TENNESSEE TECHNOLOGICAL UNIVERSITY

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

Solution of Assignment/Homework-2

Topics- Discrete Fourier Transform (DFT) and Discrete Cosine Transform (DCT)

Course No: ECE-6040 (Spring 2023)

Course Title: Signal Analysis

Date of Submission: Feb-9-2022 (10.59 PM)

SUBMITTED BY

Sohag Kumar Saha
Graduate Teaching Assistant, ECE
T Number: T-00347416
Email: ssaha42@tntech.edu

SUBMITTED TO

Dr. Corey Cooke
ORNL/TTU Joint Faculty
Electrical and Computer Engineering
Email: ccooke@tntech.edu

Homework 2

Question of Problem 1 (30 pts)

Given

The discrete Fourier transform (DFT) is an orthogonal transform defined for vectors in $\mathbb{C}^N$. The normalized version of the DFT uses the following N basis functions:

$$b_k[n] = \frac{1}{\sqrt{N}} e^{\frac{j2\pi kn}{N}}, \qquad n, k \in \{0, 1, 2, \dots, N-1\}$$

where n represents the timestep and k denotes the k^{th} basis function.

Problem Statement

Analytically prove that this set of functions form an orthonormal basis for $\mathbb{C}^N$.

Hints

- To prove orthonormality, you need to prove:
  - $\|b_k[n]\| = 1$ for all k
  - $\langle b_k[n], b_l[n] \rangle = 0$ for all $k \neq l$

- You may need the following identities:
  - $e^{j2\pi k} = 1$ for any $k \in \mathbb{Z}$
    - since $e^{j2\pi k} = \left(e^{j2\pi}\right)^k = (\cos 2\pi + j \sin 2\pi)^k = (1 + j0)^k = 1$
  - $\sum_{n=0}^{N-1} ar^n = \frac{a(1-r^N)}{(1-r)}$

Solution of Problem-1:

The solution of problem-1 is added in the next page. I have added the scanned copy of the handwritten solution of the first problem.

Solution of Prob-1 (HW: 2)

Given that,

the discrete Fourier transform (DFT) is an orthogonal transform defined for vectors in $\mathbb{C}^N$. The normalized version of the DFT uses the following N basis functions:

$$b_K[n] = \frac{1}{\sqrt{N}} e^{\frac{j2\pi Kn}{N}}, \quad n, K \in \{0, 1, 2, \cdots, N-1\}.$$

n = the timestep & K = K^{th} basis function.

Need to provide the proof that, this set of functions form an orthonormal basis for $\mathbb{C}^N$.

Solution:

The set of functions:

$$b_K[n] = \frac{1}{\sqrt{N}} e^{\frac{j2\pi Kn}{N}}; \quad n, K \in \{0, 1, 2, \cdots, N-1\}$$

should forms an orthonormal basis for $\mathbb{C}^N$, in case of the successful satisfaction of following properties as shown in the ~~following~~ below:

It is required to proof the normalization and orthogonality property.

(P.T.O)

Normalization: Each basis function is normalized in such that the magnitude of its inner product with itself is equal to 1, that is:

$$\|b_k[n]\| = 1 \quad \text{for all } k.$$

Now: $\|b_k[n]\|^2$

$$= \langle b_k[n] \cdot b_k[n] \rangle$$

$$= \langle b_k[n] \cdot b_k^*[n] \rangle$$

$$= \sum_{n=0}^{N-1} \langle b_k[n] \cdot b_k^*[n] \rangle$$

$$= \sum_{n=0}^{N-1} \left(\frac{1}{\sqrt{N}} e^{\frac{j2\pi kn}{N}} \cdot \frac{1}{\sqrt{N}} e^{-\frac{j2\pi kn}{N}} \right)$$

$$= \sum_{n=0}^{N-1} \left(\frac{1}{N} \cdot e^{j2\pi(k-k)n/N} \right)$$

$$= \sum_{n=0}^{N-1} \frac{1}{N} \cdot 1 = \frac{1}{N} \cdot N = 1.$$

Therefore $\|b_k[n]\| = 1$ for all k.

