EE113 Digital Signal Processing

Spring 2019

Homework 4

Due: Wednesday, May 8

Instructor: Prof. Christina Fragouli

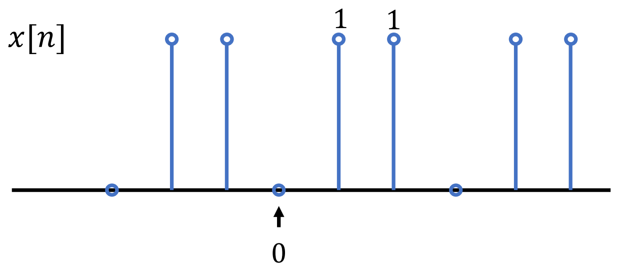
TA: Dhaivat Joshi and Joris Kenanian

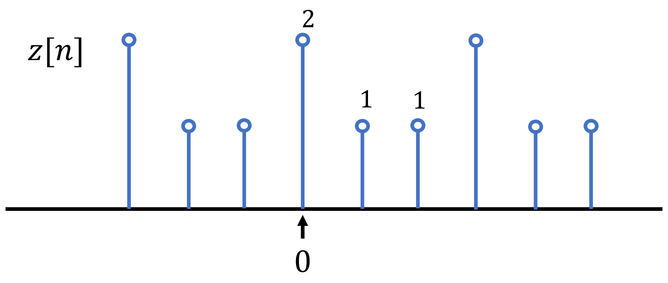
April 30, 2019

Total: 100 points

Last two problems require MATLAB. We thank Prof. Ali Sayed and Prof. Mert Pilanci for their materials on which these problems are based on.

Problem 1. ***(15 points)*** *Consider the following periodic signals $x[n]$ and $z[n]$, with period 3 each. Can you relate the DTFS coefficients of $x[n]$ with the DTFS coefficients of $z[n]$?*





(Hint: Try to find a signal $y[n]$ of period 3 such that $z[n] = x[n] \circledast y[n]$.)

Problem 2. ***(20 points)*** *Determine the DTFTs of the following sequences:*

(a). ***(10 points)*** $x[n] = \left(\frac{1}{2}\right)^{n-3} u[n] + \left(\frac{1}{3}\right)^{n} u[n-1]$.

(b). ***(10 points)*** $x[n] = \left(\frac{1}{4}\right)^{n-1} u[n] + \cos\left(\frac{\pi}{3} n\right)$.

Problem 3. *(a).* ***(10 points)*** *Determine the IDTFT of the following:*

$$X(\Omega) = \frac{\pi}{2} \cdot \operatorname{rect}\left(\frac{\Omega}{\frac{\pi}{6}}\right) \cdot e^{-j2\Omega}$$

where we define the rectangular function $\operatorname{rect}(\cdot)$ *as*

$$\operatorname{rect}\left(\frac{\Omega}{\Omega_c}\right) \triangleq \begin{cases} 1, & |\Omega| < \Omega_c \\ 0, & \Omega_c \leq |\Omega| \leq \pi \end{cases}$$

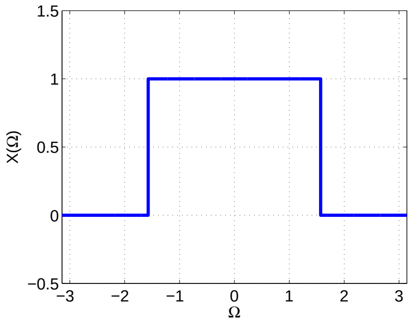
(b). ***(5 points)*** *Determine the following quantity for the same signal $x[n]$ in (a):*

$$\sum_{n=-\infty}^{\infty} |x[n]|^2$$

Problem 4. ***(25 points)*** *Consider the DTFT $X(\Omega)$ of a particular discrete-time signal $x[n]$.*

$$X(\Omega) = \begin{cases} 1, & \text{if} \quad |\Omega| < \omega_c \\ 0, & \text{else.} \end{cases}$$

For this problem, we let $\omega_c = \pi/2$ and the DTFT is plotted in the figure below.



(a). ***(5 points)*** *Use the inverse DTFT equation (or synthesis equation) and show that*

$$x[n] = \begin{cases} \frac{\omega_c}{\pi}, & \text{if} \quad n = 0 \\ \frac{\sin(\omega_c n)}{\pi n}, & \text{else.} \end{cases}$$

(b). ***(5 points)*** *Now, given the above $x[n]$, use the DTFT equation (or analysis equation) to get the DTFT of $x[n]$ as*

$$\tilde{X}(\Omega) = \frac{\omega_c}{\pi} + \sum_{\substack{n=-\infty \\ n\neq 0}}^{\infty} \frac{\sin(\omega_c n)}{\pi n} e^{-j\Omega n}.$$

At first sight, it it is not clear if the expression for $\tilde{X}(\Omega)$ is equal to that of $X(\Omega)$, but we will see that this is indeed the case using MATLAB.

