Performance and Correlation Studies

John-Philip Taylor[†]
EEE4084F Class of 2016,
University of Cape Town, South Africa
[†]TYLJOH010

***Abstract*—The importance of using vectorised notation in Octave is illustrated by comparing two uses of the rand() function. The performance of a customised correlation function is compared with the Octave built-in equivalent. This correlation function is used to perform an investigation into the phase and sample-count dependence of the correlation between sinusoids.**

I. Introduction

Correlation [1] is an important tool in algorithm analysis. The results from an optimised algorithm are often compared with the results from a so-called golden measure [2]. This paper reports on the performance of a customised correlation function.

This new correlation function is used to investigate the correlation coefficient between two phase-shifted sinusoids. Various sample sizes are considered as to investigate the dependence on sample-size, if any.

II. Methodology

A. Loops vs Vectorisation

The code in listing 1 shows a loop implementation to create white noise. It creates `t` seconds of white noise, sampled at 48 ksps. This implementation was then replaced with the built-in Octave function, shown in listing 2. The performance of these two implementations was compared.

B. Correlation

The code extract in listing 3 shows a customised implementation of Pearson correlation [1]. This was compared with the built-in Octave `cor` function. The tests were performed by means of the code in listing 4. The test data (`white_noise_sound.wav`) was generated by means of the code in listing 2.

C. Sinusoids

1) Large Datasets: Fig. 1 shows the relationship between two time-shifted sinusoidal signals for various phase-shifts. Linear correlation tries to fit a straight line to this relationship [1], where the correlation coefficient is related to how closely the data resembles a straight line. By inspection of Fig. 1, one may infer that the correlation coefficient is periodic, with a period of 360°.

When the phase shift is 0°, the correlation will be 1, because the relationship forms a straight line with positive gradient. When the phase shift is 180°, the correlation will be -1, because the relationship forms a straight line with negative gradient. Following a similar argument, one can see that the correlation will be 0 when the phase shift is 90° or 270°.

```
function Output = createwhiten(t)
 # Pre-allocate some RAM
 N      = t * 48e3;
 Output = zeros(1, N);

 # Run the loop
 for n = 1:N
  Output(n) = rand()*2 - 1;
 end
end
```

Listing 1. Custom implementation of the white-noise generator

```
whiten = rand(1, t*48e3)*2 - 1;
```

Listing 2. Built-in white-noise generator

```
function r = mycorr(X, Y)
 # Precalculate terms that are re-used
 N    = length(X);
 sumX = sum   (X);
 sumY = sum   (Y);

 # Calculate correlation
 r =      (dot(X, Y) - sumX*sumY/N)/...
     sqrt((dot(X, X) - sumX*sumX/N)*(dot(Y, Y) - sumY*sumY/N));
end
```

Listing 3. Custom implementation of the correlator

```
# Test the case where the signals are equal
x = wavread('white_noise_sound.wav');
y = x;
tic(); r1 = mycorr(x, y); t1 = toc();
tic(); r2 = cor   (x, y); t2 = toc();
# Display t1, t2, relative error (abs(r2-r1)/r2) and speed-up (t2/t1)

# Test the case where there are only two samples that are different
y(1) = 2; y(5) = -4;
tic(); r1 = mycorr(x, y); t1 = toc();
tic(); r2 = cor   (x, y); t2 = toc();
# Display t1, t2, relative error (abs(r2-r1)/r2) and speed-up (t2/t1)

# Test the case where the signals are uncorrelated
x = rand(1, 1e5); y = rand(1, 1e5);
tic(); r1 = mycorr(x, y); t1 = toc();
tic(); r2 = cor   (x, y); t2 = toc();
# Display t1, t2, relative error (abs(r2-r1)/r2) and speed-up (t2/t1)
```

Listing 4. The code used to compare the `mycorr` function with the built-in `cor` function.

2) Small Datasets: The argument above is only valid for large samples, so a small-sample data-set (where the number of samples is less that a period of the sinusoid) should also be considered. Figure 2 shows the relationship for these test functions. All the relationships resemble straight lines, so for small sample sets (relative to the frequency of the sinusoid), one may infer that the resulting correlation coefficient will be close to 1 or -1.