Orthogonality: for any two basis functions $b_k[n]$ and $b_l[n]$, the inner product of both the function or, dot product of both the function is zero; if $k \neq l$: that is:

$$\langle b_k[n], b_l[n] \rangle = 0 \quad \text{for all} \quad k \neq l.$$

$$\therefore \langle b_k[n], b_l[n] \rangle$$

$$= \langle b_k[n] \cdot b_l^*[n] \rangle$$

$$= \sum_{n=0}^{N-1} \left(\frac{1}{\sqrt{N}} e^{\frac{j2\pi kn}{N}} \right) \left(\frac{1}{\sqrt{N}} e^{\frac{-j2\pi ln}{N}} \right)$$

$$= \sum_{n=0}^{N-1} \frac{1}{N} e^{\frac{j2\pi (k-l)n}{N}}.$$

In case of $k = l$, we get

$$\sum_{n=0}^{N-1} \frac{1}{N} e^{j2\pi(k-k)n/N} = \sum_{n=0}^{N-1} \frac{1}{N} \cdot e^{0}.$$

$$= \sum_{n=0}^{N-1} \frac{1}{N} \cdot 1 = \frac{1}{N} \cdot N = 1. \quad \text{(P.T.O)}$$

When the value of k and l is not equal, then we get:

$$\langle b_k[n], b_l[n] \rangle = \langle b_k[n] * b_l^*[n] \rangle$$

$$= \sum_{n=0}^{N-1} \frac{1}{N} e^{j2\pi(k-l)n/N}$$

$$= \sum_{n=0}^{N-1} \frac{1}{N} \left\{ e^{\frac{j2\pi}{N} \cdot (k-l)n} \right\}$$

$$= \sum_{n=0}^{N-1} \frac{1}{N} \left\{ e^{\left(\frac{j2\pi}{N}\right)} \right\}^{(k-l)n}$$

$$= \sum_{n=0}^{N-1} \frac{1}{N} \left\{ \frac{1 - \left\{ e^{\frac{j2\pi}{N}} \right\}^{(k-l)N}}{1 - \left\{ e^{\frac{j2\pi}{N}} \right\}^{(k-l)}} \right\}$$

$$= \sum_{n=0}^{N-1} \frac{1}{N} \left\{ \frac{1 - \left\{ e^{j2\pi} \right\}^{(k-l)}}{1 - \left\{ e^{\frac{j2\pi}{N}} \right\}^{(k-l)}} \right\}$$

$$= \sum_{n=0}^{N-1} \frac{1}{N} \left\{ \frac{1 - \left\{ e^{j2\pi} \right\}^{(k-l)}}{1 - \left\{ e^{\frac{j2\pi}{N}} \right\}^{(k-l)}} \right\}$$

(P.T.O)

$$= \sum_{n=0}^{N-1} \frac{1}{N} \left\{ \frac{1 - \left(e^{j2\pi K} / e^{j2\pi l} \right)}{1 - \left(e^{\frac{j2\pi K}{N}} / e^{\frac{j2\pi l}{N}} \right)} \right\}.$$

$\Rightarrow$ We have $e^{j2\pi K} = 1$ for any $K \in Z$

$$\left(e^{j2\pi}\right)^K = \left(\cos 2\pi + j \sin 2\pi\right)^K$$

$$= (1 + j0)^K$$

$$= 1.$$

So, $\langle b_K[n], b_l[n] \rangle$

$$= \langle b_K[n] * b_l^*[n] \rangle$$

$$= \sum_{n=0}^{N-1} \frac{1}{N} \left\{ \frac{1 - \left(e^{j2\pi K} / e^{j2\pi l} \right)}{1 - \left(e^{j2\pi K} / e^{j2\pi l} \right)^{1/N}} \right\}.$$

$$= \sum_{n=0}^{N-1} \frac{1}{N} \times 0 = 0$$

Therefore, it is proved that, the inner product of two basis functions is zero. if $K \neq l$.

(P.T.O)

As both the normalization and orthogonality is proved, therefore we can say that, this set of functions form an ~~orthogona~~ orthonormal basis for $\mathbb{C}^N$. (Proved)

Question of Problem 2 (30 pts)

Given

The discrete cosine transform (DCT), which is defined for $\mathbb{R}^N$, uses the following basis:

$$b_k[n] = \begin{cases} \dfrac{1}{\sqrt{N}} & k = 0 \\ \sqrt{\dfrac{2}{N}} \cos\left[\dfrac{\pi}{N}\left(n + \dfrac{1}{2}\right)k\right] & k = 1, 2, \dots, N-1 \end{cases}$$

for $n \in \{0, 1, \dots, N-1\}$.

