Due: October 26, 2018 # Homework 3 CPSC 475/BENG 475

For this assignment, parts 1 and 2 are **optional**, but are worth extra credit if you work through them (up to 5% if you complete both). Parts 3, 4, and 5 are worth 15%, 10%, and 75%, respectively.

Linear independence and basis

1 a Consider the vectors $\mathbf{u} = [1\ 1]^T$ and $\mathbf{v} = [3\ -1]^T$. What is the result of the expression $\alpha\mathbf{u} + \beta\mathbf{v}$ if $\alpha = 2$ and $\beta = 1.5$? Is this a linear combination? Why or why not?

— b Find the scalars α' and β' that make the following expression true: $\alpha'\mathbf{u} + \beta'\mathbf{v} = [4\ 8]^T$.

— c Express the equation above in matrix form and compute α' and β' using MATLAB. Hint: this involves creating a matrix from $\mathbf{u}$ and $\mathbf{v}$ and computing its inverse.

— d Now consider the following 2 vectors in $\mathbb{R}^3$: $\mathbf{x} = [1\ 1\ -1]^T$ and $\mathbf{y} = [1\ -1\ -1]^T$. Are they independent? Why?

— e Find scalars α and β such that $\alpha\mathbf{x} + \beta\mathbf{y} = [1\ 0\ 2]^T$. Do $\mathbf{x}$ and $\mathbf{y}$ form a basis for $\mathbb{R}^3$? Why or why not?

— f Consider the following 3 vectors: $\mathbf{x} = [1\ 0\ 0]^T$, $\mathbf{y} = [1\ -1\ -1]^T$ and $\mathbf{z} = [0\ 3\ -1]^T$ Are they orthogonal to each other? Do they form a basis for $\mathbb{R}^3$? Why?

— g Consider the column vectors in the 3×3 identity matrix. Do they form a basis for $\mathbb{R}^3$? Is this an orthogonal basis? Why? Express the vector $[2\ 1\ 7]^T$ as a linear combination of these column vectors — by which scalars do we need to multiply each of them?

— h Now consider the vectors $\mathbf{u} = [1\ 1]^T$ and $\mathbf{v} = [1\ -1]^T$. Do they form a basis for $\mathbb{R}^2$? If yes, can you make it into an ortho**normal** basis?

— i Consider the vector $\mathbf{x} = [-1\ 3]^T$ (which basis is it in, by the way?). Find the scalar coefficients α and β needed to express $\mathbf{x}$ in terms of the orthonormal basis from h. Hint: you can find the coefficients in the form of a vector $[\alpha\ \beta]^T$ using the same approach as in c!

Intro to Fourier series

2 a Compute $y_1 = \sin(t)$ using MATLAB and make a plot of the values in `y1`. Note that, in order to see the sinusoidal curve, t must be an array, `t`, in MATLAB. It should contain a range of values so as to include one complete period, i.e., $[0, 2\pi]$ (using `linspace` is a good idea). Choose a small step size between each value in `t` so that the curve appears relatively smooth (if you're using `linspace`, that is equivalent to choosing a large number of points).

— b Using the same `t` as above, compute $y_2 = \sin(2t)$ and plot it on top of `y1` (use the `hold on` command after plotting `y`). Now compute the result of `sum(y.*y2)`. Using a new figure, repeat this procedure for y_1 and $y_3 = \sin(3t)$. What do you notice about the results?

— c What is the dot product between `y1` and `y2`? Can we say $\sin(t)$ is orthogonal to $\sin(2t)$? Can `y1` and `y2` be seen as functions of t? Can they be seen as vectors? Explain. Repeat the experiment using each of the following two pairs: y_2 and y_3; y_1 and $\cos(t)$. Comment briefly on the results.

— d Using the same `t` as above, plot the function $w = 3\cos(5t) + 7\cos(15t) + 1.2\cos(30t)$.

— e Compute the dot product between w and the following vectors: y_1, y_2, and y_3 (from above), as well as $y_5 = \cos(5t)$, $y_{-5} = \cos(-5t)$, $y_{15} = \cos(15t)$, $y_{-15} = \cos(-15t)$, $y_{30} =$

$\cos(30t)$, and $y_{-30} = \cos(-30t)$. Always divide your results by n, the length of your vector `t`. What do you observe?

— f If you created a "basis" formed by the vectors above $(y_1, y_2, y_3, y_{-5}, y_5, y_{-15}, y_{15}, y_{-30}, y_{30})$, what would be the scalar coefficients needed to express w in terms of that basis?

