CCE2203 Lab 1: Convolution

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 process a real-world signal using convolution. While it is certainly possible to use other languages, in this study unit we will exclusively use Python for such practical tasks. This lab sheet will guide you through the steps that need to be done. Some of these steps are given, and carry no marks, while others you will need to perform yourself, and carry a mark as indicated.

1 Preparation

On the VLE you have been provided with a selection of audio signals in the form of WAV files. The WAV format is an uncompressed format for audio signals, making it easily readable on a range of platforms. These audio signals are recordings of Morse code sequences being received clean or in the presence of noise. In this lab we will see how we can apply what we learned in class to clean up the noisy signals.

1.1. In Python we can read WAV files using the SciPy package, which is available through the Ubuntu repositories. This is already installed in the lab machines; to install it on your own computer, please follow the instructions given in class.

1.2. Create a folder `lab1` to keep the Python notebook and dataset for this lab.

1.3. Download the dataset for this lab from the VLE, and extract the files from the zip archive into the folder you just created. The files should now be in a `data` subfolder.

1.4. Create a Python notebook in the folder you created earlier. This will be used for all the workings in this lab, and also to generate the report.

2 Working with Real-World Audio Signals

2.1. To familiarise yourself with the handling of WAV files, let's start by loading the file titled `imperfect_sos_cw_700_15_wpm.wav`, using the code fragment below.

```
import os
from scipy.io import wavfile

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

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

Note that we use the `os.path.join()` function from the `os` package to concatenate the components of a file path in an OS-independent way. The last two lines create an interactive component that allows us to play the loaded data (in array `x`) with the given sampling rate (`fs`). Both of these were obtained from the WAV file.

2.2. We can plot the array values with the signal as read by SciPy using:

```
plt.figure()
plt.stem(x,markerfmt='b.')
plt.xlabel('Sample')
plt.ylabel('Value')
plt.title('Clean SOS')

plt.show()
```

Note that we're only passing the array `x` for the y-axis values. Matplotlib automatically plots this against the index of the value in the array. We can also see that each sample is read by SciPy as a signed 16-bit integer (with a value in $[-32\,768, 32\,767]$). The signal clearly has three short bursts, followed by three longer bursts, and then three more short bursts of higher-value samples. These represent the three dots, three dashes, and three dots of the Morse encoding for S.O.S.

2.3. We can also see that this is a fairly long array, with over 100 000 values. This makes it difficult to really see the detail of the sequence. We can focus on a shorter range, say between indices 2000–2400 (within the first burst) using:

```
plt.figure()
t = range(2000,2400)
plt.stem(t,x[t],markerfmt='b.')
plt.xlabel('Sample')
plt.ylabel('Value')
plt.title('Clean SOS')

plt.show()
```

At this scale we can see that the signal is approximately a sine wave.

3 Noisy Audio Signal

3.1. Repeat the steps of Section 2 to load the file `noisy_sos_cw_700_15_wpm.wav`. **[10 marks]**

3.2. Comment on this signal as compared to the one in Section 2. **[10 marks]**

4 Moving Average Filter

4.1. Noise can be reduced by taking a moving average of samples. Construct the impulse response h for a system that averages samples within a short (say, length $n = 5$) moving window. **[10 marks]**

4.2. Apply this system to the noisy signal x using convolution. This function is already available in the SciPy package, and can be used as follows:

```
from scipy import signal
y = signal.convolve(x,h)
```

4.3. Plot the filtered signal in a suitable way to compare with the previously plotted noisy signal. Listen to the noisy signal and the filtered signal. Comment on the effect of the filter in terms of what you see in the plot and what you hear in the playback. **[20 marks]**

4.4. Repeat the filtering process with a longer moving window. Comment on the effect of increasing the length n. **[10 marks]**

4.5. Experiment with longer moving windows, and comment on whether there is a limit to the length n that can be used. **[10 marks]**

5 Band-Pass Filter

5.1. Filter design, which is the subject of more advanced study, is the process of designing systems to satisfy a set of requirements, some of which may be conflicting. A suitably designed filter is provided for you in the form of an impulse response, which can be loaded from the NumPy file `filter.npy` using:

```
h = np.load(os.path.join('data','filter.npy'))
```

5.2. Plot the impulse response of this filter. **[10 marks]**

5.3. Apply this filter to the noisy signal, and plot the filtered signal in a suitable way to compare with the previously plotted noisy signal. Listen to the noisy signal and the filtered signal. Comment on the effect of this filter. **[20 marks]**