Code Execution Performance and Correlation Investigation

Oliver Funk[†]
EEE4084F Class of 2017
University of Cape Town
South Africa
[†]FNKOLI001

***Abstract*—The effect of parameter variations of pairs of sin waves on the correlation between them is investigated. The execution performance of two ways of generating white noise and an implementation of the Pearson correlation function versus the native Octave implementation is also looked at.**

I. Introduction

Execution performance is a measure of the amount of time it takes for a computer to perform a set of actions. Because there can be multiple ways of doing the same thing, it can be difficult to choose what the best way is. Thus it is generally beneficial to find out what the fastest way of doing the same task is, in that, what set of actions will complete it in the least amount of time.

Correlation is a statical measure of how similar two datasets are. It can be used to identify trends, relate events and derive properties from the two datasets. For example, when applied to two sin waves with some phase difference, the phase difference could be approximated by looking at the correlation between the two sin waves.

In this report, the code execution performance of the native correlation function in Octave (`corr()`) and one implemented by myself (`mycorr()`), as well as two ways to generate white noise were compared. The effect of the sampling period, the number of samples and the phase difference on the correlation coefficient between pairs of sin waves was also investigated.

II. Methodology

A. Experiment Procedure

Two functions to generate white noise were written (see 1). The `rand()` function was used to generate uniformly distributed random between $[0, 1]$, however the values needed to be between $[-1, 1]$ to generate a .wav file. Thus the values from `rand()` were multiplied by 2 and subtracted by 1. The second white noise generation method was written using a for loop to examine the performance differences between the to methods.

The execution time of the two methods was measured (see code excerpt 2). The timing test was run five times and the average time for the tests was recorded.

The Pearson correlation function was then implemented (see code excerpt 1 and figure 3).

The accuracy of my implementation versus the native Octave implementation (considered to be the golden measure) was found by looking at the correlation between two identical sets of data (the white noise generated earlier), changing some of the values and checking if both functions agreed (see code excerpt 4).

The execution times of my implementation and the native Octave implementation were then found (see code excerpt 5). The white noise sample sizes ranged from 10s, 50s to 100s. The execution times were averaged over 100 runs.

Finally, pairs sin waves of various sample sizes, sample periods and phase differences were generated and the correlation between them found (see 6 for a code excerpt on how one pair of sin waves was generated).

B. Hypothesis

I hypothesize that the correlation between two sin waves would decrease as the phase difference increased. A minimum would be reached when the phase difference = $\pi/2$. This is because at that phase difference, the two sin waves change in a completely opposite manner, as the one increases the other decreases.

C. Hardware

Processor: Intel(R) Core(TM) i5-7500 CPU @ 3.40GHz (4 CPUs)

Memory: 16384MB RAM

D. Implementation

The Octave code showing the implementation of the various functions and methods is shown.

III. Results

Only the results of the various experiments that were carried out are given in this section. An analysis of the results can be found in the Findings and Conclusion section.

A. white noise execution times

The results of measuring the execution times of the white noise generating functions are given in Table I.

My implementation was $\frac{71.59}{41224.3424} = 1.73 \times 10^{-3}$ times slower than the normal function, on average.

```
function white_out = createwhiten (N)
  if mod(N, 10) != 0
    disp("N must be a multiple of 10");
    return;
  endif

  white_out = zeros(48000 * N, 1);
  for i = 1 : length(white_out);
    white_out(i) = rand()*2 - 1;
  endfor
endfunction

white = rand(48000*10,1)*2-1;
```

Listing 1. Showing the createwhiten and normal implementation for white noise generation.

```
runtime_1_avg = 0;
runtime_2_avg = 0;
times_to_run = 5;

for i = 1:times_to_run
 tic; white = rand(48000*100, 1)*2 - 1; runtime_1 = toc();
 tic; whiten = createwhiten(100); runtime_2 = toc();

 runtime_1_avg += runtime_1;
 runtime_2_avg += runtime_2;
endfor

runtime_1_avg /= times_to_run;
runtime_2_avg /= times_to_run;
```

