Fourier Transforms

Due at 2:00pm on Friday **Dec 3**, 2021

What you need to get

- `YOU_a5q4q5.ipynb`, and `Jinan.jpg`
- `YOU_a5q6.ipynb`, and `recording.wav`

What to do

1. **[5 marks]** Let $f = [-1,\ 2,\ 3,\ 4,\ 5,\ 2,\ 3,\ 4]^{\mathrm{T}}$. Use the FFT algorithm to compute the DFT of f. Show your work in a butterfly diagram.

2. **[9 marks]** Let $[F_0, ..., F_{N-1}]$ be the DFT of a sequence of data $[f_0, ..., f_{N-1}]$ where

$$F_k = \frac{1}{N}\sum_{n=0}^{N-1} f_n W^{-nk}$$

   and $W = e^{\frac{2\pi i}{N}}$ as usual.

   (a) Give a simplified expression for F_k when $f_n = W^{3n}$. For simplicity, assume $N > 3$.

   (b) Give a simplified expression for F_k when $f_n = (-1)^n$ for *even* values of N. (It may be useful to recall that $e^{\pi i} = -1$.)

   (c) Give a simplified expression for F_k when $f_n = (-1)^n$ for *odd* values of N.

3. **[5 marks]** Consider the *real, even* vector $f = [f_0, f_1, \ldots, f_{N-1}]$, in which $f_n \in \mathbb{R}$, and $f_n = f_{N-n}$ (analogous to an even function, in which $f(x) = f(-x)$). Given the DFT,

$$F_k = \frac{1}{N}\sum_{n=0}^{N-1} f_n \exp\left(\frac{-2\pi i n k}{N}\right) \quad \text{for } k = 0, \ldots, N-1\ ,$$

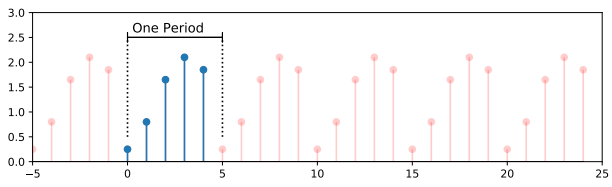
   prove that the vector of Fourier coefficients, $F = [F_0, F_1, \ldots, F_{N-1}]$, is also a **real and even** vector. Be sure to justify your steps.

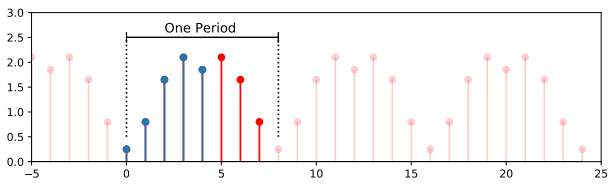
4. **[4 marks] Discrete Cosine Transform**

   The theorem you (hopefully) proved in question 3 can be summarized as,

   Theorem: The DFT of a real, even vector is also real and even.

   This property forms the basis of the Discrete Cosine Transform (DCT). The DCT is a special form of the DFT that makes some assumptions about its input. It operates on only real-valued input, and implicitly assumes that the input comes from an even vector. That is, unlike the DFT that implicitly assumes the vector is one period taken from a periodic signal (shown below on the left), the DCT assumes that the vector is *about half of one period* taken from a signal that is periodic **and** even (shown below on the right).





Periodic Extension Even Extension

A benefit of the DCT is that it produces real-valued Fourier coefficients. The other benefit is that the Fourier coefficients are even, so you only need to store about half of them.

One way to compute the DCT of a real-valued N-vector f is to perform an even extension of f – let's call it g. Then, since g is real and even, we can compute the DFT of g and extract only the first N Fourier coefficients. This also works in 2-D, 3-D, etc.

Complete the two functions `myDCT` and `myIDCT` (in the notebook `YOU_a5q4q5.ipynb`) that implement the DCT and its inverse for 2-D graylevel images. You should use the supplied functions `EvenExtension` and `IEvenExtension` to do (and undo) the even extensions. See the functions' help files for more details.

5. **Image Compression**

   In this question, you will use the DCT to perform JPEG-like image compression. The notebook `YOU_a5q4q5.ipynb` contains the function `myJPEGCompress`, which takes an image as input and is supposed to generate an output array containing DCT coefficients. The notebook also contains the function `myJPEGDecompress`, which is supposed to do the opposite; takes an array of DCT coefficients and generates an image from it.

