EEE4120F Practical 1 Report

Jesse Arendse (ARNJES009)

February 24, 2023

0.1 Introduction

The aim of this practical is to use Julia to do statistical operations, compare different methods of executing these operations by means of performance tests and correlation.

0.1.1 Functions Created

White Noise Generator 1:
This function initializes the whiteNoise output array, and assigns the output of the single rand() call to it. The rand() call returns an array with 48000 elements, each element being a random floating point value between -1 and 1. The whiteNoise array is used to create an audio file with desired length in seconds (n). The number of elements in whiteNoise is printed and whiteNoise is returned.

White Noise Generator 2:
This function initializes the whiteNoise output array, and appends the value generated by the rand() call to it, 48000 times. The rand() call returns a single value with each call instead of an array of values, each value being a random floating point value between -1 and 1. The whiteNoise array is used to create an audio file with desired length in seconds (n). The number of elements in whiteNoise is printed and whiteNoise is returned.

Pearson's Correlation:
This function computes Pearson's Correlation between 2 input vectors, using elementwise-wise computation instead of loops, for improved performance.

Custom Sine Wave Generator:
This function creates an array of a sine wave with a specified frequency and desired number of elements (numSamples).

A link to a github repo with all the .jl files used in this practical will be listed below.

0.2 Method

The two White Noise Generator functions' execution time will be compared, with various number of samples, and will then be compared in terms of correlation with each other. Pearson's correlation (mentioned above) will also be compared to Statistics.correlation, in terms of speed and difference. Statistics.correlation will be computed between a whiteNoise array and the same whiteNoise array, circular shifted.

0.3 Results and Conclusions

Table of whiteNoise generator execution times for various samplingTimes

heightsamplingTimes	whiteNoise1Times	whiteNoise2Times	speedUp
10.0	0.0085	0.0093	0.914
50.0	0.0264	0.0152	1.7368
100.0	0.0278	0.011	2.5273
150.0	0.0202	0.0357	0.5658
200.0	0.0135	0.0149	0.906
400.0	0.0469	0.0101	4.6436
500.0	0.0195	0.01	1.95
1000.0	0.0201	0.0077	2.6104
1500.0	0.0211	0.0088	2.3977
2000.0	0.0107	0.0073	1.4658

It is mostly clear that the method 2 of generating white noise is a faster than method 1. There is not a noticable trend with samplingTime vs speedUp. It is safe to assume that the few that actually slowed down with method 2 are simply outliers, since the actual elapsed times are so tiny.

Table of different correlation functions' execution times used to compare whiteNoise to itself for various samplingTimes

samplingTimes	myCorrTimes	statCorrTimes	speedUp	myCorr	statCorr
10.0	0.0019	0.0007	0.3684	1.0	1.0
50.0	0.0018	0.0008	0.4444	1.0	1.0
100.0	0.0013	0.0008	0.6154	1.0	1.0
150.0	0.0026	0.0011	0.4231	1.0	1.0
200.0	0.0019	0.0008	0.4211	1.0	1.0
400.0	0.0019	0.0007	0.3684	1.0	1.0
500.0	0.0028	0.0007	0.25	1.0	1.0
1000.0	0.0018	0.0127	7.0556	1.0	1.0
1500.0	0.0043	0.0023	0.5349	1.0	1.0
2000.0	0.0023	0.0006	0.2609	1.0	1.0

My Correlation function clearly performs slower than the Statistics Correlation function, possibly due to using iterations instead of element-wise computations. Though the Statistics Correlation function is better in terms of speed, the two functions result in the same value, meaning they are equally accurate, and correct. Overall, the Statistics Correlation is better, since the only comparable aspect is speed. It is safe to assume that the few that actually sped up drastically are simply outliers.

Table of correlation between whiteNoise generated using different methods for various samplingTimes

samplingTimes	Correlation
10.0	-0.0008
50.0	0.0041
100.0	0.004
150.0	-0.001
200.0	-0.0027
400.0	0.0068
500.0	0.0015
1000.0	0.0062
1500.0	0.006
2000.0	-0.0048

When comparing the two whiteNoise arrays, it is clear that they are indeed random, since the correlation is virtually zero, which is no correlation.

Table of correlation between a whiteNoise signal and the same signal timeShifted for various samplingTimes

samplingTimes	Correlation
10.0	1.0
50.0	-0.8057
100.0	0.3083
150.0	0.6688
200.0	0.8089
400.0	0.951
500.0	0.9686
1000.0	0.9921
1500.0	0.9965
2000.0	0.998

Based on the previous conclusion, arrays made with the same elements but different order have a lower correlation value, which makes sense, and is what we expect to be the case with time shifted signals. But this is not the case. As the samplingTime increases, the correlation value increases, tending strongly towards 1, virtually identical sigals. This too makes sense since the timeShift value was kept at a constant value of 10, so as the samplingTime increases, the percentage of the signal that gets timeShifted tends to 0, making them more similar. Though, even at larger values of percentage timeShifted, the correlation is a lot higher than expected, as stated above. I conclude that the correlation computation is a fairly decent way to compare two signals, since the effects of a timeShift are minimal.

0.4 Appendices

https://github.com/JesseJabezArendse/EEE4120F/PRAC1/