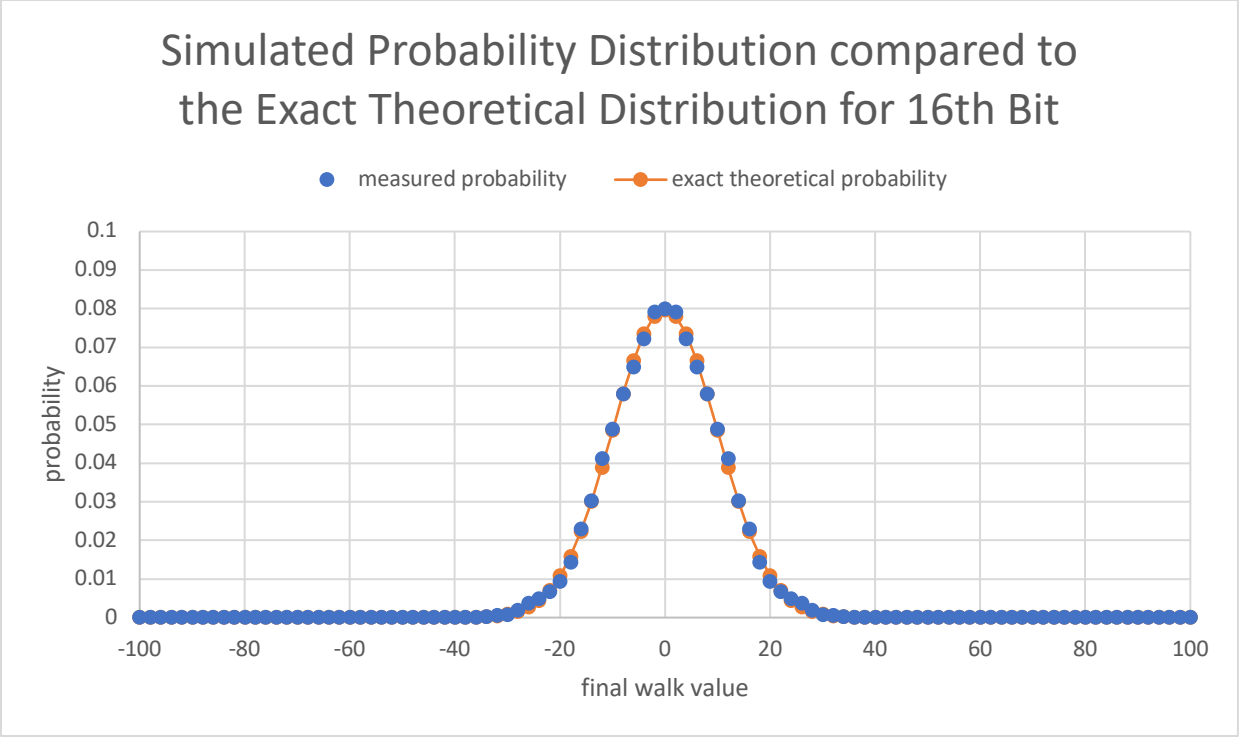

Figure 4: Simulated Probability Distribution compared to the Exact Theoretical Distribution for 16th Bit on a linear scale

The graph above shows the distribution for the 16th bit and it can be claimed that distributions almost completely overlap, making the measured distribution extremely close to the theoretical one.

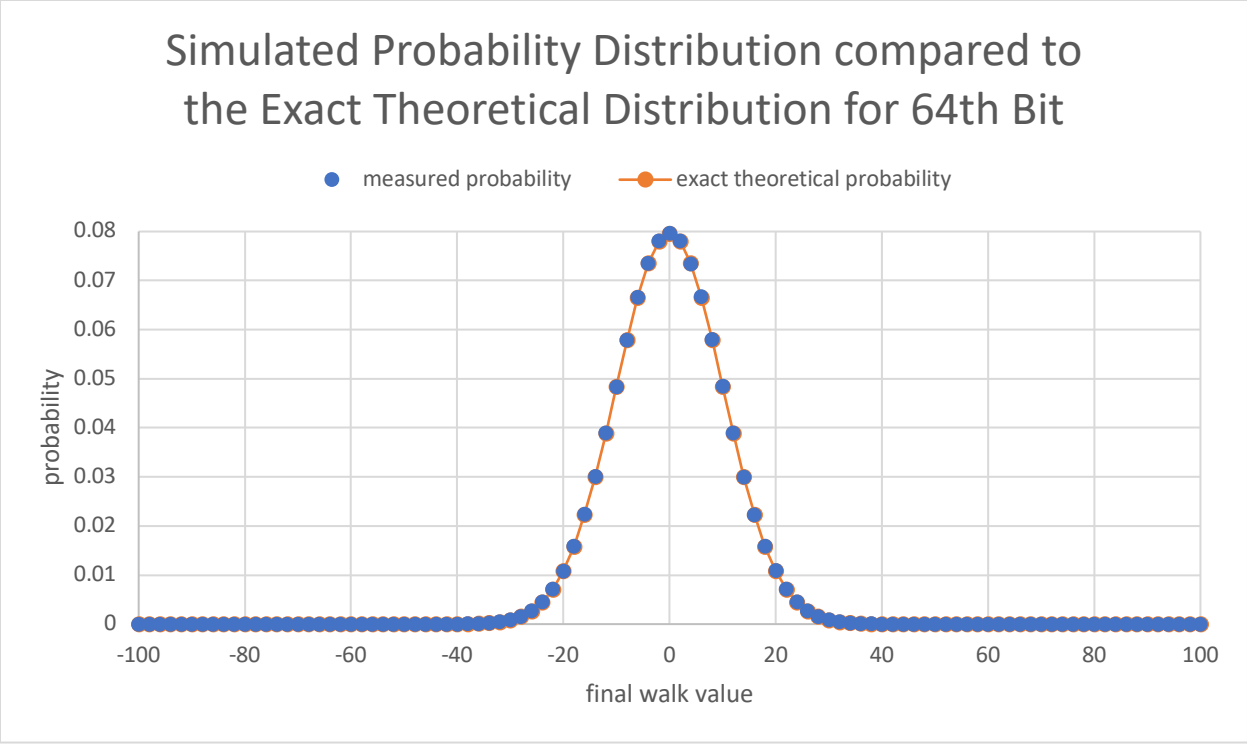

Figure 5: Simulated Probability Distribution compared to the Exact Theoretical Distribution for 64th Bit on a linear scale

