Note

For this problem set I started using jupyter notebooks and I have explained my thought process more than the previous problem sets. I have also included some of the calculations using equations in markdown in those files so this report will be less comprehensive than the previous ones and will act as more of an overview of what has been done.

6.1 Random Variable Distribution Function

randomGen() function

This function takes a number as input, then generates N random integers with values from 0 to 9 and outputs them in an array. We can test the evenness of this generator's distribution by plotting a histogram like below:

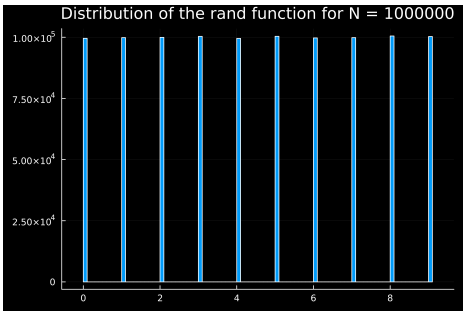

Figure 1: Distribution of the randomGen() random generator.

distributionCount() function

This function takes the randomly generated array as input, then counts the number of times each element has been generated and returns a 10 element array containing the number of times each of the numbers have appeared.

deviation() function

This function takes the distributions array and the number of generated numbers as input, then calculates $\frac{\sigma}{N}$ where σ is the deviation from the theoretically perfect distribution where each number is generated $\frac{N}{10}$ times.

Main Calculation Cell

The code in this cell is used to generate a number of data points, each average over a number of runs. This code generates an array for the values of $\frac{\sigma}{N}$ and $\frac{1}{\sqrt{N}}$, so when we plot these values on a linear graph, the slope will be the constant (If the function turns out to be linear which it is). We fit the slope to the data using the Polynomials.jl package. We then use the constant we just calculated to once more show the correlation between $\frac{\sigma}{N}$ and $\frac{1}{\sqrt{N}}$.

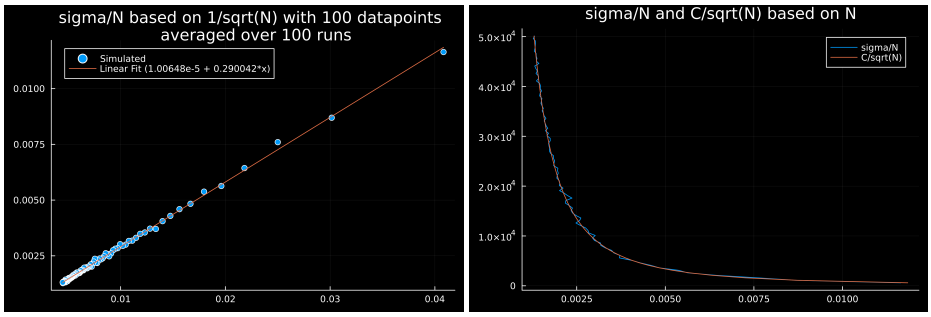

Figure 2: Results for the correlation between $\frac{\sigma}{N}$ and $\frac{1}{\sqrt{N}}$.

This problem is really similar to the Random Ballistic Deposition problem since in both problems a random number is generated and the frequency of each random number being generated is plotted.

6.2 Random Correlation

In this section we test if there are correlations between each generated random number and the next one in the built in random generator.

conditionalRandom() function

This function takes a number as input, then generates N numbers, each of which are numbers that are generated right after a 4 has appeared. The results are outputted in an array. The distribution of this generator can be plotted in the same manner as the previous problem like below: As we can clearly see, there

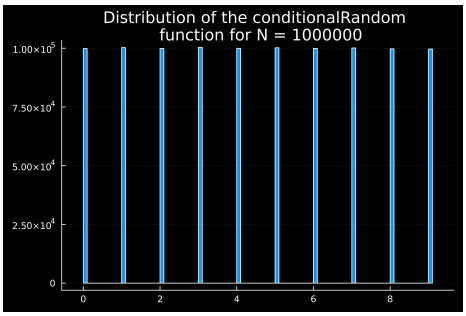

Figure 3: Distribution of the conditionalRandom() random generator.

doesn't seem to be much of a correlation between the random numbers so the built in generator is good enough for now.

