Prac 1: Octave

Roan J. Song†
EEE4084F Class of 2017
University of Cape Town
South Africa
†SNGROA001

***Abstract*—This report is a result of familiarisation with the Octave environment, and concerns the implementation and timing of native vs bespoke wave generation and correlation algorithms.**

I. Introduction

This report aims to cover the variance in performance between native Octave algorithms and a custom implementation of them, with comparison of their relative performances. The algorithms in question are noise generation and correlation.

Correlating two signals that are identical in shape but differing in phase/time introduces new questions to be considered: will the two signals still show perfect correlation? How will the correlation coefficient be affected by a shift in time, and how drastically? Does the sample rate of the signal have an effect?

II. Methodology

A. Hardware

The experiments conducted in this report were performed on a computer with the following specifications:

- CPU: i5-2500 (4 cores @ 3.3-3.7 GHz)
- RAM: 8 GB DDR3
- OS: Ubuntu 15.04

B. Introductory Experiments

To foster familiarity with Octave, a task was issued that involved generating a .wav file with a given method, and then creating one for comparison. The difference between the two functions being that the first one relies on a single call to rand() to generate multiple values, while the new method makes a call to rand() for each value in turn. A sample rate of 8000Hz was used, to decrease computation time.

The code for the custom implementation is shown below:

```
function whiten = createwhiten(time)
  sample_rate = 8000;
  whiten = zeros(time*sample_rate,1);
  samples = time*sample_rate
  for i = 1:samples
    whiten(i) = rand();
  end

  whiten = whiten*2-1; %fit into .wav bounds
end
```

It is unlikely that the custom implementation will be able to perform better than the likely highly optimized native function.

C. Custom Correlation Implementation

The source code for the bespoke correlation solution is shown below. I would expect this code to run more quickly than the native implementation at least for some cases, as it relies on basic arithmetic principles, and offers no additional functionality.

```
function r = mycorr(x,y)
        % readability >> succinctness
        sx  = sum(x);
        sy  = sum(y);
        sxy = sum(x.*y);    % elementwise multiplication
        sx2 = sum(x.^2);    % elementwise squares
        sy2 = sum(y.^2);
        n   = max(size(x)); % just in case we get some column/row vector mixups
        num = sxy - sx*sy/n;
        den = sqrt((sx2-(sx.^2)/n)*(sy2-(sy.^2)/n));
        r = -2; % default, in case correlation could not be performed
        if(den != 0)
                r = num/den;
        end
end
```

D. Comparing Signals Shifted in Time

Firstly, a signal must be generated, and then compared against a shifted version of itself. This can easily be done by generating a signal and then using a small 'window' of the signal compared to a window shifted slightly to either side, i.e. if the signal x is comprised of 100 samples, one signal could be x[1:50] and the shifted version 20 samples ahead, at x[21:70]. For all instances, I used a window size of 50 samples.

The signals generated were sine waves, at frequencies of 1rad/s, 2rad/s, and 10rad/s. All of the signals generated range from -pi to pi. 100, 1000, and 10000 samples were used for each frequency. The choice of windows are [1:50] (the first 50 samples of the signal), [2:51] (a shift by 1), and [11:61] (a shift by 10). Figure 2 shows the absolute value of the differences between shifted signals at different sample rates, indicating that higher sample rates have smaller differences between nearby samples than lower sample rates.

These values were chosen to cover a wide range of signals, and to avoid jumping to conclusions.

It is hypothesised in this report that correlating a sinusoid with a time-shifted version will have the following effect on the Pearson's Correlation Coefficient:

- Sinusoids are periodic, so shifting by one period will give the same correlation value
- Sinusoids are odd, so after half a period, the correlation will be at a minimum. It should be that r=-1 at this point, as the signals are inverses.

- The correlation coefficient will smoothly vary between 1 and -1 as the signal is shifted.
- Higher sample rate signals will be less susceptible to sample-based shifts but equally susceptible to time-based shifts.

The correlation between each signal was calculated using the native corr function as follows:

```
a = linspace(-pi,pi,100); %100 samples
x1  = sin(a*1); %frequency of 1rad/s
w1 = 1:50;
w2 = 2:51;
r = corr(x1(w1),x1(w2));
disp(r);
```

III. Results & Discussion

A. Introductory Experiments

The given implementation took 173ms to generate 1000s of 8000Hz white noise. The custom implementation took 64244ms. This new method is a handy 370 times slower than the native method. This speed-down is likely due to the overheads of calling the rand() function multiple times.

