EEE 424 Coding Assignment 1 Spring 2024-25

Due-date: 3 April 2025, 23:59

This assignment focuses on the computation of Discrete Fourier Transform (DFT) through different methods. You will use MATLAB in this assignment. You should provide the MATLAB code you have written in each question, and provide the results of your code to each part of the assignment.
At each stage, while computing the DFT, make sure to measure the time it takes to execute the implementation of the DFT. You can use the *tic* and *toc* functions within MATLAB to measure the time it takes to compute the DFT.
You should include all the work you have done, including your answers to each question and the code you have written, in a single pdf file. Please do not submit other file types, such as .m, .docx, .txt etc.

Q1) In this question, you will compute the DFT of vectors using different approaches. You will work with vectors of different lengths. It is recommended that you write your code in modular/functional forms so that you can repeat your work for different array lengths easily.

First, you will create a complex array of length $N = 32$. Please paste the following code on top of your MATLAB file to construct this array:

```
rng(2,"twister")
N = 32;
real_part = randn(1,N);
imag_part = randn(1,N);
x = real_part + 1i*imag_part
```

Now, you have a 32-pt complex array in the variable **x**.

(a) Plot the real and imaginary parts of $\mathbf{x}[\mathbf{n}]$.

(b) Compute the 32-pt DFT of the array $\mathbf{x}[\mathbf{n}]$ in MATLAB by directly using its definition in summation form.

Note: *You should not use fft or any related command of MATLAB in this part. You are asked to directly implement the DFT definition using its mathematical definition.*

(c) Create the DFT matrix in MATLAB without using any specialized commands, directly by using its mathematical definition.

Note: *You should not use fft or any related command of MATLAB in this part. You are asked to directly compute the DFT matrix.*

(d) Using the DFT matrix you have constructed, compute the DFT of the vector $\mathbf{x}[\mathbf{n}]$.

(e) Now, implement the Fast Fourier Transform (FFT) using the decimation-in-time algorithm. Find the FFT of the vector $\mathbf{x}[\mathbf{n}]$ using the decimation-in-time algorithm.

(f) Then, implement the FFT using the decimation-in-frequency algorithm. Find the FFT of the vector $\mathbf{x}[\mathbf{n}]$ using the decimation-in-frequency algorithm.

Note: *For parts (e) and (f), you should not use fft or any related built-in MATLAB command. You are asked to implement the two algorithms without using any related built-in commands and compute the DFT of* $\mathbf{x}[\mathbf{n}]$ *using the functions you have implemented.*

(g) Finally, you can use the built-in MATLAB command for the FFT to compute the DFT of the vector $\mathbf{x}[\mathbf{n}]$.

(h) Compare your results in parts (b), (d), (e), (f) and (g). You can use the subplot command of MATLAB to compare all six approaches. You should plot the magnitude and phase plots produced by each method and compare your findings. **Please make sure that all of your plots and findings are clearly visible in the pdf file.** Show that all of the plots are equivalent. Also, show the equivalence of these methods numerically. If they are not equivalent, you are making a mistake in your computations of the DFT and you should re-check your implementation.

Hint: *To show the numeric equivalence between the methods, you can show that the difference of the norms between the arrays you have obtained is very small (close to machine precision), i.e.,*

$$\text{norm}(y_{\text{method}_1} - y_{\text{method}_2}) < \epsilon,$$

for some infinitesimally small ϵ.

	$N = 32$	$N = 256$	$N = 2^{12}$
Part (b)			
Part (d)			
Part (e)			
Part (f)			
Part (g)			

Table 1: Time it takes to compute different DFT approaches on signals of varying length.

(i) Repeat steps (a)-(h) for a vector of length $N = 256$. You should first create a new vector by using the following script:

```
rng(2,"twister")
N = 256;
real_part = randn(1,N);
imag_part = randn(1,N);
x = real_part + 1i*imag_part
```

(j) Repeat steps (a)-(h) for a vector of length $N = 2^{12}$. You should first create a new vector by using the following script:

```
rng(2,"twister")
N = 2^12;
real_part = randn(1,N);
imag_part = randn(1,N);
x = real_part + 1i*imag_part
```

(k) Compare the efficiencies of each approach for different signal lengths in terms of the time it takes to compute the DFTs at each step. Present your results in a table. Comment on whether your findings align with the theoretical costs. A sample table format is included in Table 1. Make sure to include all approaches and all three signal lengths.

