CAMBRIDGE UNIVERSITY ENGINEERING TRIPOS PART IB

IB INTEGRATED COURSEWORK: EXTENDED EXERCISE REPORT

Student Name: Jack Henry Email: jbh48@cam.ac.uk College: Churchill

Date submitted for marking: 13/03/24 Lab Group: 54

Title: Investigation into windowing of the discrete Fourier transform

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:

Investigation into windowing for the discrete Fourier transform

1. Introduction and Plan

Introduction:

The DFT algorithm, like the Fourier transform, is designed to operate on a continuous signal (i.e., continuous in time). As it is not possible to measure a signal for all time, the continuous signal is synthesised from that measured during a finite time period (period T). This will usually produce a discontinuity in the signal when an attempt is made to match up the first and last sampled points, as shown by the two figures below.

This causes a problem, as the Fourier transform treats the discontinuity as part of the original function, leading to spurious results (termed spectral leakage). This can be mitigated by the use of windowing. A window function multiplies the time domain signal to reduce or remove this discontinuity. This is demonstrated in the diagrams below using a Hamming window - this is a particular type of window function, of which there are many.

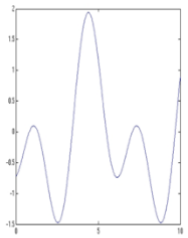


Figure 1: Sampled wave

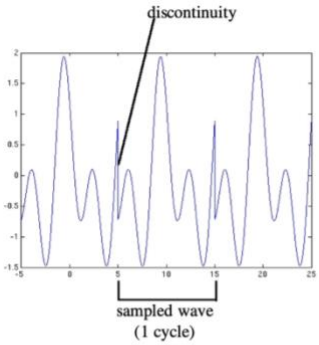


Figure 2: Repeated wave used in the Fourier transform/DFT

While the use of a window function can be beneficial, it can also have drawbacks. In this report these effects are investigated further.

Objectives:

1. Understand the purpose of windowing, when it might be needed, and what factors should be taken into account.
2. Research various different types of window functions and investigate their properties.
3. Comment on how the signal properties affect the choice of window.
4. Design and conduct an experiment to test the effects of various types of windowing function using python code and creating a signal out of the modes of the model building.

Why?:

Almost every other investigation and experiment within the extended exercises uses and relies on a DFT function in order to collect data about frequencies. If this process is not accurate or understood, incorrect conclusions may be made about the behaviour of the earthquake-building system. For instance mass dampers have to be tuned to exact frequencies, and spectral leakage from the DFT could be misinterpreted as noise created from the building or earthquake system

2. Set up and method

This investigation used a purely coding base for signals in order to maximise control over variables such as frequency and noise with precision. Whilst windowing can be used for any arbitrary signal, it was decided to use a signal made up of 3 sine waves, made from the 3 modes of vibrations of the model building being investigated, as these were the most important frequencies.

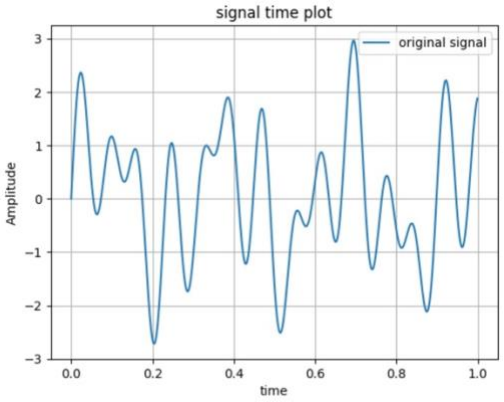


Figure 3: Signal made from modes of vibration

These were: 3.3Hz, 9.0Hz, and 13.3Hz. The signal is shown on the right. It is worth noting that this signal itself is generated at 1kHz, effectively sampling the signal at a sampling rate of 1kHz.

