CAMBRIDGE UNIVERSITY ENGINEERING TRIPOS PART IB

IB INTEGRATED COURSEWORK: EXTENDED EXERCISE REPORT

Student Name: Oliver Griffiths Email: [owg21@cam.ac.uk](mailto:owg21@cam.ac.uk) College: Selwyn College

Date submitted for marking: 28/11/2023 Lab Group: 101

Title: Investigating Windowing to Improve Earthquake Detection

Main topic area(s): *(delete as appropriate)* ~~Vibration~~ / ~~Soils~~ / ~~Structures~~ / Signals

Marker: ............................................................................... Date: ............................

*Please highlight one mark in each row:*

| Technical content mark: | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Report quality mark: | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

**Marker’s Comments:**

**IB Integrated Coursework. Investigating Windowing to Improve Earthquake Signal Analysis**

**Abstract:** This reportinvestigates the use of the windowing function to improve the accuracy of the fast fourier transform when analysing signal data. The analysis will be undertaken with particular consideration to its effectiveness in relation to earthquake signals that may be unpredictable and noisy.

**Introduction:** In seismic hazard zones, structural engineers are challenged with designing structures capable of survivng large seismic shocks, requiring careful material selection and design practices such as inertia damping. Not only is earthquake monitoring important for alert systems, knowing the precise magnitudes and frequencies of earthquakes is vital to allow for the effective design and maintainance of these structures.

The aims and objectives of this report are as follows:

* **Understand** the purpose of windowing. When is it needed, and what factors should be taken into account.
* **Compare** different types of windowing functions, investigating their properties.
* **Evaluate** these functions, commenting on how they performance is affected by the properties of the signal being analysed.

These aims will be researched and confirmed through analysis of generated data of known frequencies and amplitudes, performing this analysis in python with the NumPy and SciPy libraries.

To perform the Fast Fourier Transform, the signal we are interpreting will be sampled over a finite period, and then analysed as if it is a periodic function. This can provide problems if the period of our signal is unknown, and our periodic signal is not sampled over its period. large discontinuities in the time domain will occur. These discontinuities in the time domain can result in high amplitude and high frequencies in the frequency domain leading to false data known as spectral leakage.

Figure 2: Sampled Sine Signal with Blackman Windowing Function Overlayed

Figure 1: Sampled Sine Signal

**A graph of a function

Description automatically generated with medium confidenceA graph of a function

Description automatically generated with medium confidence**

Frequency

Frequency

To address these large discontinuities, seen at the edge of the window, shown by green dashes in **Figure 1**, we can apply a windowing function between the window where our signal is interpreted. This involves multiplying our interpreted signal by a function that aims to eliminate the discontinuities by bringing our signal to 0 at the boundaries of our window. The Blackman windowing function can be viewed in green on **Figure 2**. This results in our windowed signal, which can be seen in orange in **Figure 3.**

A graph of a function

Description automatically generated with medium confidenceA graph of a function

Description automatically generated with medium confidence

Figure 3: Windowed Interpreted Signal

Figure 4: Windowed Interpreted Signal over larger window

Frequency

Frequency

In **Figure 3,** the discontinuities have been removed, but the amplitude of the signal has been reduced. To minimise this, a large window should be used, covering a range greater than the period of the lowest expected frequency signal. In **Figure 4** we can see the effect of a larger window on the amplitude of our signal.

In this report, 3 windowing functions available in the NumPy library will be used. Blackman, Bartlett, and Hamming. They are visualised below.

A graph of a line graph

Description automatically generated with medium confidence

Figure 5: Window Functions Visualised

Hamming appears most interesting as it does not drop to 0 at the edges of the window. This may provide some interesting results for signals that are less predictable in nature.

**Testing and Results:**

For all testing, the signal being transformed will consist of 3 sine functions at 5Hz, 20Hz and 40Hz, chosen as these are most indicative of the frequencies present in an earthquake. Their amplitudes will be 0.3, 0.2 and 0.1 correspondingly.

**Rectangular (Figure 6):**

No windowing function is applied. Good peak definition, but with a high noise floor

**Blackman (Figure 7)**

The Blackman function is formed of the first three terms of a summation of cosines, coming to 0 at the edges of the window.