Listing 2. Showing how the execution time of the white noise generating functions was measured.

$$r = \frac{\sum(XY) - \frac{\sum X \sum Y}{N}}{\sqrt{\left(\sum(X^2) - \frac{(\sum X)^2}{N}\right)\left(\sum(Y^2) - \frac{(\sum Y)^2}{N}\right)}}$$

Fig. 1. Pearson correlation coefficient equation

```
function r = mycorr (X, Y)
  N = length(X);
  if length(Y) != N
    disp("X and Y must be the same size");
  endif

  num = sum(X .* Y) - (sum(X)*sum(Y))/N;
  den = sqrt( ( sum(X .* X) - (sum(X)^2)/N ) * ( sum(Y .* Y) - (sum(Y)^2)/N  ));

  r = num/den;
endfunction
```

Listing 3. Showing the implementation of the Pearson correlation coefficient equation

```
x = wavread('white_noise_sound.wav');
y = x;
r1 = mycorr(x,y)
r2 = cor(x,y) # note that in some versions, this is called "corr"
disp(r2 - r1);
y(1) = 2; y(5) = -4; # i.e. fudge some of the value
r1 = mycorr(x,y)
r2 = cor(x,y)
disp(r2 - r1);
x = rand(1,10); y = rand(1,10);
r1 = mycorr(x,y)
r2 = cor(x,y)
disp(r2 - r1);
```

Listing 4. Shoiwng how the execution time of my implementation of the correlation function and the native Octave implementation was measured.

```
x = wavread("white_noise_sound_100.wav");
y = x;

runtime_1_avg = 0;
runtime_2_avg = 0;
times_to_run = 100;

for exp = 1:times_to_run
 % change some values
 for i = 1:1000
  rand_idx = floor(rand()*(length(y) - 1) + 1);
  y(rand_idx) = rand()*2 - 1;
 endfor

 tic; r1 = mycorr(x,y); runtime_1 = toc();
 tic; r2 = corr(x,y); runtime_2 = toc();

 runtime_1_avg += runtime_1;
 runtime_2_avg += runtime_2;

endfor

runtime_1_avg /= times_to_run;
runtime_2_avg /= times_to_run;
```

Listing 5. Shoiwng how the execution time of my implementation of the correlation function and the native Octave implementation was measured.

```
A = 1;
w = 2*pi;
n = 0:500;
T = 0.05;
nT = n*T;
sin_wave = A*sin(w*nT);
phi = pi/4;
sin_wave_shifted = A*sin(w*nT - phi);
r = corr(sin_wave, sin_wave_shifted)
```

Listing 6. Shoiwng how the pairs of sin waves were generated. The n, T and phi values were varied.

TABLE I
EXECUTION TIME OF WHITE NOISE GENERATING FUNCTIONS

Run	**Duration [ms]**	
	createwhiten(100)	**rand(48000*100, 1)*2 - 1**
1	41080.8022	69.1841
2	41684.763	72.191
3	40878.917	71.1889
4	40968.8981	72.192
5	41508.332	73.194
Average;	41224.3424	71.59

B. Accuracy of the correlation functions

The results of the accuracy test (see 4) are as follows:

```
Test 1:
r1 =  1
r2 =  1.00000
r2 - r1 = -1.5987e-014
Test 2:
r1 =  0.99995
r2 =  0.99995
r2 - r1 = 9.7700e-015
Test 3:
r1 = -0.20858
r2 = -0.20858
r2 - r1 = -1.2212e-015
```

C. Execution times of the correlation functions

The execution times of my implementation of the correlation function and the native Octave correlation function, for different sized files, averaged over 100 runs, is shown in Table II.