As can be seen from the graph above, the distribution for the 64th bit seems to repeat the theoretical one, making it look like an accurate representation of a truly random walk.

Analyzing these graphs, it is reasonable to claim that the greater the bit number, the closer the distribution to the theoretical one, the closer the results of the bit to representing a truly random walk.

In addition to this graphs, it is reasonable to plot the very same ones but with a logarithmic probability scale. Due to a logarithmic scale it is easier to observe smaller probabilities and, therefore, see how close the data points to theoretical distributions. Adjusted graphs for all 5 bits are presented below.

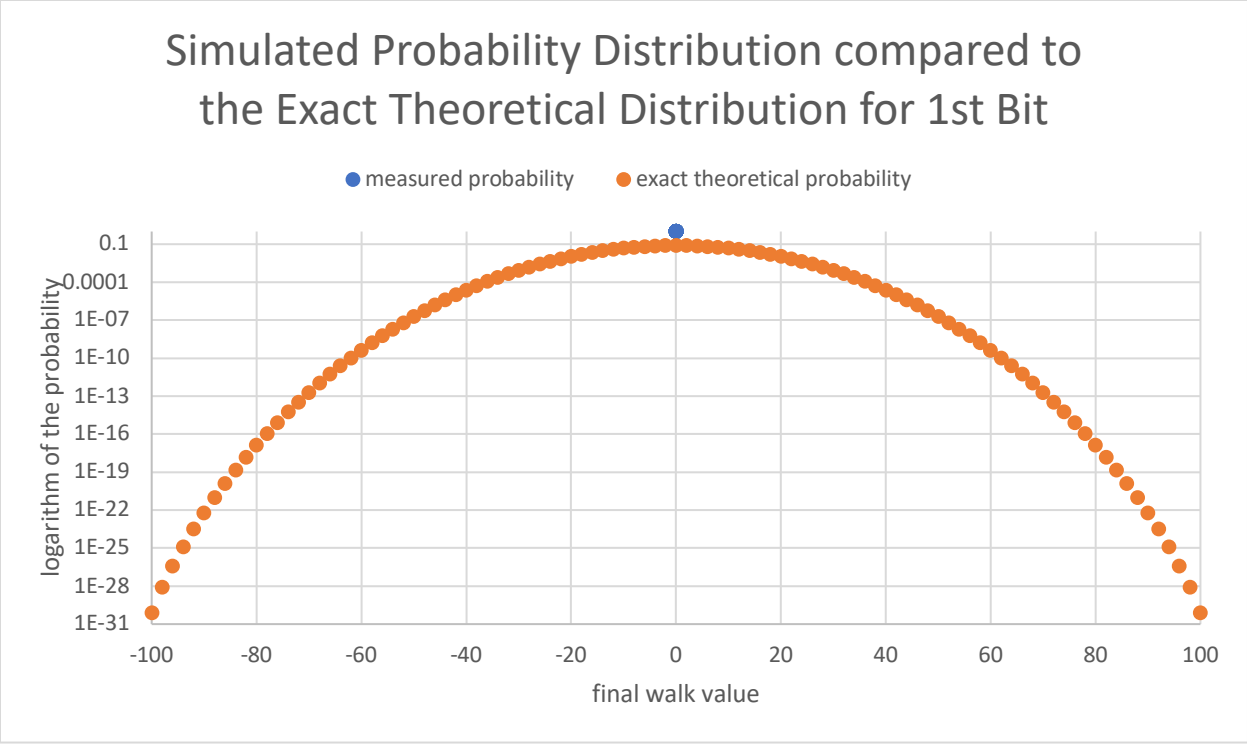

Figure 6: Simulated Probability Distribution compared to the Exact Theoretical Distribution for 1st Bit on a logarithmic scale