The numpy library was used to find the FFT of this signal as well as use various windows and show their effects. Ideally, a signal made of 3 sine waves should have a spectrum of 3 impulses at the corresponding frequencies. This is worth noting in order to compare the theoretical result to the ones collected in this experiment

It was discovered quickly that more detailed and accurate frequency plots could be achieved by increasing the window time. This process was tested showing the DFT was sensitive location and number of discontinuities. This is not shown in this report for the sake of conciseness, but it is important to note. A window time of 10 seconds was eventually used as a good compromise between processing speed and accuracy.

The windowing process in the time domain is shown on the right for clarity. This particular example uses the bartlett (or triangular) window. The most important part of the window is that the amplitude starts and ends at 0, so that when the sample is repeated, there is no (or minimal) discontinuities.

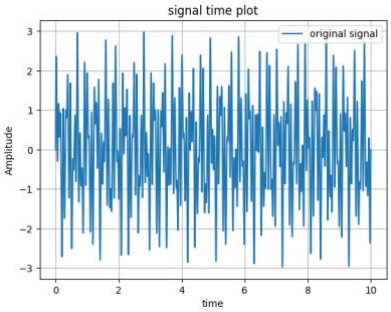


Figure 4: 10s of signal with no window

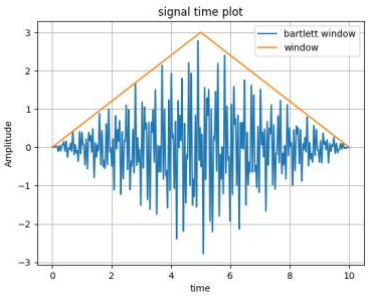


Figure 5: 10s of signal with bartlett window

3. Windows

5 (or technically 6) windows were used and analysed. Due to a complete DTFT requiring an infinite number of sums, the signal without a window is equivalent to a uniform or rectangular window as shown below. This itself also has an effect on the signal which can be seen later in the frequency plots.

Each window has it's own Fourier transform (as shown in the figures). This means that when the signal is multiplied by the window, the theoretical impulse spectrum of the signal is convoluted with the spectrum of each of the signals (as shown by figure 13[2]). This explains a lot of the behaviour in the results. Each window spectrum has particular characteristics – this is discussed later.

The collection of windows with their respective Fourier transform spectrums is shown below[1].