— g Now, compute the discrete Fourier transform of your signal `w` using the function `fft`; use `fftshift` to translate the zero-frequency to the middle of the spectrum, followed by `abs` to compute the magnitude of the complex values obtained—you can do all of this using a single line of code: `W = abs(fftshift(fft(w)))`. Now divide `W` by `n` (the length of your `t` vector) and plot it against its corresponding frequency values, `f`—this is the vector containing the range of values `-n/2:n/2-1`, i.e., the maximum frequency will be half of the sampling rate of your discrete signal (does the name Nyquist come to mind?).

Execute `plot(f,W)` and zoom in to have a better look at the peaks in your plot: what do they represent? How do their frequency and magnitude values relate to the equation for w? How does this relate to what you did in f? Finally, what would you expect to happen to your frequency plot if we changed the coefficients 3, 7, and 1.2 in the equation for w to different values? What if we changed the frequencies 5, 15, and 30 to different values?

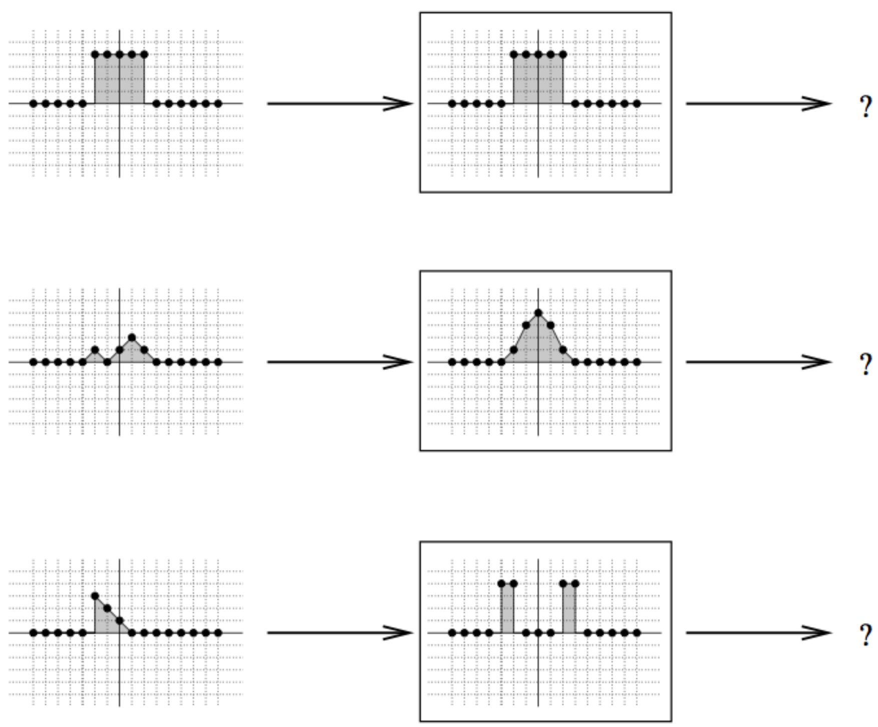
— h Now increase the number of periods in `t` to at least four (if T is the number of periods, your new range for the values in `t` should be $[0, 2\pi T]$). Let `w = sawtooth(t, 0.5)` (a triangular wave) and examine its plot: what can you say about this function? How is it different from the sum of sinusoidals you investigated above? Is it still periodic? Continuous? Smooth?

Compute its discrete Fourier transform and compare it with what you got above. In particular, what can you say about the frequency distribution? Note: this time, to the get correct frequency values, you must divide the frequency scale by number of periods chosen (T).

— i Finally, repeat this procedure for `w = sawtooth(t)` (a sawtooth wave). What is different now? Is it still periodic? Continuous? Smooth? What do you notice about its frequency distribution?

Discrete convolution in 1-D

3 a Inside the following boxes you are given the impulse response of a few linear systems. Manually (**without** MATLAB) compute the output of each system to the signal (left column). Assume the grids below represents integers centered at $(0, 0)$. Apply your computations only to the integer points, i.e. a discrete convolution. Input each answer into MATLAB and use the `stem` function to plot it (also use `axis equal` for proper scaling and `grid on` to add a grid). Note: you don't need to show all the steps of the computation, but make sure you understand them!



Fourier Transform and convolutions in 2-D

In this problem set, we will examine filters and their Fourier transforms. In class, we discussed an expansion of functions into a basis of sines and cosines. Recall from class that one could take the dot product of two functions f and g as follows:

$$\langle f, g \rangle = \int f(x)g(x)\, dx.$$

Therefore, we can think of expanding a function into sines and cosines much like expanding a vector onto an orthogonal basis. Let $\hat{f}_{\sin}(\omega) = \int f(x)\sin(\omega x)dx$ be the sine expansion of the function f and $\hat{f}_{\cos}(\omega) = \int f(x)\cos(\omega x)dx$ be the cosine expansion, where ω is the angular frequency. Then, using Euler's formula ($e^{i\theta} = \cos\theta + i\sin\theta$), the Fourier transform of the function f is:

$$\hat{f}(\omega) = \hat{f}_{\cos}(\omega) + i\hat{f}_{\sin}(\omega) = \int f(x)e^{-i\omega x}\, dx$$

(the negative sign in the exponent is simply a convention!).