TABLE II
EXECUTION TIMES AND SPEED-UP FACTORS OF THE TWO CORRELATION FUNCTIONS

Duration [s]	Execution Time [ms]		Speed-up factor
	mycorr	corr	
10	7.3746	8.8986	1.21
50	38.522	43.1704	1.12
100	76.6087	84.7403	1.11

My implementation was $\frac{1.21+1.12+1.11}{3} = 1.15$ times faster than the Octave native correlation function.

D. Correlation of sin waves

Table III shows the different parameter that were changed for the tested sin waves. The effect of these changes on the correlation coefficient (compared to a sin wave with the same parameters with no phase) is also given.

An example of a pair of sin waves can be seen in figure 2.

IV. FINDINGS

A. white noise execution times

The execution time of my white noise generating function was significantly slower than the normal method of generating the white noise.

This is due to a for loop being used in my method, which needed to iterate through a large data set in the the order of $\approx 10^4$ to 10^6. Using matrix manipulations proved to be a far more efficient way of creating the white noise, because matrix manipulations can be optimised by making use of parallel algorithms.

B. Accuracy of the correlation functions

The accuracy of my implementation of the correlation function when compared to the golden measure (the `corr()` function) was 'perfect'. Although there may have been some computed differences between the two calculated correlation coefficients, these can be attributed to floating point errors. Conceptually, the implementation was correct and worked as intended.

C. Execution times of the correlation functions

A surprising finding was that my implementation of the correlation function performed better than the native Octave function. Although, as the dataset grew (the duration increased), the speed-up did become less and tended to 1. However, it is unknown if the speed-up may have decreased to less than 1 if a large enough dataset was used, but such a dataset could not be generated (it took too long).

D. Correlation of sin waves

Changes to the phase difference ϕ made the largest impact on the correlation function, with a minimum of 0 correlation being achieved at $\pi/2$. This agreed with my hypothesis.

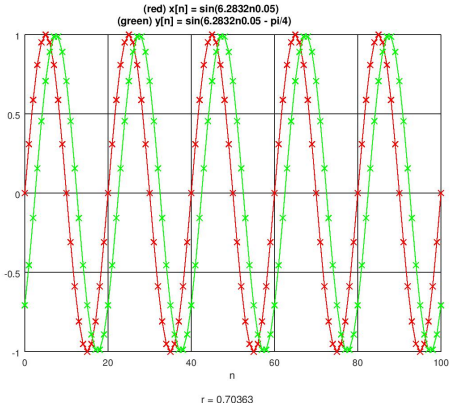

Fig. 2. A pair of sin waves with $\omega = 2\pi$, n = 100, T = 0.05, $\phi = \pi/4$

TABLE III
THE EFFECT PARAMETER VARIATIONS OF SIN WAVES HAVE ON THE CORRELATION COEFFICIENT

Sample Size	T [s]	ω [rad/s]	ϕ [rads]	Correlation Coefficient r
100	0.05	2π	$\pi/4$	0.70363
	0.05	2π	$\pi/2$	-2.81e-16 ≈ 0
500	0.05	2π	$\pi/4$	0.70640
	0.05	2π	$\pi/2$	2.9147e-15 ≈ 0
1000	0.05	2π	$\pi/4$	0.70675
	0.05	2π	$\pi/2$	3.9635e-15 ≈ 0
	0.005	2π	$\pi/4$	0.70675

Changes to the sample size for a pair of sin waves with the same phase difference did have an effect on the correlation, however it was minimal.

The changes to the sampling period had no apparent effect on the correlation.

V. CONCLUSION AND FURTHER WORK

It was found that the white noise generating function that used a for loop performed significantly worse than a function that achieved the same result using matrix manipulations. My implementation of the correlation function was found to be accurate when compared to the golden measure, the native Octave correlation function and it out performed the native Octave function, which was surprising, however, a larger dataset was needed to properly test the performance. The correlation between a pair of sin waves that had a phase difference of $\pi/2$ was 0, which agreed with my hypothesis.