From the figure we can see a good reduction in the noise floor, as well as a better distinction between the lower frequencies. The peak amplitudes have been smeared, however.

Figure 7: Blackman Window Spectrum

Figure 6: Rectangular Window Spectrum

**A graph with blue lines

Description automatically generated**A graph with blue lines

Description automatically generated

Frequency

Frequency

**Bartlett (Figure 7)**

The Bartlett window function is a triangle wave, coming to zero at the edges of the window. Less smearing of the peaks occurs with this function, with less noise reduction.

**Hamming (Figure 8)**

The Hamming window is formed of a raised cosine signal, designed such that it doesn’t go to zero at the edges of the window function and good at eliminating the side lobes of an FFT. Here, we can see that there is little peak smearing, but at the expense of noise reduction at higher frequencies.

**A graph with blue lines

Description automatically generated**A graph with blue lines

Description automatically generated

Frequency

Frequency

Figure 9: Hamming Window Spectrum

Figure 8: Bartlett Window Spectrum

**A graph of different colored lines

Description automatically generatedDiscussion:**

Figure 10: All Window Type Spectrums

Frequency

**Figure 10** shows the Spectrum created using the 3 windowing functions. We can see that bartlett is most effective at reducing the noise floor but has significant peak smearing. For earthquake signals, which will typically be from 0.2Hz to 20Hz, the hamming function appears the most effective. With it having both the most accurate peaks, as well as clear peak separation and a good reduction in the noise floor, it means that frequency that are close together can be reliably separated, as well as eliminating the chance of spectral leakage.   
Although, as earthquake data is likely to be unpredictable, with large amplitude waves occurring only briefly, there is a chance that any windowing function may supress these amplitudes if they occur near to the window edges. This means buildings may be subject to significantly higher amplitudes than the hamming window suggests.

I think opting to analyse using a rectangular window in conjunction with the Hamming window may be the best method to interpret earthquake data. By first consulting the rectangular window to check for any potential abrupt peaks, and later consulting the Hamming window to evaluate any seemingly anomalous results or spectral leakage, random earthquake data won’t be omitted. Any seeming anomalous results should be investigated further, using alternate methods to understand if the result is valid.

**Appendix:**

View the python code used to analyse different window functions.

import numpy as np

from scipy.fft import fft, fftfreq, fftshift

import matplotlib.pyplot as plt

signaltime = 10

sr = 1000 #SAMPLE RATE, samples per second

sp = 1/sr

sn = signaltime \*sr #number of samples

signals = ([40,0.1],[5,0.3], [20,0.2])

time = np.linspace(0,signaltime, sn)

signal = np.zeros(len(time))

for i in signals:

    signal += i[1] \* np.sin(2 \* np.pi \* i[0] \* time + np.random.rand(time.size))

windowupper = 700

windowlower = 200

windowwidth = windowupper - windowlower

sampledsignal = signal[windowlower:windowupper]

windowfunction1 = np.hamming(windowwidth)

windowedsignal1 = sampledsignal\*windowfunction1

windowfunction2 = np.bartlett(windowwidth)

windowedsignal2 = sampledsignal \* windowfunction2

ws3 = sampledsignal \*np.blackman(windowwidth)

repeats = -(sn // -windowwidth) +1

repeatedsamplesignal = np.tile(sampledsignal,repeats)[windowwidth-windowlower%windowwidth:(windowwidth-windowlower%windowwidth)+sn]

repeatedwindowsignal = np.tile(windowedsignal1,repeats)[windowwidth-windowlower%windowwidth:(windowwidth-windowlower%windowwidth)+sn]

yf = fft(sampledsignal)[:windowwidth//2]

yfw1 = fft(windowedsignal1)[:windowwidth//2]

yfw2 = fft(windowedsignal2)[:windowwidth//2]

xf = fftfreq(windowwidth, sp)[:windowwidth//2]

yfw3 = fft(ws3)[:windowwidth//2]

Various plots were used to visualise the data. The absolute values for the amplitudes of the signal were used in the plots.