CCE2203 Lab 3: Fourier Transform

Johann A. Briffa

September 23, 2024

Instructions

- This unit of assessment is to be attempted individually. It is essential that the work you submit and present consists only of your own work; use of copied material will be treated as plagiarism. Discussion is only permitted on general issues, and it is absolutely forbidden to discuss specific details with anyone.
- Your lab submission consists of the following deliverables:
  - A report, in the form of a Jupyter Notebook, submitted on the VLE as a single PDF file. The file needs to be less than 20 MiB in size. Be particularly careful with the sizes of any included images, which can easily cause the PDF to be too big.
  - A completed and signed Plagiarism Form for this unit of assessment.

  If any of these submissions is late, the whole unit of assessment will be considered a late submission, even if any part was submitted on time. Other methods of submission will not be considered.
- The report should be paginated on A4 paper, and exported directly as PDF from Jupyter notebook.
- Separate your responses for each question, including a Markdown header to separate the various parts.
- Use a sequence of code blocks, answering each question separately, rather than putting all the code in one big block. Questions need to be answered in sequence.
- Textual answers to any questions must be included in the report, as a Markdown cell. Each answer should appear immediately after the results to which it refers.
- In your submission, include only content that *directly* answers the questions asked. Submission of irrelevant material may lead to a reduction in the grade obtained.
- The deliverables are to be submitted on the VLE by the deadline specified there; late submissions will be rejected and assessed with a grade of zero.
- If there are extenuating circumstances which do not allow you to complete the unit of assessment on time, you are required to follow the procedure specified in the regulations.

Aims and Objectives

The aim of this laboratory session is to use the Fourier transform to analyse real-world signals in the frequency domain. Based on this analysis, a simple frequency-domain noise-removal filter is implemented, and its performance investigated.

1 Preparation

In this lab we will use the same WAV files provided for Lab 1.

2 Frequency Domain Analysis

2.1. As in Lab 1, we start by loading the file titled `imperfect_sos_cw_700_15_wpm.wav`, using the code fragment below.

```
fs, x = wavfile.read(os.path.join('data','imperfect_sos_cw_700_15_wpm.wav'))

ipd.display(ipd.Audio(data=x, rate=fs))
```

Use the interactive component to play the loaded data, confirming this is the clean version of the SOS signal.

2.2. We can now use the NumPy implementation of the discrete Fourier transform to determine the frequency spectrum of the given signal, as shown in the code fragment below.

```
X = np.fft.fft(x,norm='forward')
F = np.fft.fftfreq(len(X), 1/fs)
```

Note that we use the forward normalisation method, which places the normalisation during the forward transform; this allows us to interpret the magnitude of the frequency domain coefficients as the amplitude of the corresponding frequency component. The second line uses a helper function to obtain the frequency corresponding to each coefficient.

2.3. To plot the magnitude of the frequency response, we can use the following code fragment.

```
plt.figure()
plt.stem(np.fft.fftshift(F),np.fft.fftshift(np.abs(X)),markerfmt='b.')
plt.xscale('log')
plt.yscale('log')
plt.ylim(1e-1,1e5)
plt.grid(True)
plt.xlabel('Frequency␣$f$')
plt.ylabel('Value␣$X(f)$')
plt.title('Clean␣SOS')

plt.show()
```

Note the use of the `fftshift` function to shift the zero-frequency component to the center of the spectrum. (This will not show up on a log-scale x-axis, but will be visible on a linear axis.) We also limit the range of the y-axis. Since the WAV file encodes samples with 16-bit precision, the range of representable values is about $\pm 2^{15} \approx 10^5$.

2.4. Comment on the observed frequency spectrum for the clean SOS signal. Can you use this to determine the frequency of the carrier wave? **[10 marks]**

2.5. Repeat this analysis for the `noisy_sos_cw_700_15_wpm.wav` file. Load the file and plot its frequency response in a comparable way to what we did earlier. **[20 marks]**

2.6. Comment on the observed frequency spectrum for the noisy SOS signal. Can you say anything about the frequency characteristics of the noise? **[10 marks]**

3 Frequency Domain Filtering

Given the limited frequency range of the signal of interest, and the wide frequency range of the noise, a rudimentary way to remove noise is simply to keep the coefficients for the frequency range of interest, and discard the rest.

3.1. From the frequency responses observed, determine a suitable range of frequencies of interest to keep. **[10 marks]**

3.2. Implement a frequency domain filter by discarding (zeroing out) the coefficients in `X` that correspond to frequencies `F` outside the range of interest. Plot the frequency response of the filtered signal. **[20 marks]**

3.3. Using the NumPy `ifft` function, from this filtered frequency response obtain the corresponding time-domain signal. *Hint: you are only interested in the real part of the result, so if the inverse transform returns any imaginary component, this may be safely discarded.* **[10 marks]**

3.4. Plot the time-domain filtered signal in a similar way to what we did in Lab 1, and comment on the result obtained. **[10 marks]**

3.5. Play back the filtered signal and comment on the efficacy or otherwise of this approach. **[10 marks]**