Applications of the Discrete Fourier Transform

Application 1 – Signal Detection

What is the given data?	
A noise contaminated signal x[n] = s[n] + w[n]	
What are we doing?	**How?**
Trying to identify the dominant frequencies (aka ***harmonics***) of s[n]	- By ***direct observation of the magnitude spectrum*** of x[n]
	- By a technique called ***DFT averaging***

Steps:

1. Download and load into the MATLAB workspace the signal xn **(username ELEC459 password as given in class by course instructor)**

2. Use MATLAB to **generate and record** (that is, **save** the plot of) time and frequency domain representations of x[n].

   Include the plots in your report. Correctly label the axes. Please scale the frequency axis to the interval of interest: either [0, fs/2] – preferred; or [0,fs].

   Helpful MATLAB functions: **plot**, **axis, label, fft, abs**.
   What representation reveals more information about the underlying signal's harmonics?

   Identify and state the value of the harmonics.

3. Construct subsets of **xn**, consisting of the first 128, 256, 512, 1024 and 1792 samples.

   Generate and include in your report the magnitude spectrum of each subset. **Identify and include in your report** the values of the harmonics.

   Helpful MATLAB functions: **fft, abs**, **subplot**, and the **:** operator.

   Comment on the influence of the number of signal samples on the quality of the frequency representation.

4. Implement the DFT averaging **method** – possible implementation idea:
   a. generate a matrix with K rows and L columns in which
      each column is a sequence of K consecutive samples of **xn**.
   b. compute the DFT of the matrix.
   c. *compute the average of all **columns** in the DFT matrix.*
   d. **plot** the magnitude of the resulting vector.

5. **Identify and state** the values of the dominant harmonics (spikes), when K = 128 and L = 14. Helpful in identification: once you plot the magnitude spectrum, you can use the **Data Cursor**; the ***x*-coordinate** is the frequency of interest.

6. Change the value of L to **identify and include in your report** the smallest value of L for which the harmonics remain clearly visible. Include representative plots to justify your answer.

7. **State** whether other values of K such as K = 100 or K = 135 can be used. Please explain/ include some representative plots to justify your answer.

Application 2: Signal interpolation

Setup:	
Given some discrete time values of a signal	
What are we doing?	**How?**
Trying to identify the true signal	By zero insertion in the frequency domain (a method of guessing the values of the missing samples, aka signal interpolation)

Steps:

1. Define variable N = 20,000.
2. Load the signal **handel** into MATLAB (by typing in the Command Window or in a MATLAB script **load handel**);
3. Denote the first N samples of **y** by **x:** **x = y(1:N);**
4. Downsample **x** to generate vectors **x2, x3, x4** e.g., **x2 = x(1:2:N);**

In the steps described below, xd is a placeholder for x2, x3 or x4. It is recommended you save Steps 5-8 as a MATLAB function with input arguments xd and K, such that you can call it with input parameters x2 and 1 for part 2.4.6 (a), x3 and 2, for (b) and x4 and 3 for (c). The output of this function should be x_inter

---

5. Compute the DFT of **xd** and store it as variable **Xd** – helpful MATLAB function: **fft**.
6. Denote the length of **Xd** by **Nd** and obtain **Xe** by inserting ***K·Nd*** zeros in the middle of **Xd.**
7. Inserting the ***K·Nd*** zeros should be done taking into account the parity of ***Nd*** – helpful MATLAB function: **mod**.

   If **Nd** is **odd**

   ```
   N1 = (Nd+1)/2;
   Xe = [Xd(1:N1);  zeros(K*Nd,1);  Xd((N1+1):Nd)];
   ```

   If **Nd** is **even**

   ```
   N2 = Nd/2;
   Xe = [Xd(1:N2); Xd(N2+1)/2; zeros(K*Nd-1, 1); Xd(N2+1)/2; Xd((N2+2):Nd)];
   ```
8. Obtain the time domain representation of Xe, by computing its inverse Discrete Fourier transform **ifft**
9. Rescale the amplitude of the signal obtained at step 7, by multiplying it by K+1. Name this signal **x_inter**

---

10. Call the function written at steps 5 – 8, with input arguments x2 and K = 1. Name your output x_inter2. Make sure that the signals x and x_inter2 have the same length, by cropping the last samples of the longer signal.
11. **Display in the same figure <u>the first 50 samples</u>** of the original signal **x** and the interpolated signal. Use **different colors and/ or markers**.

    Sample code:

    ```
    figure;
    plot(x(1:50),'bo');     % Displays signal samples using blue circles
    hold on                 % Holds the plot in the current figure
    plot(x_inter2(1:50),'rx'); % Displays signal samples using red crosses
    legend('Original','Interpolated') % Inserts a legend in the plot
    ```

12. Compute and include in your report the norm of the difference between the original signal x and its interpolated version, using the MATLAB function **norm**.
13. Repeat steps 10 – 12 for x3 and K = 2
14. Repeat steps 10 – 12 for x4 and K = 3
15. Comment on the different values obtained for the three interpolated signals. Which of the three interpolated signals is closest to the true signal x? [Hint: A low value of norm(a – b) indicates that signal a is almost equal to b, sample-wise].

Marking scheme

Lab work	Points
2.4.3. plots of all dfts	1.75
2.4.4. k 128 L 14	1.75
2.4.4. smallest L	1.75
2.4.4. other Ks	1.75
2.4.6.a plot	0.5
norm	0.5
2.4.6.b plot	0.5
norm	0.5
2.4.6.c plot	0.5
norm	0.5
TOTAL	**10**

Technical content	Points
2.4.3. plots of all dfts	1.75
2.4.4. k 128 L 14	1.75
2.4.4. smallest L	1.75
2.4.4. other Ks	1.75
2.4.6.a plot	0.5
norm	0.5
2.4.6.b plot	0.5
norm	0.5
2.4.6.c plot	0.5
norm	0.5
TOTAL	**10**

Presentation	Points
front page	2
structure	2
labels on figures	2
legible figures	2
spelling/ grammar	2
TOTAL	**10**

Overall mark: Lab work × 40% + Technical content × 40% + Presentation × 20%
Attendance is needed for mark to be computed.
Report due: Jan. 31, by 4PM
Late submissions: 10% of your mark is deducted per day.