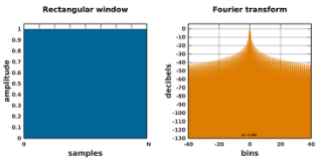


Figure 6: Rectangular window

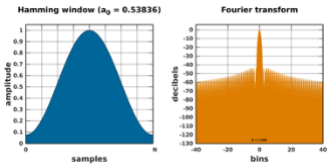


Figure 7: Hamming window

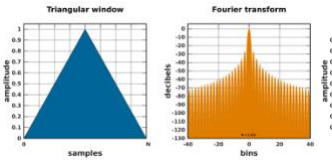


Figure 8: Bartlett window

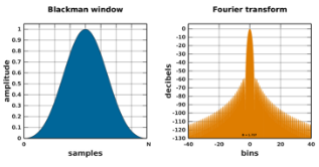


Figure 9: Blackman window

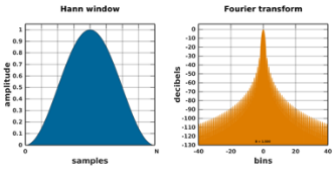


Figure 10: Hanning window

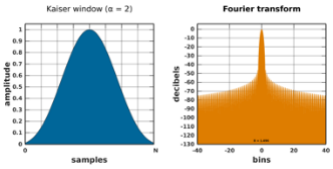


Figure 11: Kaiser (2) window

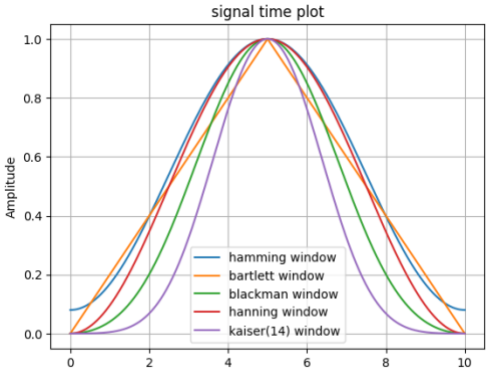


Figure 12: Python generated windows

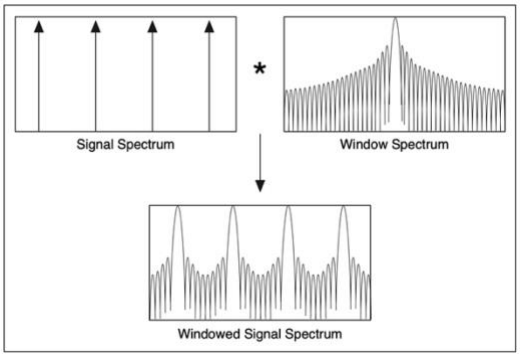


Figure 13: Convolution process

Noteworthy windows are:

The hamming window; the only window not to start and end at 0 (other than rectangular).

The kaiser window; this has a 'narrowing parameter' – of which we used a value of 14, i.e. reasonably narrow as shown by figure 12.

4. Results and Analysis

The combined frequency plots for all the various FFT spectra of the windows applied to our 3 mode signal are shown in figures 14 and 15 below.

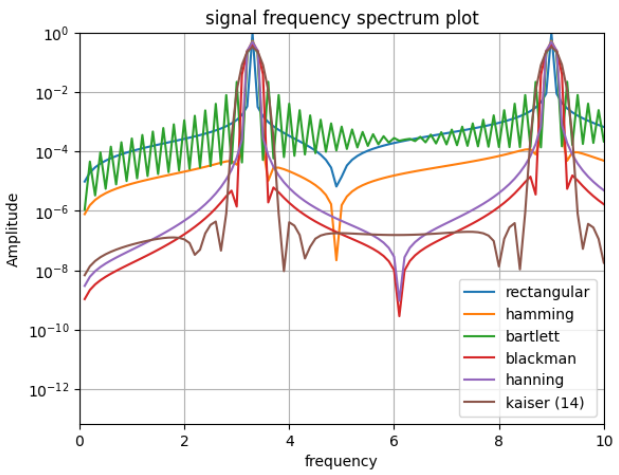


Figure 14: spectrum plot of FFTs for 0 to 10Hz

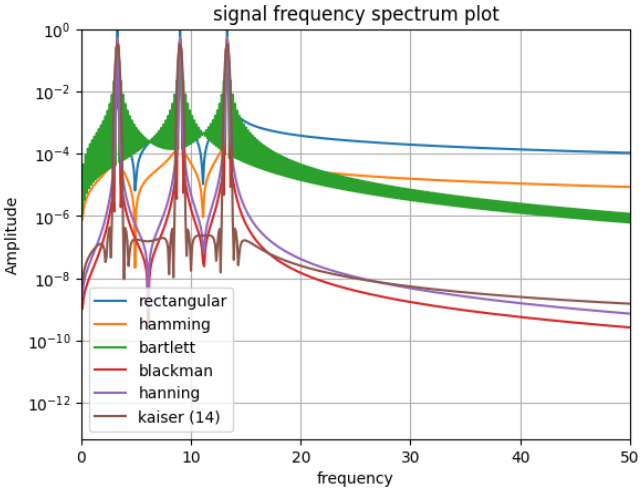


Figure 15: spectrum plot of FFTs for 0 to 50Hz

(It is worth noting that the side lobes which have a 'bouncing' behaviour cannot be seen in these graphs due to the rate of 'bouncing' causing them to combine into a smooth line. This behaviour can still be observed in the bartlett window spectrum due to the lower rate).

Whilst these graphs seem somewhat congested at first, some key characteristics can be seen immediately. It is clear which windows have lower spectral leakage from figure 15 – rectangular, hamming and triangular all have more spectral leakage whilst blackman, hanning and kaiser have lower (and lower spectral leakage is generally considered desirable). The same can be said for the 'side-lobe peak level', with the windows with less spectral leakage having a lower amplitude immediately outside of the main lobes which are centred on the frequencies of the modes.