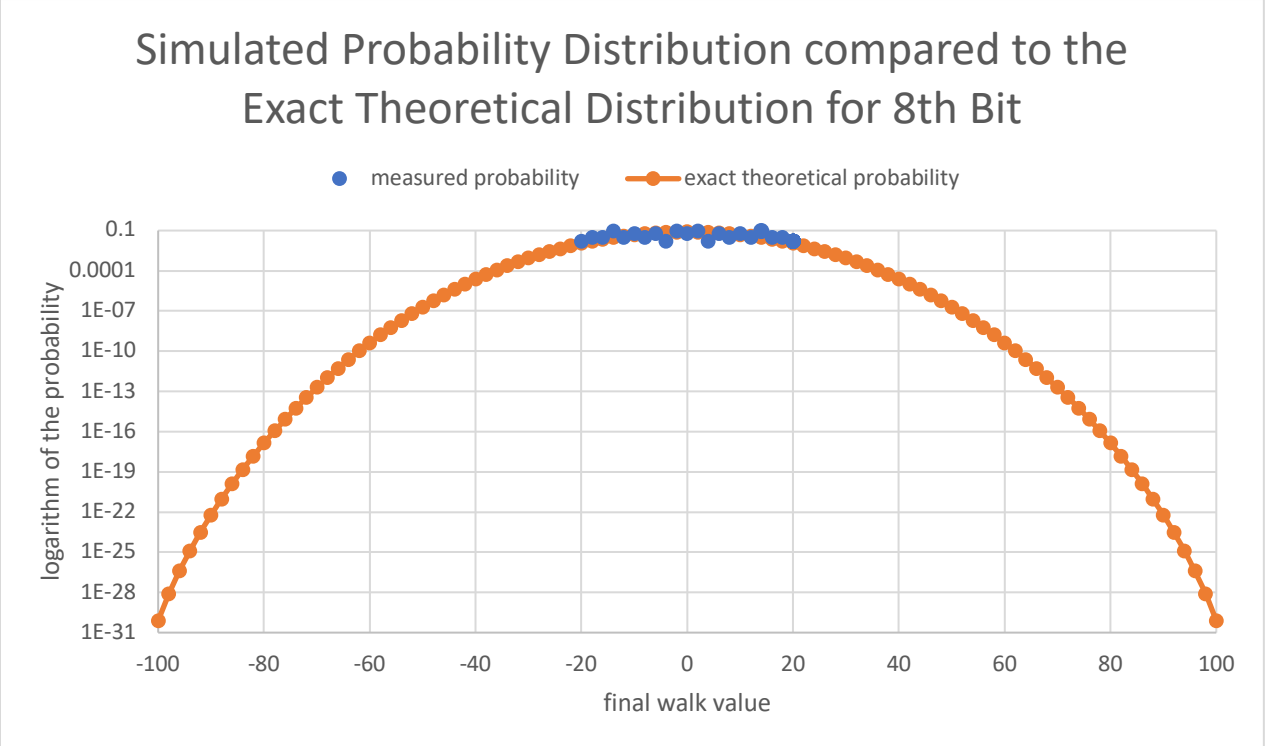

Figure 7: Simulated Probability Distribution compared to the Exact Theoretical Distribution for 8th Bit on a logarithmic scale

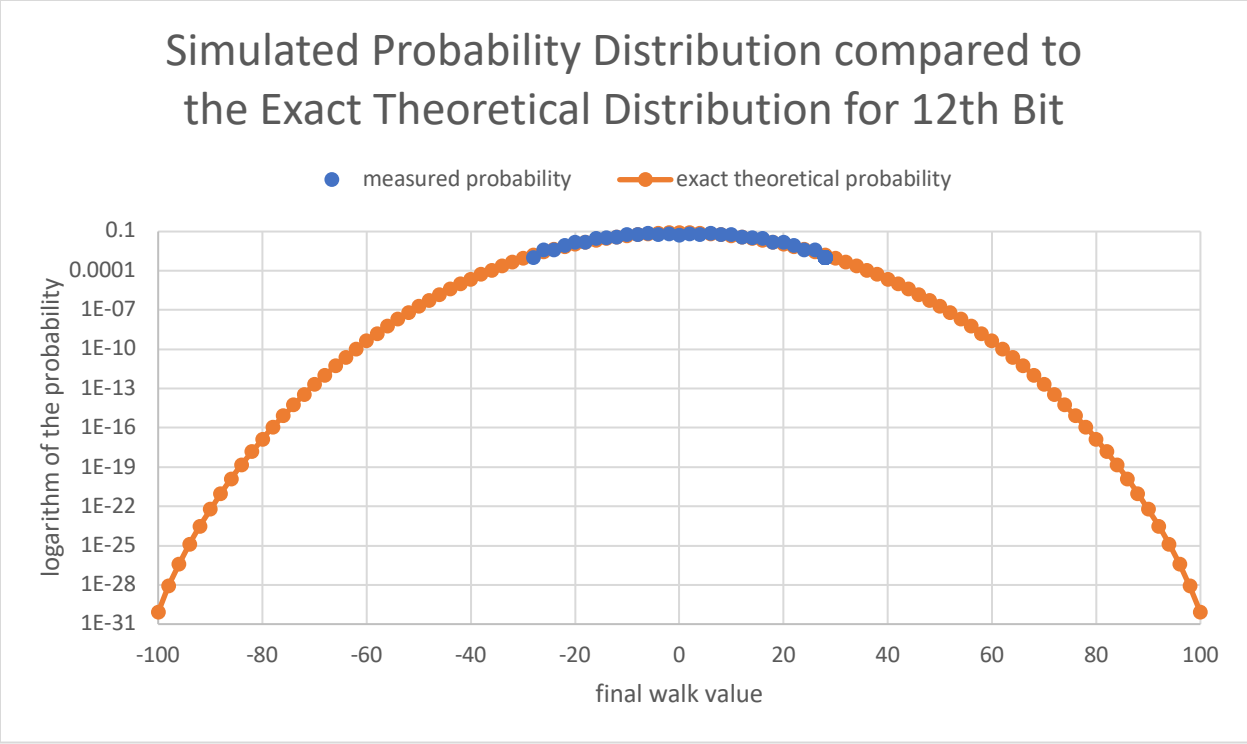

Figure 8: Simulated Probability Distribution compared to the Exact Theoretical Distribution for 12th Bit on a logarithmic scale