4 a Do *one* of the following:

1. *Prove* that $\widehat{(f * g)}(\omega) = \hat{f}(\omega)\hat{g}(\omega)$ where $f * g$ is convolution. In other words, the Fourier transform of a convolution is the product of the Fourier transforms of the functions. Feel free to do this on paper and upload a scan or photo of your work.

2. *Demonstrate* using MATLAB that the Fourier transform of the convolution of two functions is the product of the Fourier transforms of the functions, i.e. $\widehat{(f * g)}(\omega) = \hat{f}(\omega)\hat{g}(\omega)$. For this problem you should do the following steps:

```
x = 0:.5:10;
f = x.*(10-x);
g = x.*(10-x).*(5-x);
```

   Now plot the real and complex parts of the *Fast Fourier Transform* (FFT) of the convolution (`conv`) of f and g (that is, $\widehat{(f * g)}$). Use two separate plots for that—one for the real part (use `real`) and another for the complex part (`imag`).

Then do the same thing for the elementwise product between the FFT of f and the FFT of g (that is, $\hat{f}(\omega)\hat{g}(\omega)$). Compare the two results. (Compute the FFTs by using `fft(function,50)` where 50 will pad the function with zeros to eliminate edge effects.)

5 a Write a function that creates a Gaussian blur filter (flesh out the code in `blurfilter.m`). Your function should create a matrix M (called `filt` in the code), where

$$M_{ij} = e^{-\frac{x_{ij}^2 + y_{ij}^2}{2\sigma^2}}.$$

Use the matrices `xs` and `ys` provided in `blurfilter.m` (e.g., `xs(i,j)` contains the x_{ij} component that should be used in the expression above). Normalize the sum of the entries to one by dividing by the sum of the entries. Why is this last step desired?

b Using `fft2`, compute the 2-dimensional Fourier transform of a filter created using your function from a with $\sigma = 1$, and display the absolute value of the transform (`abs` function) using `surf(⋯, 'EdgeColor', 'none')`. What (familiar function) does the Fourier transform of this filter look like? (Hint: Use `fftshift` to translate your transformed image in the ω_x and ω_y directions by half of the image width and height.)

The appearance of the filter can be improved by adding padding to your FFT by using `fft2(filter, pady, padx)` where `padx` and `pady` are larger than your filter dimensions.

c Load the Paolina image (remember to use `im2double`!). Filter Paolina with a blur filter having $\sigma = 3$ (use MATLAB's `filter2` function to do that) and show the result. **Compare** this with computing the inverse Fourier transform (`ifft2`) of the pointwise product between the Fourier transforms of Paolina and your filter (compute the transforms of the filter and the image, separately, then compute the inverse transform of the product of the two transforms). Be sure to pad the `fft2` of Paolina and your filter to the same size (larger than the image). In other words, use `fft2(image, padx, pady)` and `fft2(filter, pady, padx)`, where `padx` and `pady` are both larger than your image dimensions. How does this relate with what you did in part 4-b?

— d Using your function from a, build a 13x13 Difference of Gaussians filter, with $\sigma_1 = 1$ and $\sigma_2 = 2$. In other words, build a 13×13 matrix containing the pointwise difference of two Gaussian filters, the first having σ-parameter σ_1 and the second having σ-parameter σ_2, with $\sigma_2 > \sigma_1$. Normalize this filter so that the sum of its entries is zero, by subtracting the mean of the entries of this matrix. Apply this filter (using `filter2`) to `Loki.tiff` and `Steve.tiff`, displaying the result using `imshow(···, [])`. How does this filter affect features in the image? Compute the 2-dimensional Fourier transform of this filter. Why is this considered to be a band-pass filter? (Note: Once again you should use `fftshift` to shift the Fourier-transformed image.)

— e Flesh out the code in `unsharpmask.m` to create a function that performs unsharp masking on an input image. Unsharp masking an image consists of three steps:

1. Blurring an image;
2. Subtracting the blurred image from the original image to create the "mask";
3. Scaling the mask by some amount and adding it back to the original image.

Use the function to perform unsharp masking on `Paolina.tiff`, `Loki.tiff`, and `Steve.tiff`, with the parameters `sigma = 1` and `amount = 1` (although feel free to experiment!). Compare the filtered images to the original images (use `imshow(···)`, this time **without** the `[]`). Describe what the unsharp mask filter appears to be doing to the image.

— f Would it be possible to design an impulse response you could convolve against the image to perform unsharp masking in a single step? If yes, prove it by showing how to create such a filter; otherwise, explain why not.