Fourier Theory

Due at 4:00pm on Tuesday 3 December 2019

What you need to get

- **https://www.overleaf.com/read/vztpkwrpmtcn**: (optional) Overleaf template for Q1, Q2, and Q3
- `YOU_a5q4q5.ipynb`: jupyter notebook for Q4 and Q5
- `Swiss_refuge.jpg`: image file for Q5

What to do

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

2. **[5 marks]** Let f be a real-valued signal, $f_n \in \mathbb{R}$ for $n = 0, \ldots, N-1$. Its Fourier coefficients are defined using the DFT,

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

   Define g to be a reflected version of f, so that $g_n = f_{N-n}$. If the Fourier coefficients of g are G_k, prove that

$$G_k = \overline{F_k}\ .$$

   Be sure to justify your steps.

3. **[3 marks]** Find the condition number of M (the DFT matrix) using the 1-norm. Show your work.

   For a complex matrix, the 1-norm is defined as

$$\|A\|_1 = \max_c \sum_r |A_{r,c}|$$

   where $|z|$ represents the modulus of the complex number z.

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

   Theorem 1 *Suppose that f is a* real, even *vector, $f = [f_0, f_1, \ldots, f_{N-1}]$, in which $f_n \in \mathbb{R}$, and $f_n = f_{N-n}$. Then, the vector of Fourier coefficients computed using the Discrete Fourier Transform (DFT) of f,*

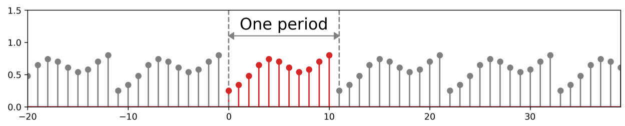
$$F_k = \sum_{n=0}^{N-1} f_n \exp\left(\frac{-2\pi i n k}{N}\right) \quad \text{for } k = 0, \ldots, N-1\ ,$$

   is also a real, even vector.

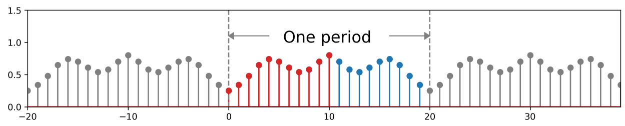
   In other words, the DFT of a real, even vector is also real and even. Perhaps you noticed that the signal in question 1 is a real, even vector.

   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 *one half of one period* taken from a signal that is periodic **and** even (shown below on the right).

© Jeff Orchard, Reinhold Burger 2019



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 half of them.

As it turns out, the DFT of an even vector g yields real-valued Fourier coefficients if g is real-valued. So, one way to compute the DCT of a real-valued N-vector f is to perform an even extension of f to get g, compute the DFT of g, then 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 generate 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.

   (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. 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. For each tile, only 4×4 DCT coefficients are kept. Hence, the compressed representation would be 48×48. 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 `Swiss_refuge.jpg` image (supplied) using a variety of D-values. Use a tile size of 20 for each. Try to achieve compression ratios close to 4:1, 25:1, and 100:1. Display each of the (de)compressed images. The title of each image should state the compression ratio for that image.

What to submit

You must submit a series of PDF documents to Crowdmark. For each coding question, export your jupyter notebook to PDF. When a proof or manual calculation is requested, you can typeset your solution (eg. using LaTeX or Word), or write your solution by hand. In either case, your solution should also be submitted as a PDF. **If you submit photos of handwritten work, it is your responsibility to ensure that they are legible.**