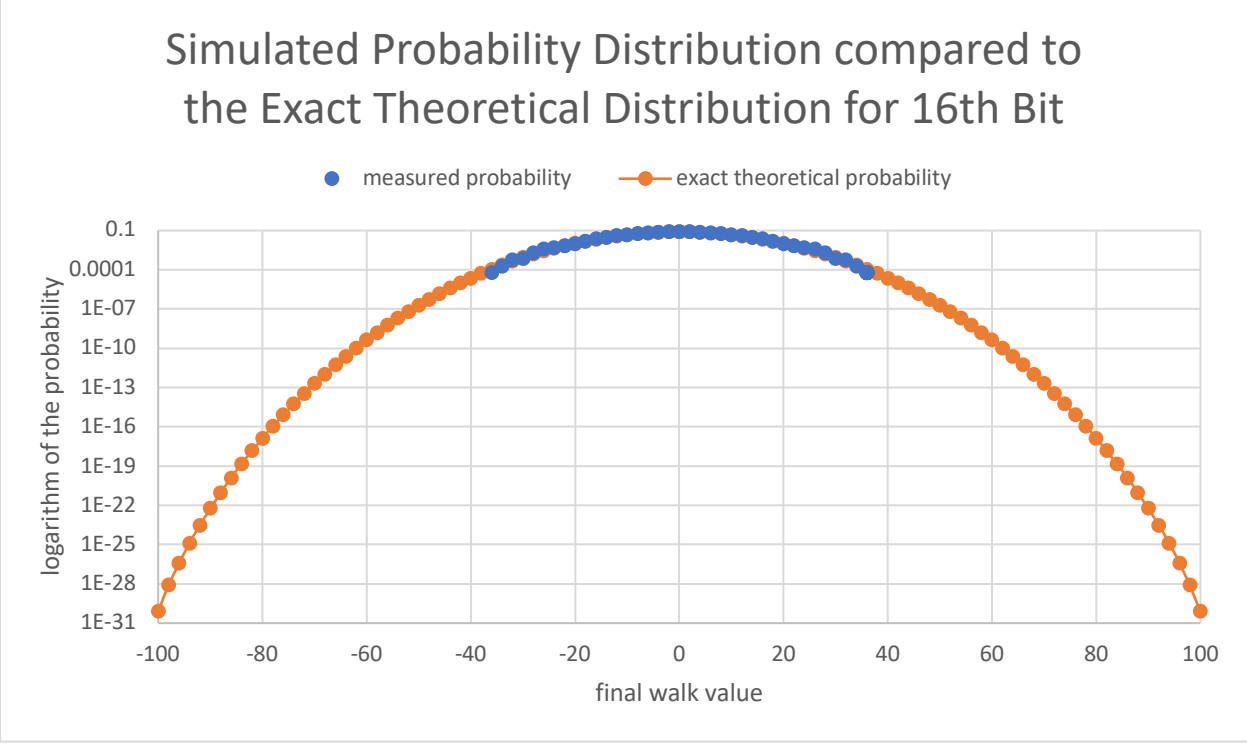

Figure 9: Simulated Probability Distribution compared to the Exact Theoretical Distribution for 16th Bit on a logarithmic scale

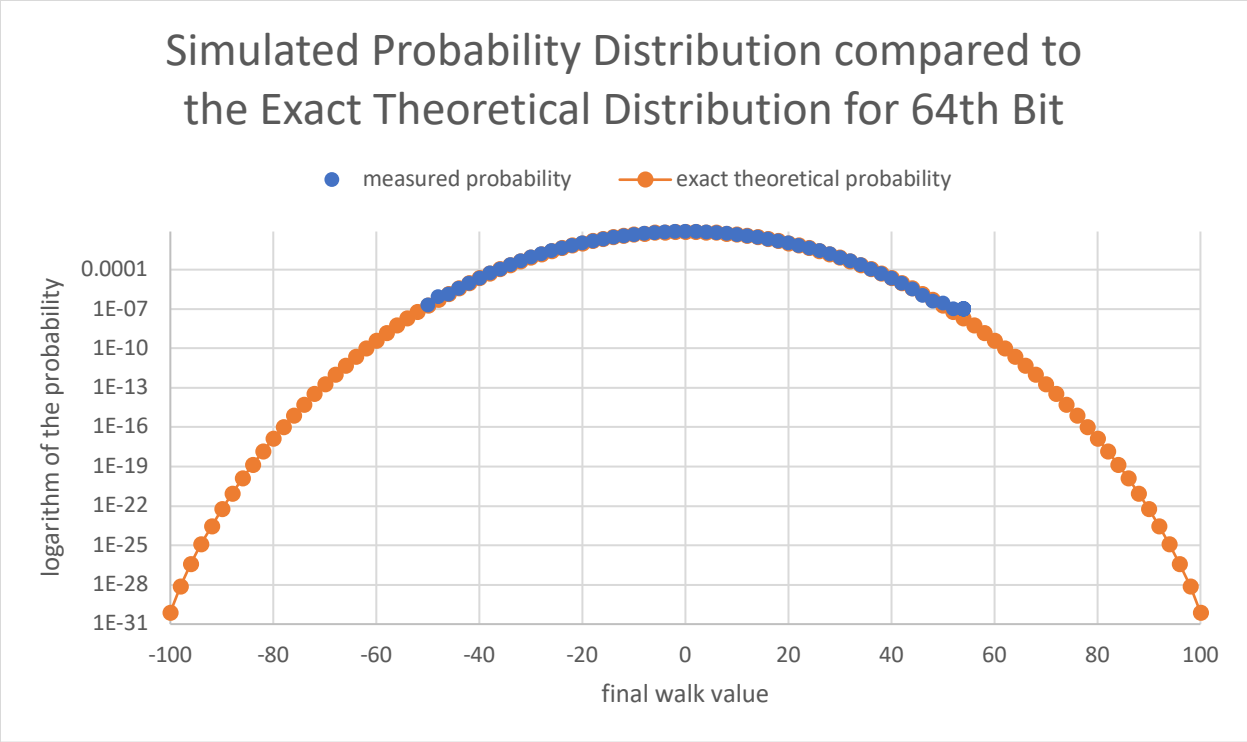

Figure 10: Simulated Probability Distribution compared to the Exact Theoretical Distribution for 64th Bit on a logarithmic scale

Looking at these graphs in the order of increasing bit number one can claim that the "blue arch" is growing with growing bit number. The reason for that is that the width of the arch represent the range of the values obtained for the final bit. In fact, that means that the greater the bit number, the greater the range, therefore, the greater the number and magnitude of the deviations from the expected final value of zero (as due to equal probability of going up and going down the expected final value is zero). It confirms the claim that bits with greater number perform better in terms of simulating a truly random walk.