D. White Noise

The correlation coefficient of time-shifted versions of the same white noise signal is expected to be an impulse, in

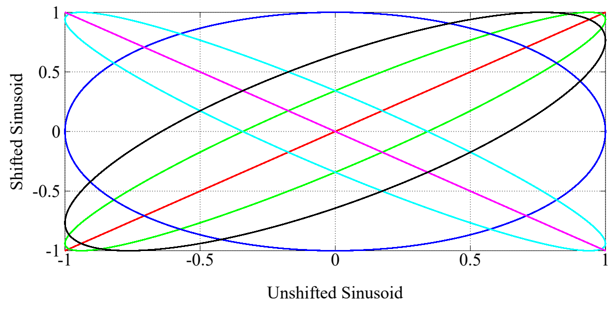

Fig. 1. Relationship between two time-shifted sinusoids. The frequency is 0.005 periods per sample, the sample-size 10 000 samples and the time-shift 0° (red), 20° (green), 90° (blue), 160° (cyan), 180° (magenta), 320° (black)

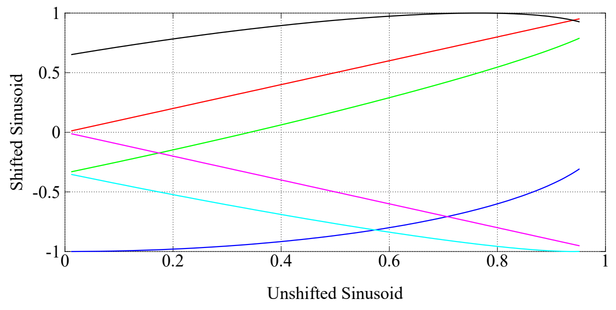

Fig. 2. Relationship between two time-shifted sinusoids. The frequency is 0.002 periods per sample, the sample-size 100 samples and the time-shift 0° (red), 20° (green), 90° (blue), 160° (cyan), 180° (magenta), 320° (black)

the ideal case. This is due to the two signals being perfectly correlated only at a time-shift of 0, and nowhere else.

III. Results

A. Loops vs Vectorisation

Table I shows the results from the comparison test on listings 1 and 2.

The built-in Octave `rand` function is much faster than the loop implementation. The memory is pre-allocated by means of the `zeros` function, so the problem does not relate to the Octave implementation of variable-length arrays.

According to the Octave website [3], Octave is a "high-level interpreted language". This implies that the script is not compiled to machine-code, but rather interpreted. Dynamic interpretation is a computationally expensive process. The vast drop in performance may therefore be attributed to the body of the loop being re-interpreted during every iteration.

The built-in `rand` function has been pre-compiled to machine code, thereby avoiding the cost of dynamic interpretation within the function, which explains the improvement in performance.

B. Correlation

The results from the `mycorr` function in listing 3 was compared with the results from the built-in `cor` function,

TABLE I
Performance comparison between loop and built-in implementations

Sample size [s]	**createwhiten() runtime [ms]**	**rand() runtime [ms]**	**Speed-up**
10	6734	20.517	328
20	13429	34.524	389
50	33644	84.546	398
100	67295	171.124	393

showing less than 10^{-15} relative error for datasets ranging from 100 to 10 000 samples. Table II shows the performance results of the two functions.

The customised function executes faster than the built-in function. As sample-size becomes very large, the relationship tends towards unity, which might indicate that the built-in function has overhead that is independent of sample-size.

Another interesting observation is that for large samples, both algorithms have a time-complexity of O(N), where N is the sample size. At small sample sizes (less than 10 000 samples), both algorithms have the same time-complexity. This is most likely related to the function calling overhead of the Octave system.

C. Correlation Between Time-Shifted Signals

The hypothesis presented in section II-C1 was tested by plotting correlation coefficient vs phase shift, which is presented in Fig. 3. The correlation coefficient is a periodic function of phase shift and resembles a cosine function, as expected.

