Systems and Signals 414 Practical 2: Using the DFT

Aim: Understand the use of the DFT in discrete-time signal analysis.

Handing in: Hand in via your `stnumber@sun.ac.za` email address. (To restate, use your student number email address! Do not use a special alias email address such as `johnsmith@sun.ac.za`.) Attach both your Jupyter `.ipynb` notebook file, and an `.html` copy of your output (`File` → `Download as` → `.html`). Email this to `sfstreicher+ss414@gmail.com` with subject as `prac2`. In summary, your email should look like this:

```
Recipient : sfstreicher+ss414@gmail.com
Subject   : prac2
Attachment: c:\...\filename.ipynb
Attachment: c:\...\filename.html
```

You may send your work multiple times; only the last submission will be marked. The process is automated (so we will not actually read the emails; i.e. trying to contact us using `sfstreicher+ss414@gmail.com` is futile). Make sure your `.ipynb` file returns your intend output when run from a clean slate!

Task: Do the following assignment using Jupyter. Document the task indicating your methodology, theoretical results, numerical results and discussions. Graphs should have labelled axes with the correct units indicated.

Hints: Plot "continuous-time" signals with Matplotlib's `plot` function, and discrete-time signals with `stem` instead. For example `pl.plot(np.arange(10))` and `pl.stem(np.arange(10))`. For zero-padding, the following function might serve usefull:

```
In [1]: import numpy as np
        def zpad(signal, pad_left_right):
            return np.pad(signal, pad_left_right,
                          mode='constant', constant_values=0)
```

```
In [2]: #For example
        zpad(np.array([1,2,3]), (2,3))

Out[2]: array([0, 0, 1, 2, 3, 0, 0, 0])
```

Additionally, here is some useful Jupyter preamble code:

```
In [2]: #All the necessary imports
        %matplotlib inline
        import pylab as pl
        pl.style.use('bmh') #pretty plots
        pl.rcParams['figure.figsize'] = (9, 2)
        import numpy as np

        #pl.figure()
        #pl.title(r"$f_s \ldots$ Hz") #Try not to have extremely long lines in your file, like this one pl
        #pl.ylabel("...")
        #pl.xlabel("...")
        #pl.stem(np.abs(np.arange(-5,5)));
```

Consider the following continuous-time signal $x(t)$:

$$x(t) = \cos(900 \cdot 2\pi t) + 0.15\cos(800 \cdot 2\pi t).$$

1. Sampling the continuous-time signal:

1. What frequency components are present in $x(t)$? Sketch (by hand) the magnitude spectrum $|X(f)|$ of $x(t)$, where the frequency axis is labelled in cycles/second (Hz).
2. Plot $x(t)$ for $0 \leq t < 0.02$, where t is time in seconds, so that you can see the shape of the continuous-time signal.
3. The discrete-time signal $x[n]$ is obtained by sampling $x(t)$ at a sampling frequency of $f_s = 2\,\text{kHz}$. Obtain the 40 samples of $x[n]$ for $n = 0, \ldots, 39$ and plot these on a graph where the horizontal axis shows discrete time n (samples). Does aliasing occur during sampling?

4. Again plot $x(t)$ for $0 \le t < 0.02$, but now also superimpose the 40 samples obtained in the previous question on this graph so that you are able to see the sampling instants clearly. Hint: Follow the same procedure as in Task 4 of Practical 1.
5. Use the Numpy function `np.fft` to calculate the DFT of $x[n]$. Plot the amplitude (magnitude) spectrum $|X[k]|$, using `np.abs`. Label your axes correctly!
6. Estimate the frequencies present in $x[n]$ directly from this amplitude spectrum. State the frequencies in cycles/sample, and then determine the corresponding frequencies in Hz using the (known) sampling frequency.
7. Zero-pad $x[n]$ by appending 160 zeros to the 40 samples you have taken. Now determine and plot the amplitude spectrum using the DFT. Explain what you see.

2. Obtain 50 **samples of** $x[n]$ **for** $n = 0, \ldots, 49$**:** That is, keep the sampling rate $f_s = 2\,\text{kHz}$ but sample the signal up to $t = 0.025\,\text{s}$.

1. Plot these samples.
2. Determine and plot the amplitude spectrum using the DFT.
3. Estimate the frequencies present in $x[n]$ directly from this amplitude spectrum. Why is it more difficult than before?
4. Zero-pad $x[n]$ by appending 150 zeros to the 50 samples you have taken. Now determine and plot the amplitude spectrum using the DFT. Are you better able to determine the frequencies present in $x[n]$?
5. Apply a Hamming window to the 50 samples of $x[n]$ with `np.hamming(50)`. Determine and plot the amplitude spectrum of this windowed signal. Explain the changes that have occurred.
6. Zero-pad the Hamming-windowed 50 samples by appending 150 zeros. Again, determine and plot the amplitude spectrum using the DFT. Are you better able to determine the frequencies present in $x[n]$? Explain why (or why not).

3. Now consider the continuous-time function

$$x(t) = cos(50 \cdot 2\pi t).$$

Obtain 40 samples $n = 0, \ldots, 39$ of the discrete-time signal $x[n]$ by sampling $x(t)$ at a sampling frequency of $fs = 2\,\text{kHz}$.

1. Determine the DFT $X[k]$ of these 40 samples, and plot its magnitude.
2. Zero-pad the DFT by inserting 160 zeros into the middle (i.e. between samples 20 and 21) of $X[k]$. In other words, zero-pad the spectrum, and not the time signal! Plot the magnitude of this new sequence.
3. Determine the IDFT of the above zero-padded DFT, using `np.fft.ifft` function. Plot the result. Note that there may be a (very small!) imaginary component after taking the IDFT due to round-off errors. Remove this by means of `np.real` function. Explain what you see.

4. Repeat the previous question, but now use 50 samples of $x[n]$ and insert 150 zeros during zero-padding of $X[k]$. Experiment with inserting the zeros into the DFT spectrum between samples $X[24]$ and $X[25]$, between samples $X[25]$ and $X[26]$, and half the zeros before sample $X[25]$ and half after it. Explain your observations.

5. (Optional) Investigate and plot the Fourrier pairs of tut test 1.