Deviations

Another helpful metric to analyze is the deviation. Simulations calculated deviations Δ corresponding to single bits in each set of walks and produced files with the product $\sqrt{N_w}\Delta$ (where N_w is the number of walks) for all bits. This metric is helpful in terms of analyzing how

similar is the behavior of the random generator to that of the theoretical random number generator that follows the laws of probability. In theory, deviations should be proportional to the inverse of the square root of the number of walks, meaning that with enough walks the product should approach a value of 1. Below are the graphs for weighted deviations of bit values for different number of walks.

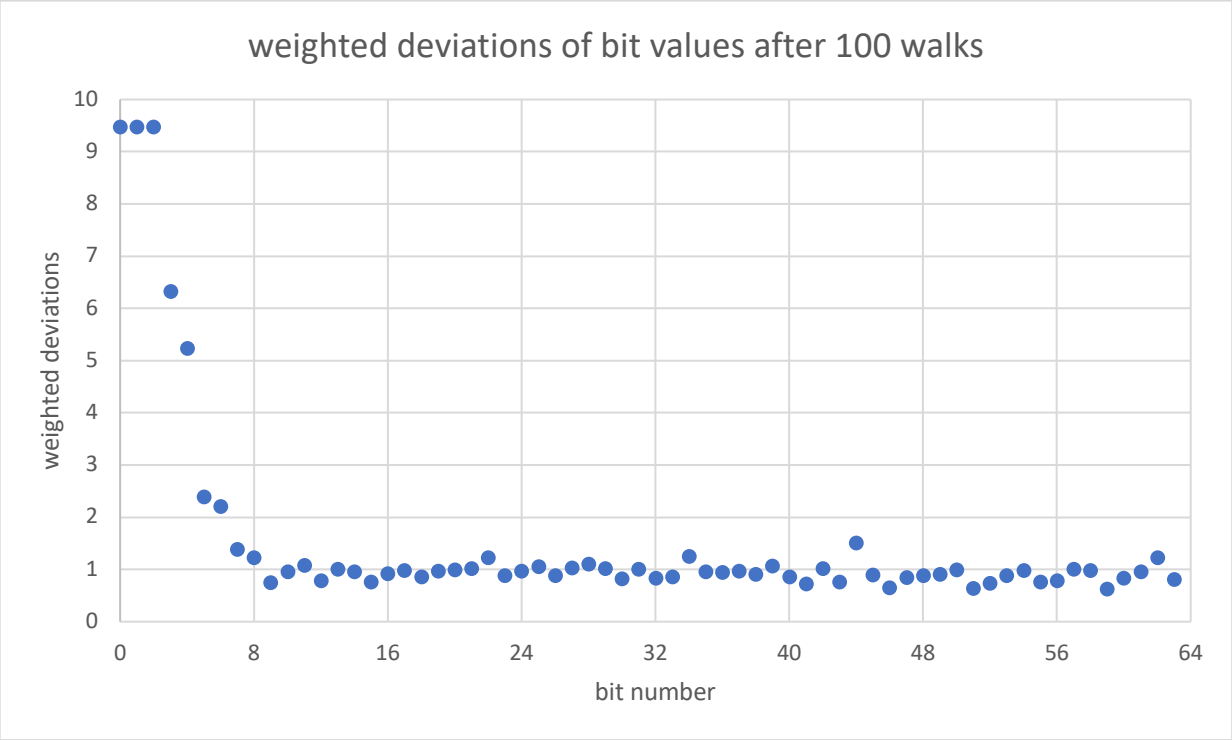

Figure 11: weighted deviations of bit values after 100 walks

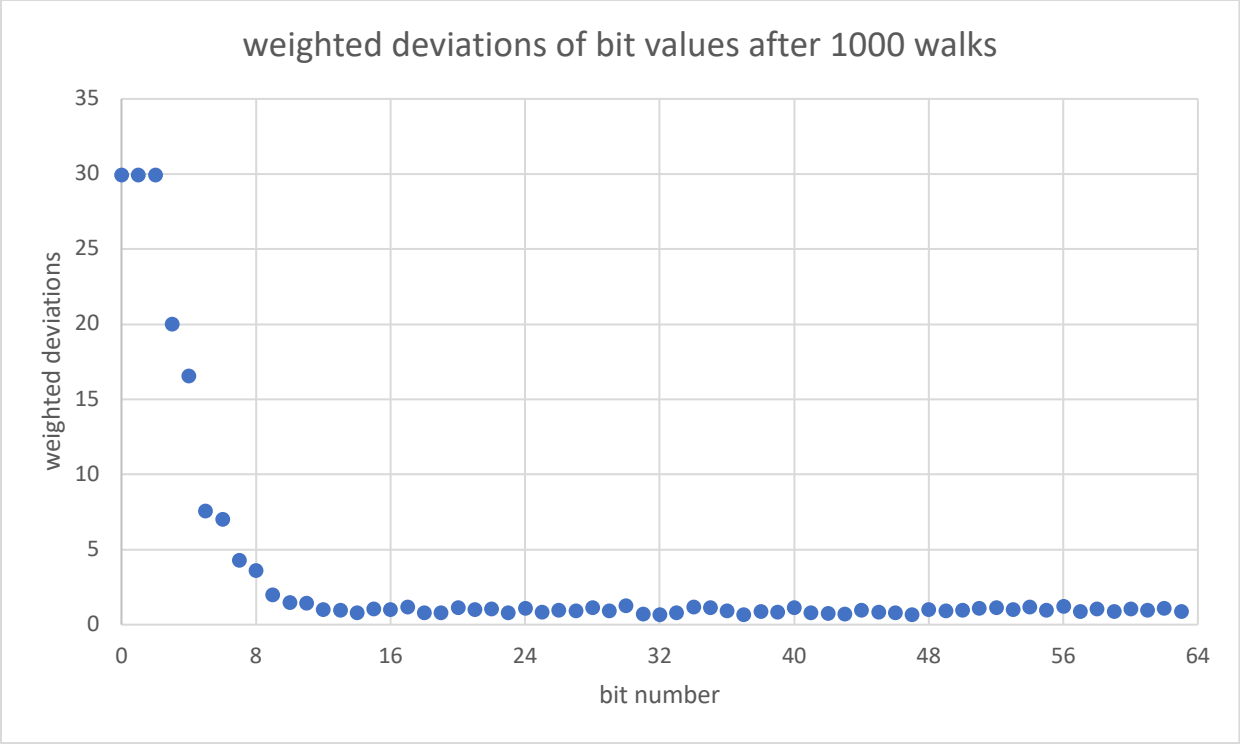

Figure 12: weighted deviations of bit values after 1000 walks

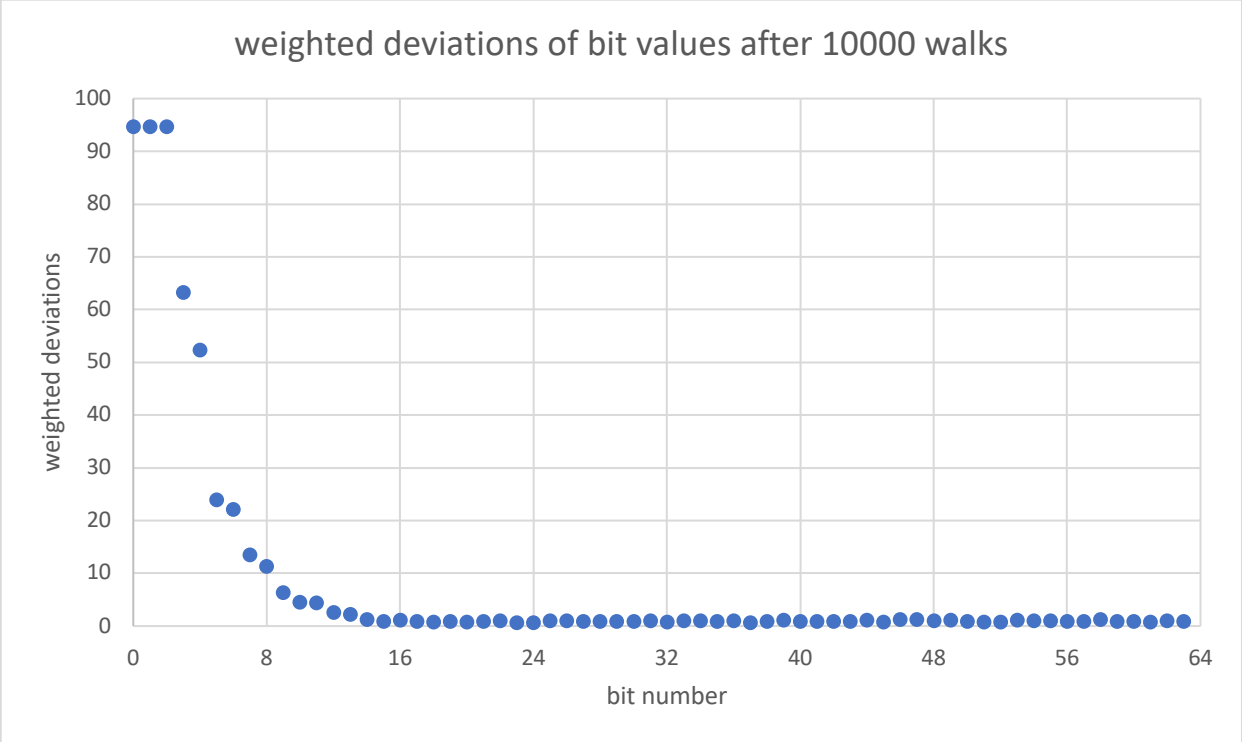

Figure 13: weighted deviations of bit values after 10000 walks

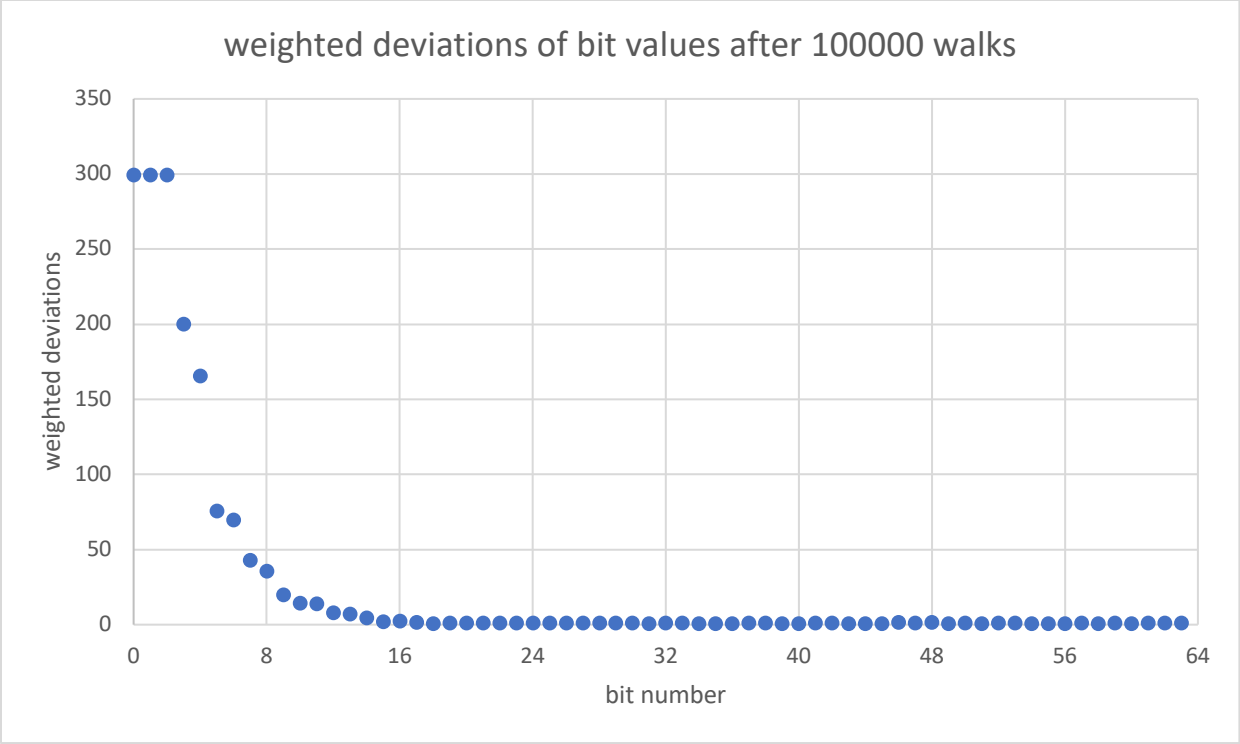

Figure 14: weighted deviations of bit values after 100000 walks

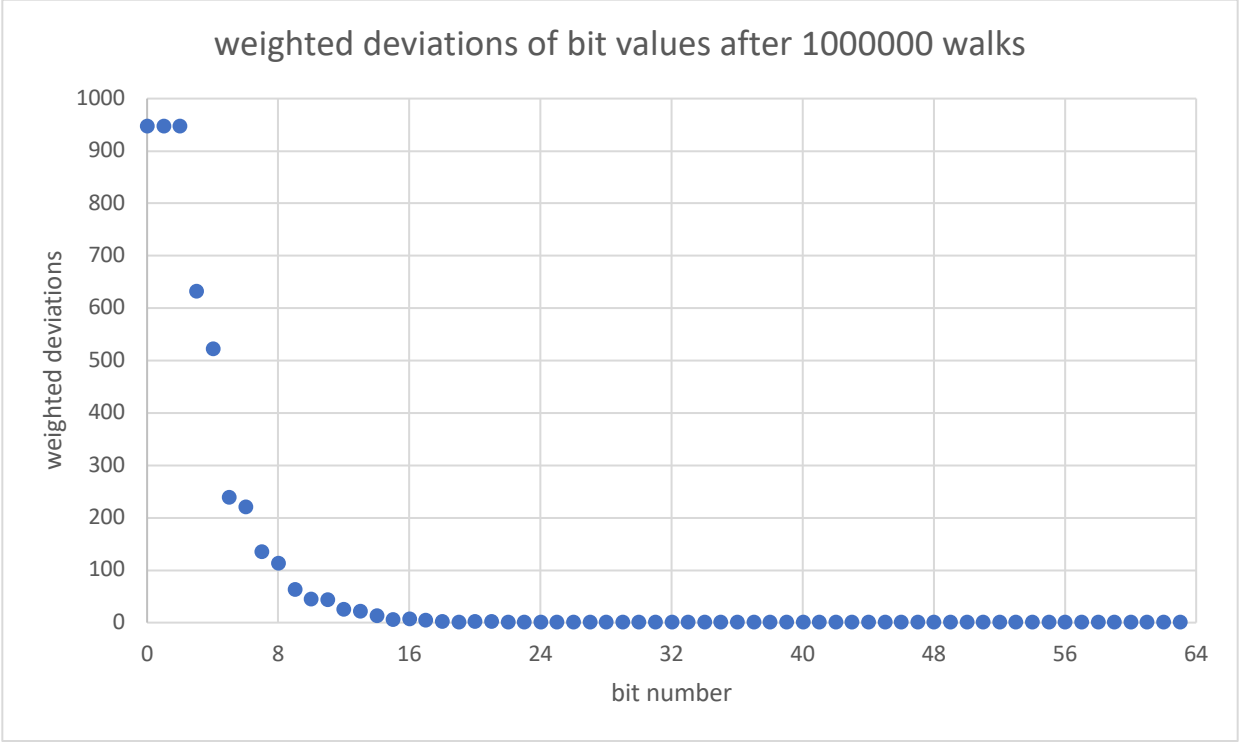

Figure 15: weighted deviations of bit values after 1000000 walks

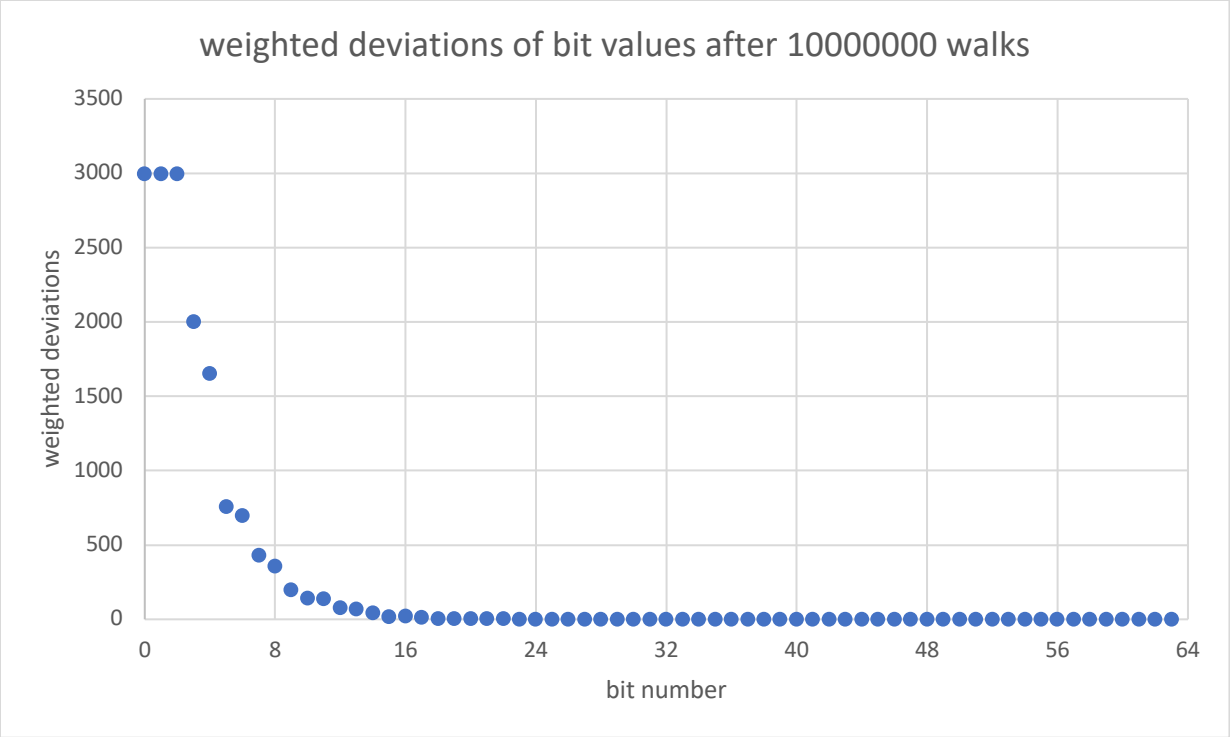

Figure 16: weighted deviations of bit values after 10000000 walks