   (a) **[4 marks]** Complete the function `myJPEGCompress`. The compression algorithm breaks the input image into tiles of size $T \times T$ (one of the inputs to `myJPEGCompress` is T, the tile size). Note that there might be leftover margins on the right and bottom edge of the image that are not part of the tiling. You can ignore those margins.

   Each tile is compressed by storing only a subset of the tile's DCT coefficients. Hence, the compression method performs lossy compression. Each $T \times T$ image tile is processed like this:

   i. Compute the DCT of the $T \times T$ tile.

   ii. Extract a $D \times D$ sub-array of low-frequency coefficients from the array of DCT coefficients (note that D must be smaller than T).

   iii. Save the $D \times D$ sub-array as the compressed representation of the tile.

   Do this for each tile in the image, and assemble the DCT blocks into another array; this array is your compressed representation of the image. You should use your `myDCT` and/or `myIDCT` to help you with this question. (If you had trouble with question 4, then you can use the versions commented out in the notebook.)

   (b) **[4 marks]** Complete the function `myJPEGDecompress`. It should take the compressed representation (and T and D) and undo the compression process to produce an approximation to the original image. For each $D \times D$ block,

   i. Embed the $D \times D$ array of DCT coefficients into a $T \times T$ array of zeros.

   ii. Compute the inverse DCT on the array.

   Assemble all the $T \times T$ tiles back into an image. Note that this compressed image might be a bit smaller than the original (if there were margins that were dropped). Again, use `myDCT` and/or `myIDCT` for this question.

   (c) **[2 marks]** The **compression ratio** is the number of pixels in your original image divided by the number of elements in your compressed representation, often expressed using "x:1" notation.

For example, suppose your original image is 120×120, and it is broken into tiles of size 10×10; there will be a 12×12 grid of such tiles. For each tile, only a 4×4 array of DCT coefficients are kept. Hence, the compressed representation would be 48×48 (that is, a 12×12 grid of 4×4). In this case, the compression ratio is 6.25:1, meaning the original image consists of 6.25 times as many numbers as the compressed representation.

Edit the notebook to demonstrate your compression and decompression functions. Compress and decompress the image `Jinan.jpg` (supplied) using a variety of D-values. Use a tile size of 10 for each. Try to achieve compression ratios close to 2:1, 4:1, and 25:1. Display each of the (de)compressed images. The title of each image should state the compression ratio for that image.

6. **[8 marks]** After unlocking Dennis Rillerson's smartphone, the police check his calendar app and find a suspicious appointment labelled "Report/handoff". The appointment seems odd because the police have established that Dr. Rillerson also has a dental patient during that time. This prompted the police to install covert listening devices throughout the dental clinic. Unfortunately, the recording captured during the suspicious meeting is contaminated by noise from the dental drill. The police would like to know what was said in the recording, but need the forensic specialists to help them isolate the conversation.

   Your job, as a forensic specialist, is to process the recording to isolate the speech, and convey your analysis in a report, as follows.

   The jupyter notebook `YOU_a5q6.ipynb` reads in the recording, constructs some useful variables, and plots the sound in the time domain. Edit the notebook to also do the following:

   (a) Plot the modulus of the Fourier coefficients in zero-shifted format (ie. with the DC in the middle, positive frequencies to the right, and negative frequencies to the left). Note that the variable `omega` is an array of frequencies corresponding to the Fourier coefficients.

   (b) Try to filter out the sound of the dentist's drill. To do this, choose a threshold frequency and set all Fourier coefficients above that frequency to zero. Plot your filtered Fourier coefficients (also in zero-shifted format).

   (c) Reconstruct the sound corresponding to the filtered signal. Plot the filtered sound (in the time domain).

   (d) What did Dr. Rillerson say? Write a transcript of what he said.

   Ensure that the plots are labelled appropriately, including a title and axis labels.

What to submit

For handwritten or typeset questions, you can:

- typeset your solutions using a word-processing application, such as Microsoft Word, LaTeX, Google docs, etc., or
- write your solutions using a tablet computer, or
- write your solutions on paper and take photographs.

In either case, it is **your responsibility** to make sure that your solutions are sufficiently legible. Crowdmark will accept PDFs or image files (JPEG or PNG). You may submit multiple files for each question, if needed.

For programming questions, you must submit your code in two places: on Crowdmark, and on D2L.

Crowdmark: We suggest you rename your jupyter notebooks, replacing "`YOU`" with your WatIAM ID. For example, `jorchard_a5q6.ipynb`. Export each jupyter notebook as a PDF, and submit the PDF to Crowdmark.

D2L: Submit any `.ipynb` files to Desire2Learn in the designated dropbox. You do not need to zip these files.