(c). ***(10 points)*** *Consider $\tilde{X}(\Omega) = \frac{\omega_c}{\pi} + \sum_{\substack{n=-\infty \\ n\neq 0}}^{\infty} \frac{\sin(\omega_c n)}{\pi n} e^{-j\Omega n}$. We are unable to see what $\tilde{X}(\Omega)$ is because of the infinite number of terms in the summation. One way to get around this problem is by truncating the summation to a finite number of terms, i.e.:*

$$\tilde{X}(\Omega) \approx \frac{\omega_c}{\pi} + \sum_{\substack{n=-N \\ n\neq 0}}^{N} \frac{\sin(\omega_c n)}{\pi n} e^{-j\Omega n},$$

where N is an integer of our choice. On MATLAB, plot $\tilde{X}(\Omega)$ using the above approximation for $N = 1, 3, 5, 10, 20, 50$. Also plot $X(\Omega)$ in each of these plots for comparison. An example plot for $N = 7$ is shown in the figure below. You can use `subplot()` in MATLAB to plot these.

(d). ***(5 points)*** *As we increase N, what do you observe about $\tilde{X}(\Omega)$?*

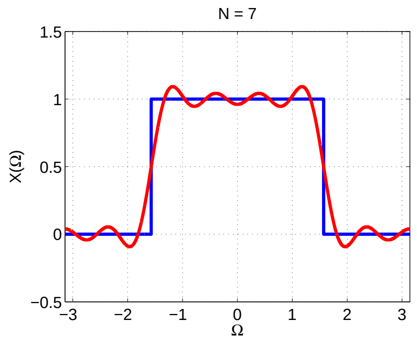


Figure 1: Plot of $\tilde{X}(\Omega)$ for $N = 7$ in red. $X(\Omega)$ is also plotted for comparison.

Problem 5. ***(25 points)*** *In this problem, we will see how a change of basis helps in image compression. From CCLE, download the bitmap image 'Lena.bmp'. On MATLAB, use the command*

```
>>A = double('Lena.bmp');
```

This reads the bitmap image and stores the color scale of each pixel in a matrix. Notice that the variable `A` *is now a* 512×512 *matrix. Each value of the matrix corresponds to the colorscale of the pixel in the image. To recover the image from the matrix, use the command*

```
>>imshow(uint8(A));
```

This should output the following image: So now, we can forget about the image and work with

the matrix. Note that we need $512 \times 512 = 262144$ *real values to represent this image. We will now try to reduce the number of real values needed to represent this image, albeit at the cost of the clarity of the image.*

At this point, we introduce something called the singular-value decomposition (SVD). Any matrix A *can be decomposed as* $A = USV^T$*, where the rows of* U *form an orthonormal basis for the rows of* A *and the columns of* V^T *form an orthonormal basis for the columns of* A*. In effect, we are representing the matrix* A *in a different orthonormal basis. Note that this is analagous to DTFS – there we represent a signal (or a vector) in a different orthonormal basis. To understand SVD further, you can check out any intermediate linear algebra course online. For this homework, this is all you need to know: SVD is essentially representing a matrix using a different orthonormal basis. The information about the bases are contained in matrices* U *and* V *and the representation of* A *in these bases is given by the matrix* S*, which is a diagonal matrix (this matrix has non-zero values* *only in its diagonal entries, it is zero everywhere else). The diagonal entries of S are called the "singular values" of A.*

The tasks:

(a). ***(20 points)*** *On MATLAB, use the command*

```
>>[U,S,V]=svd(A)
```

to get the singular value decomposition of the image matrix. Extract the diagonal entries of S and store it in an array `singvals`*. The array* `singvals` *now contains the singular values in decreasing order. Now, from the array* `singvals`*, extract the indices of those singular values which are at least* 0.01 *of the largest singular value. You can use the MATLAB command*

```
>>indices = find(singvals >=0.01*max(singvals))
```

Now construct new matrices `U_red,S_red,V_red` *as follows:*

- `U_red` *is a matrix which contains only the columns of* `U` *corresponding to the indices obtained above. For example, if* `indices=[1,4,5]`*, then* `U_red` *is a matrix with only 3 columns, where the first column of* `U_red` *is the first column of* `U`*, the second column of* `U_red` *is the fourth column of* `U`*, and the third column of* `U_red` *is the fifth column of* `U`*.*
- `S_red` *is a diagonal matrix corresponding to the indices obtained above. For example, if* `indices=[1,4,5]`*, then* `S_red` *is a* 3×3 *matrix with the 3 diagonal entries as the first, fourth and fifth diagonal entries of S.*
- `V_red` *is a matrix which contains only the columns of* `V` *corresponding to the indices obtained above. This is similar to constructing* `U_red` *from* `U`*.*

Now construct an approximation of A using matrices `U_red,S_red,V_red` *as follows:*

```
>>A_red = U_red * S_red * V_red';
```

Use `imshow()` *to display the image corresponding to A_red.*

Attach the MATLAB script you wrote above with your submission, and also the image you obtained with** A_red**.

(b). ***(5 points)*** *Question: We saw earlier that we need* 262144 *real values to represent A. Similarly, we need 262144 real values to represent A_red as well. Instead, note that* `U_red,S_red,V_red` *contain all the information needed to construct A_red. How many real values do we need to store* `U_red,S_red,V_red`*? How much better is this compared to storing A?*