All these graphs seem to represent a similar pattern and according to the actual values, they all demonstrate values of weighted deviations of approximately 1 for all bits starting from the 16th one. However, weighted deviations of values of bits from 0th to 15th (approximately) seem to grow with the growth of the number of walks, meaning that their deviations grow faster than the square root of the number of walks. This yields a concern with regards to the performance of these bits as they seem to poorly represent a truly random process. That is also confirmed by the previous part of the report where 1st, 8th, and 12th bits were not close enough to the theoretical distribution.

Evaluation

It is clear from the previous analysis that the bits with smaller numbers perform worse than the ones with larger numbers. The first bit, for example, always produces an equal number

of upward moves and downward moves which causes its final value to be the same and not follow the distribution. The reason for that seems to be the mechanics of the linear congruential generator. The formula for the generator is presented below.

$$r_{k+1} = a * r_k + c$$

Where a, c are constants, r_k is the previous random number and r_{k+1} is the new one. In the code the values of the constants were the following.

$$a = 2862933555777941757$$

$$c = 1013904243$$

Due to the parity of these values whenever r_k is even, product $a * r_k$ is even, and $a * r_k + c$ is odd. For odd value of r_k we get odd $a * r_k$, and, therefore, even $a * r_k + c$. It means that numbers produced by this generator will be alternating between even and odd. Since the first (form the end) bit of the number is exactly the bit responsible for divisibility by 2, we know that the value of this bit will be alternating between 0 and 1, which, in turn, means that the sequence of steps will be alternating between upward and downward moves. It's important to mention that the same chain of reasoning can be applied to other possible coefficients, making it the flaw of the linear congruential generator. Additionally, similar chains of reasoning can be used to explain poor ranges of final values corresponding to other bits with smaller numbers, as these bits are partially responsible for divisibility by other smaller powers of 2, which does not provide enough variety. As the bit number increases, it becomes less dependent on such "alternating" patterns, making it change in a less predictable order. This makes the distributions for such bits look much closer to the theoretical ones. To conclude, the results received are caused by the mechanics of the

linear congruential generator, making bits with smaller numbers perform “less random” than the ones with larger numbers.

Conclusion

The initial purpose of analyzing the behavior of the bits is to see whether such a generator can be used to produce random floating-point numbers by linearly converting integers between 0 and $2^{63} - 1$ to the range $[0, 1)$. We noticed that higher order bits perform (approximately the ones starting with the 16th one) perform relatively well in terms of simulating randomness. These bits will make larger numbers more random and since large numbers is exactly what is generated, the results should be relatively good in terms of randomness. In fact, consequent linear conversion to floating point number in the range $[0, 1)$ will make higher order bits responsible for figures of bigger magnitude (closer to the point), making the final random floating-point numbers (especially with proper rounding) seem random. Therefore, such an approach when lower order bits are not properly randomized but also affect the result less than the higher order bits that affect the result more seems to be reasonable and efficient, making linear congruential generator relatively good at generating random floating-point numbers in the range $[0, 1)$.