The less obvious characteristic can be seen after a closer look at figure 14. It can be observed that different windows have differing main lobe widths, with the wider main lobes have a more rounded peak. Generally, the windows with lower spectral leakage have a wider and more rounded main lobe.

It is clear to see that there is a balance between spectral leakage and main lobe width and roundness.

5. Discussions

Figure 16 shows the main characteristics of the window spectra observed.[2]

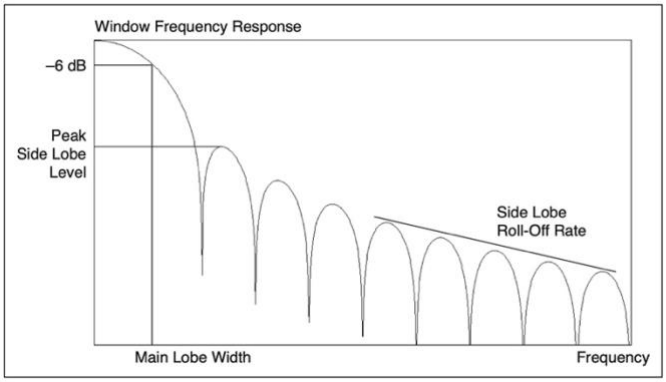


Figure 16: Characteristics of a window spectrum

Each window has its own characteristics, and different windows are used for different applications. Choosing a window also depends on whether knowledge of the frequency content is known. If the signal contains strong interfering frequency components distant from the frequency of interest, a window with a high side lobe roll-off rate is often more desirable. If there are strong interfering signals near the frequency of interest, a window with a low maximum side lobe level is now more desirable.[2]

If the frequency of interest contains two or more signals very near to each other, spectral resolution is important. In this case, it is best to choose a window with a very narrow main lobe. If the amplitude accuracy of a single frequency component is more important than the exact location of the component in a given frequency bin, a window with a wide main lobe is preferable, whilst if the signal spectrum is rather flat or broadband in frequency content, the Uniform window (no window) is now preferable.[2]

6. Conclusion

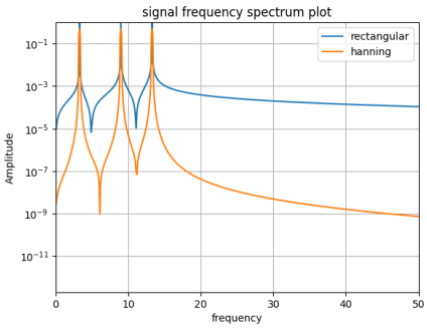


Figure 17: Rectangular and Hanning window spectrum

In general, the Hann window is satisfactory in 95% of cases. It has good frequency resolution and reduced spectral leakage. For this investigation specifically, the exact frequency content is known, and they are reasonably spread out, therefore spectral leakage is less relevant and accuracy of location of the frequencies is of higher priority. This means that the rectangular window seems to be more desirable. This is only the case due to the order of our investigation however. In reality, the signals frequency content is unknown, and there may be frequencies close to each other, so a balance of accuracy and low spectral leakage is needed. The Hanning window has the best compromise for our specific signal, however other windows may perform better for different signals – more investigation would be needed to conclude which windows are better in which situations.

7. Appendix

Python code used for investigation:

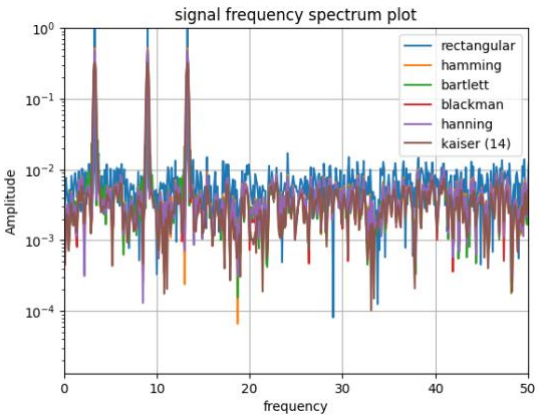
```
import numpy as np
from numpy.fft import fft, fftfreq
import matplotlib.pyplot as plt

signaltime = 10       # time that signals run for
sr = 1000             # sample rate (per second)
sp = 1/sr             # sample interval
sn = signaltime *sr   # number of samples
t = np.linspace(0,signaltime, sn)       # time array

def signal(f1,f2,f3, noise=False):
  signals = ([f1,1],[f2,1], [f3,1])   # signals generated (based on mode frequencies)
  sig = np.zeros(len(t))            # 0 array for signals

  if noise==False:
    for i in signals:     # effectively sampling the signals at desired sample rate and superposing over each other
      sig += i[1] * np.sin(2 * np.pi * i[0] * t) # + np.random.rand(t.size))

  if noise==True:
    for i in signals:     # with noise
      sig += i[1] * np.sin(2 * np.pi * i[0] * t + np.random.rand(t.size))
  return sig

f1 = 3.3
f2 = 9
f3 = 13.3
signal = signal(f1,f2,f3, noise=True)

uw = (1/100)*100   #percentage of total time for window
lw = (1/100)*0

windowupper = int(uw*signaltime*sr)     # upper window limit (in time)
windowlower = int(lw*signaltime*sr)     # lower window limit (in time)
windowwidth = windowupper - windowlower     # window width (in time), and also number of samples within window

sampledsignal = signal[windowlower:windowupper]     # sampled signal only in windowed time space
wf1 = np.hamming(windowwidth)           # creating hamming window function
ws1 = sampledsignal*wf1                 # applying hamming window to reduced sampled signal
wf2 = np.bartlett(windowwidth)          # creating bartlett window
ws2 = sampledsignal * wf2               # applying bartlett window to reduced sampled signal
wf3 = np.blackman(windowwidth)
ws3 = sampledsignal *wf3       # creating and applying blackman window to reduced sampled signal
wf4 = np.hanning(windowwidth)
ws4 = sampledsignal *wf4
wf5 = np.kaiser(windowwidth,14)
ws5 = sampledsignal *wf5

repeats = -(sn // -windowwidth) +1
repeatedsampledsignal = np.tile(sampledsignal,repeats)[windowwidth-windowlower%windowwidth:(windowwidth-windowlower%windowwidth)+sn]

x = fftfreq(sn, sp)[:sn//2]
y = fft(signal)[:sn//2]
xf = fftfreq(windowwidth, sp)[:windowwidth//2]  # frequency array
yf = fft(sampledsignal)[:windowwidth//2]        # FFT of sampled signal
yfw1 = fft(ws1)[:windowwidth//2]    # FFT of sampled signal with hamming window
yfw2 = fft(ws2)[:windowwidth//2]    # FFT of sampled signal with bartlett window
yfw3 = fft(ws3)[:windowwidth//2]    # FFT of sampled signal with blackman window
yfw4 = fft(ws4)[:windowwidth//2]    # FFT of sampled signal with hanning window
yfw5 = fft(ws5)[:windowwidth//2]    # FFT of sampled signal with kaiser window

yw1 = fft(wf1)[:windowwidth//2]     # FFTs of purely the window
yw2 = fft(wf2)[:windowwidth//2]
yw3 = fft(wf3)[:windowwidth//2]
yw4 = fft(wf4)[:windowwidth//2]
yw5 = fft(wf5)[:windowwidth//2]
```

Appendix 1: Python code used to generate signal, windows, and corresponding spectra

References:

1. https://en.wikipedia.org/wiki/Window_function
2. https://www.sjsu.edu/people/burford.furman/docs/me120/FFT_tutorial_NI.pdf

A note on noise: (outside of our investigation)



Appendix 2: Spectra of investigation with noise within signal

Whilst the role of noise within this investigation is not clear, and outside the scope of this investigation, this additional figure is included to show the rough effect of noise on the spectra. It seems that spectral leakage is now less relevant due to the main lobe being applied to the noise as well, and so the main lobe characteristics dominate. The rectangular window maximises both the desired frequency signal as well as the noise, whilst the kaiser window's characteristic of having a lower amplitude more rounded main lobe has the effect of also reducing noise.