Q2) Up to now, you have mainly focused on using the FFT on arrays of length $N = 2^v$. We will focus on an alternative scenario in this question where $N = 3^v$, and work with a 9-pt signal array.

This question has an analytical part at its beginning.

(a) Derive an algorithm for the $N = 3^v$-pt FFT using decimation-in-frequency. Clearly show the steps in your derivation.

(b) You are provided with a sample flow graph for the 9-pt FFT operation below. The entire operation is segmented into multiple 3-pt DFTs. Based on your derivations in (a), construct the flow graph for a 9-pt decimation-in-frequency FFT algorithm. You can consider the sample below and try to fill in the missing parts of the flow graph. You can connect the input variables to corresponding nodes at the first 3-pt DFT layers and fill in the empty parts of the plot with appropriate variables and outputs. Question marks are added to indicate some of the points and connections where you have to label the corresponding variable and the output.

Hint: *A label assigned to a connection means that the signal is multiplied with the label while passing through that link.*

(c) Compute the 9-pt DFT of a signal using the DFT definition in summation form. Again, you will need to construct a 9-pt array for the MATLAB questions. Please paste the following code snippet on top of your MATLAB code file:

```
rng(2,"twister")
N = 9;
real_part = randn(1,N);
imag_part = randn(1,N);
x = real_part + 1i*imag_part
```

Hint: *You should not use fft or any related command of MATLAB in this part. You are asked to directly implement the DFT definition using its mathematical definition.*

(d) Implement the algorithm you have derived in (a) for the 9-pt decimation-in-frequency FFT operation in MATLAB. Using the algorithm, find the FFT of $\mathbf{x}[\mathbf{n}]$.

(e) Now, to assess that your implementations are correct, compute the 9-pt FFT of the vector using MATLAB's built-in fft command.

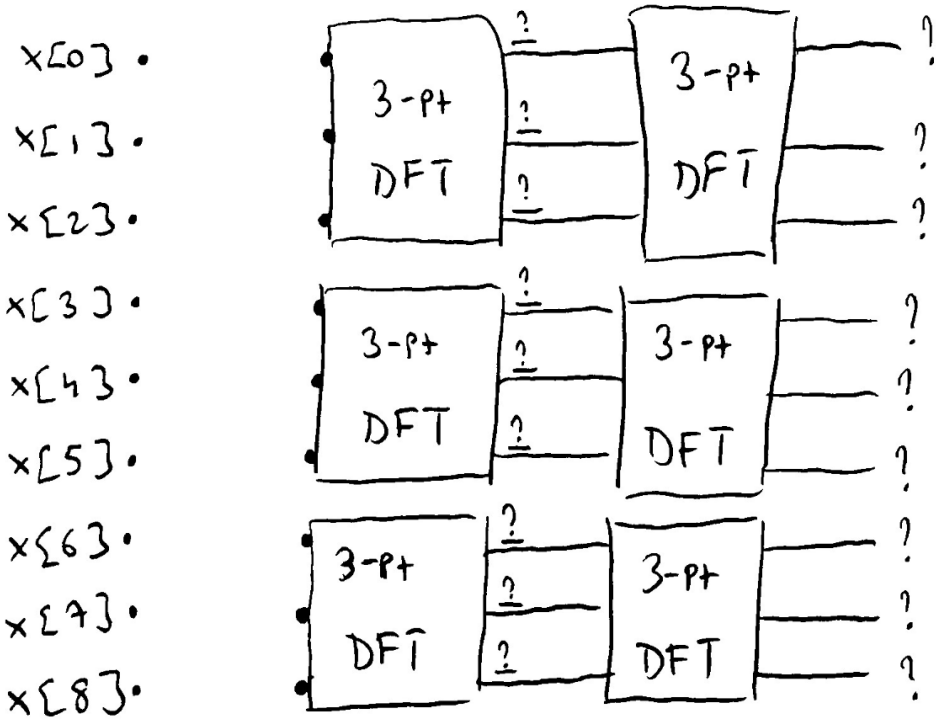

Figure 1: Sample flow graph for the 9-pt decimation-in-frequency FFT algorithm.

(f) Show the equivalence of your implementations in parts (c), (d), and (e) numerically. You can compute the norm of the difference as in **Q1)**(i). If the difference is not too small, you should re-check your implementation.

(g) Compare the efficiencies of the three approaches you have followed to compute the DFT of the 9-pt vector based on the time it takes to execute them. Discuss whether your findings align with your expectations.

Hint: *Your observations from* ***Q1)****(k) can be useful for your discussion in this part.*