distributionCount() and deviation() function and Calculations

the distributionCount() and deviation() functions are the same as the last problem, and the rest of the calculations and plottings are done in the same manner

as the last problem. The resulting graphs are as follows

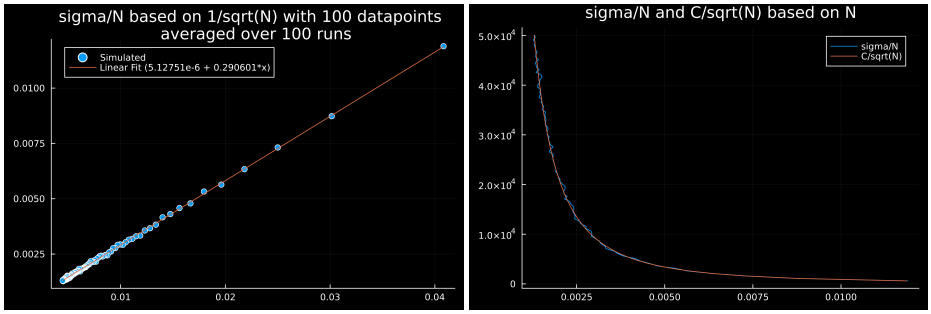

Figure 4: Results for the correlation between $\frac{\sigma}{N}$ and $\frac{1}{\sqrt{N}}$.

6.3 Central Limit Theorem

We define y as follows:

$$y = \frac{1}{N}\sum_{i=1}^{N} x_i$$

where the selected x_i is a member of the set $[5, 10, 100, 1000]$. From the central limit theorem section of the book we have the following equations:

$$< y > = y_0 =< x >$$
$$\sigma_y = \frac{\sigma_x}{\sqrt{N}}$$
$$P_{(y)} = \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{-\frac{(y-y_0)^2}{2\sigma_y^2}}$$

randomGen() function

This function takes N', the number of numbers we want to generate, N, the number of samples we take from the set, and the set we want to choose our samples from as input, then it generates random numbers based on the equation used to calculate y and outputs the results in an array.

Further Calculations and Plotting

we will also calculate the values of $< x >$, σ_x, $< y >$ and σ_y using the equations previously mentioned and we used these values to generate the $P(y)$ distribution to check the distribution of our random generator. The results for the histogram and the theoretical value plotted on top is as follows:(The histogram is normalized)

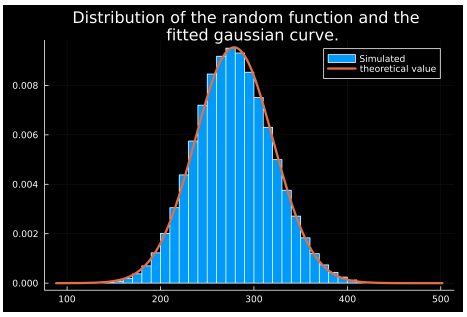

Figure 5: Distribution of the randomGen() random generator.

The distribution is just as expected and this is similar to the distribution formed in the random walk problems.

6.4 Random Generator with Gaussian Distribution

In this section we try to create a random generator with a Gaussian distribution using the built in rand() function that generates numbers with a flat distribution between 0 and 1.

gaussianRandomGenerator() function

This function takes the σ of our desired Gaussian distribution as input and generates random numbers with said distribution. The transformation done on the built in rand() function is as follows:

$$y = \sqrt{2ln(x_1)\sigma^2} \times \cos(2\pi x_2)$$

Checking the Generator

We use a loop to generate a lot of random numbers using our generator with $\sigma = 1$ and then plot the histogram of the distribution with the theoretical value on top. The result is as follows:

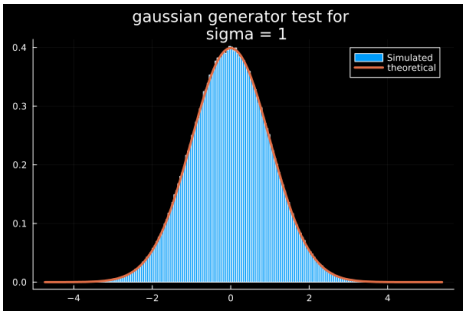

Figure 6: Distribution of the gaussianRandomGenerator() random generator.

7.1 Gaussian Integral

We want to Integrate the gaussian function using the simple sampling method and the smart sampling method.