The small data-set consideration of section II-C2 was tested by finding the correlation coefficient as a function of sample size. It was calculated for a phase shift of 90°. The result is presented in Fig. 4.

As can be seen from Fig. 4, a correlation coefficient of 0 is obtained only for large sample sizes. For small sample sizes, the correlation coefficient is close to 1, as expected.

As a final test, the correlation coefficient was calculated for various time-shifts of a white noise signal. This is presented in Fig. 5. The correlation coefficient resembles an impulse at the origin, as expected.

IV. Conclusion

This paper showed that vectorised notation in Octave execute much faster than loop-based code. Interestingly enough, the custom-implemented correlation function is faster than the built-in function, which could be attributed to the fact that the built-in function is general-purpose, whereas the customised function is application-specific.

The correlation coefficient of time-shifted signals was investigated. Time-shifted sinusoidal signals result in a sinusoidal correlation trend. Time-shifted white-noise signals results in an impulse-like correlation trend.

TABLE II
PERFORMANCE COMPARISON BETWEEN MYCORR AND BUILT-IN IMPLEMENTATIONS

Sample size [s]	**cor() runtime [ms]**	**mycorr() runtime [ms]**	**Speed-up**
100	0.505	0.512	0.986
1000	0.501	0.500	1.002
10000	1.001	0.500	2.002
100000	3.002	1.003	2.993
1000000	23.020	17.019	1.353
10000000	222.158	165.118	1.345

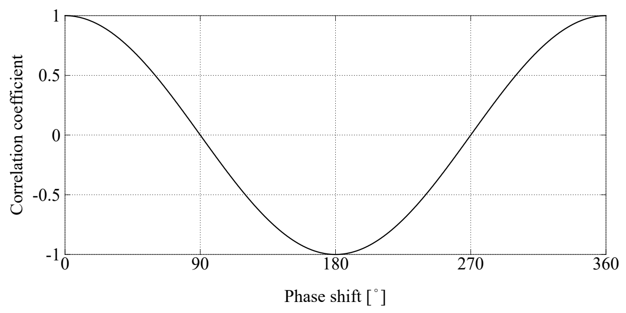

Fig. 3. Correlation as a function of phase shift of sinusoidal signals. The frequency is 0.01 periods per sample and the sample-size 10 000 samples

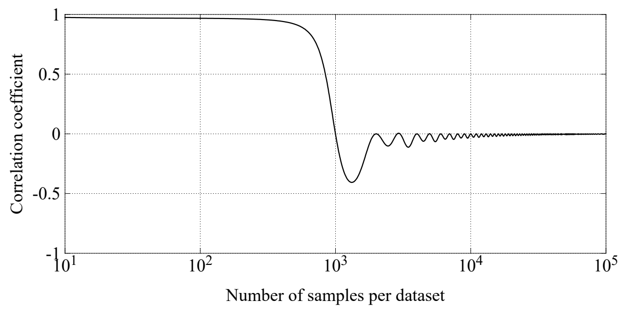

Fig. 4. Correlation as a function of sample size. The frequency is 0.0005 periods per sample and the phase shift 90°

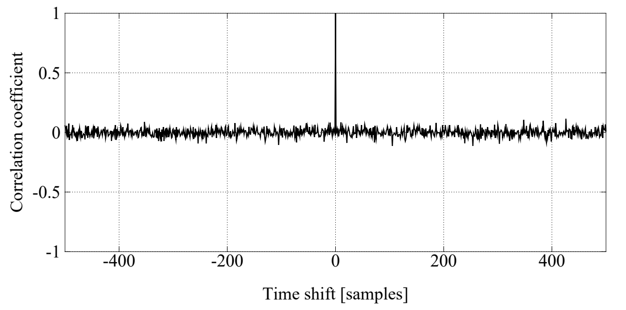

Fig. 5. Correlation as a function of time shift for a white noise signal. The data-set has 1000 samples.

REFERENCES

[1] "Pearson Product-moment Correlation Coefficient," https://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient.
[2] "EEE4084F Class Notes."
[3] "Octave Website," http://www.octave.org.