B. Custom Correlation

It appears that the custom correlation implementation performs faster than the native Octave algorithm. The average speedup varies between $\tilde{1}6$ on smaller vectors to just above 1.2 on very long (10^8) vectors.

mycorr (ms)	corr (ms)	Speed-up	Sample Size
2.4128	3.7909	6.364	10
2.1110	1.3089	16.128	100
2.1570	1.5306	14.092	1000
2.4791	2.6298	9.427	10000
3.7451	1.3590	2.756	100000
21.465	11.776	1.823	1000000
140.40	104.61	1.342	10000000
1344.2	1042.5	1.290	100000000

TABLE I
Table comparing two correlation implementations

C. Effects of Time Shifts on Correlation

Table II shows Pearson's Correlation Coefficient for each signal, at each frequency, with a varying shift, for each sample rate.

100 samples	1 rad/s	2 rad/s	10 rad/s
shift by 1	0.989	0.991	0.802
shift by 10	0.494	0.294	0.998
1000 samples	**1 rad/s**	**2 rad/s**	**10 rad/s**
shift by 1	1.000	1.000	0.990
shift by 10	1.000	0.999	0.479
10000 samples	**1 rad/s**	**2 rad/s**	**10 rad/s**
shift by 1	1.000	1.000	1.000
shift by 10	1.000	1.000	1.000

TABLE II
Correlation on various shifted signals

Experiments show that a signal correlated with a time-shifted version of itself will have an increasingly low Pearson's Correlation Coefficient. In the case of an odd, periodic signal, shifting by half its period will result in a

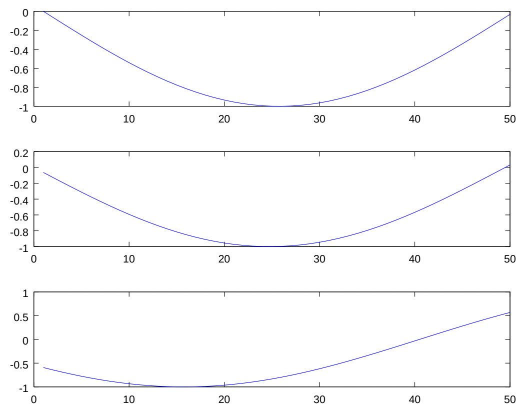
Fig. 1. Different windows of the same signal

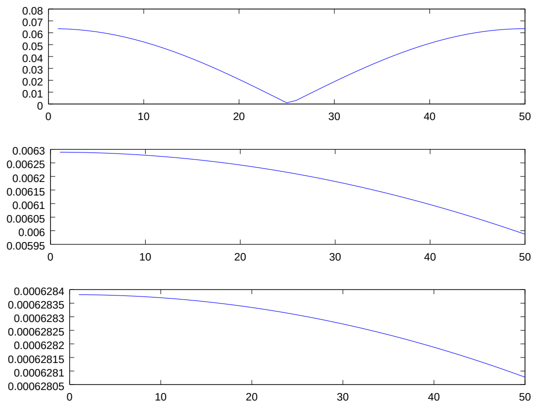
Fig. 2. Absolute differences between a signal and shifted counterpart, according to sample rate

correlation of r = -1, and r=1 if the signal is shifted by an exact multiple of the signal's period. Higher frequency signals are more susceptible to sample shifts, as each sample constitutes a higher proportion of a single period of the waveform. Signals with a higher sample rate are less susceptible to sample shifts, given that they have more samples constituting a single period of the waveform. I further infer that a shift in seconds instead of samples would provide the same results for each frequency, and ignore the sample rates, as the values at each point would be the same given that the point was defined for each signal.

IV. Conclusions

The experiments performed served to confirm the expectations of how correlation of two sinusoids would be affected by sample-based shifts. Further research on this topic can be done, to clarify some points regarding time-based vs sample-based shifting of signals. This report also served to confirm that a custom implementation of the white noise generation algorithm failed to surpass the performance of the native implementation.