Simple Sampling Method

randRanged() function

This function takes the upper and lower limits of a range as input, then outputs a randomly generated floating point number in that range. The function is created using the same method discussed in the book using the function:

$$y = b + (a - b)x$$

The distribution of this generator is checked to ensure it works well.

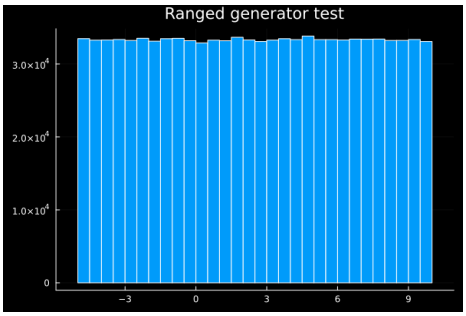

Figure 7: Distribution of the randRanged(-5, 10) random generator.

simpleIntegrate() function

This function takes the desired function, lower bounds, upper bounds and the resolution of the integral as input, then calculates the integral using the method discussed in the book and the following formula:

$$I = (b - a) < f >$$

The function then outputs the results.

simpleIntegralInaccuracy() function

This function takes the same inputs as the previous function but outputs the inaccuracy of the integral using the following formula:

$$\Delta = \frac{\sqrt{< f^2 > - < f >^2}}{\sqrt{N}}$$

Integration

The Gaussian function is then integrated using the above functions. The accurate measurement is done using integral-calculator.com with the following result:

$$I_{accurate} = 0.8820813907$$

The results are as follows:

Sample count	Execution time (s)	Results	Delta	Deviation from accurate value (%)
10000	t <0.0	0.885	0.001	0.3
10000000	0.3	0.8819	0.0001	0.02
100000000	3.5	0.88206	0.00003	0.002

Smart Sampling Method

exponentialRandomGenerator() function

For this section we need a random generator with the distribution $g(x) = e^{-x}$ so we create one using the same method as before and the following transformation:

$$y = -ln(x)$$

We can check this generator to ensure it works well:

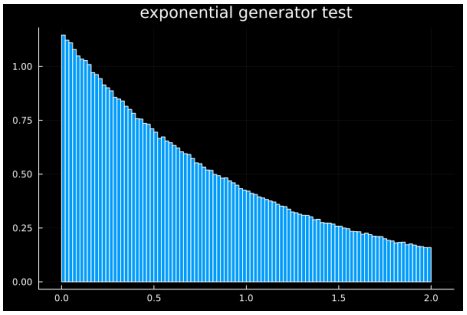

Figure 8: Distribution of the exponentialRandomGenerator() random generator.

(

smartIntegralInaccuracy() and smartIntegrate() functions) Both these functions are similar to the ones used in the simple sampling method but they use the new random generator and they used the formulas discussed in the book instead.

Integrate

The integration procedure is also the same as the previous section and we are using the same value as our accurate measurement. The results are as follows:

Sample count	Execution time (s)	Results	Delta	Deviation from accurate value (%)
10000	t <0.0	0.879	0.003	0.3
10000000	1.0	0.8820	0.0001	0.01
100000000	9.7	0.88206	0.00003	0.002

7.2 Center of Mass of a Sphere

The calculations are discussed further in the notebook file but we have to take the following integrals to solve the problem:

$$z_{cm} = \frac{1}{M}\int_0^R \int_0^{2\pi} \int_0^{\pi} r^3 \cos(\theta) \frac{\rho_{max}}{2}(\frac{3}{2} + \frac{r\cos(\theta)}{2R}) dr \sin(\theta) d\phi d\theta$$

where

$$M = \int_0^R \int_0^{2\pi} \int_0^{\pi} \frac{\rho_{max}}{2}(\frac{3}{2} + \frac{r\cos(\theta)}{2R}) r^2 dr \sin(\theta) d\phi d\theta$$

randRanged() function

This function is explained before and is also used here.

Integration

The integration itself uses the same method as the simple sampling segment of the previous problem with the difference that the multiple variables are given a random number and the formula changes to:

$$I = (up_\theta - low_\theta)(up_r - low_r) < f >$$

For a sphere with $R = 1$ we calculate that the center of mass will be approximately 0.06656 units above the center of the sphere.