Proving orthonormality for this set of functions analytically requires some "tricks" that are probably beyond the scope of this course, so we will approximately prove it using numerical methods for a few values of N.

Problem Statement

Compute the inner product between each element of the basis set $\{b_k[n]\}_{k=0}^{N-1}$ for $N = 16, 64, 128$ numerically by writing a Python/NumPy program. You should print each inner product to the screen such that your program output (in the terminal) looks something like this:

```
N = 16:
k = 0, l = 0, inner_product = (print your inner product value here)
k = 1, l = 0, inner_product = (print your inner product value here)
k = 2, l = 0, inner_product = (print your inner product value here)
k = 3, l = 0, inner_product = (print your inner product value here)
⋮
⋮
⋮
k = 15, l = 15, inner_product = (print your inner product value here)

N = 64:
k = 0, l = 0, inner_product = (print your inner product value here)
k = 1, l = 0, inner_product = (print your inner product value here)
k = 2, l = 0, inner_product = (print your inner product value here)
k = 3, l = 0, inner_product = (print your inner product value here)
⋮
⋮
⋮
k = 63, l = 63, inner_product = (print your inner product value here)

N = 128:
k = 0, l = 0, inner_product = (print your inner product value here)
k = 1, l = 0, inner_product = (print your inner product value here)
k = 2, l = 0, inner_product = (print your inner product value here)
k = 3, l = 0, inner_product = (print your inner product value here)
```

⋮
⋮
⋮
`k = 127, l = 127, inner_product = (print your inner product value here)`

Copy and paste the terminal output into a text file and include it as a separate file from your report. Alternatively, you could just write the output to a file if that is easier for you than copying and pasting. (Here is a nice tutorial on file IO in Python).

Do these results make sense? Why or why not? What *should* each inner product be? (i.e. what should the inner product be when k == 1, and what should it be when k != 1?)

Hints

- In Python, use the print() function to print text and numbers to the screen.
- You can use a C-like syntax to format numerical values inside string arguments for display, e.g.
    - `print("The current year is %d. It is %g degrees Fahrenheit outside."%(2023, 54.3))`

Solution of Problem-2:

The inner product between each element of the basis set $\{b_k[n]\}_{k=0}^{N-1}$ for $N = 16, 64, 128$ is calculated numerically by writing a Python program using Numpy library. The code of problem-2 is attached in the Teams Assignment submission window and also in the Appendix of this solution After running the program, it should print each inner product as required in the question. The text file is also added in which I saved the output of the program by showing the inner product values by varing N, k and l.

Comments on Results:

Do these results make sense? Why or why not?

- Yes the results seems satisfactory as it meets the condition of the given discrete cosine transform (DCT) function for $\mathbb{R}^N$. The output of the inner function is 1 when both k and l is equal, whereas the output of the inner function is 0, when both k and l have different values (i.e. $k \neq l$).

What should each inner product be? (i.e. what should the inner product be when k == l, and what should it be when k != l?)

- The inner product should be either 0 or 1. When k==l, the inner product should be 1 and when k != l, the inner product should be 0.

Question of Problem 3 (40 pts)

Given

$$x[n] = 0.1n, \quad n = \{0, 1, \dots, 7\}$$

We will denote the DFT of $x[n]$ as $X_f[k]$ and the DCT of $x[n]$ as $X_c[k]$.

Problem Statement

Using the inner product with the elements of the orthonormal basis set, write a Python program to compute both the DFT and the DCT of $x[n]$. Each transform should return a vector that is $N = 8$ elements long. In the case of the DFT, the output vector will be in $\mathbb{C}^N$ and the output of the DCT will be a vector in $\mathbb{R}^N$.

Produce a plot of $|X_f[k]|$ for $k \in \{0, 1, \dots 7\}$ and a separate plot of $|X_c[k]|$ for $k \in \{0, 1, \dots 7\}$. Use the `plt.stem()` command instead of `plt.plot()` because `stem` makes the discrete nature of the spectrum more clear. Due to Parseval's theorem, we can interpret both $|X_f[k]|^2$ and $|X_c[k]|^2$ as the signal energy contained in the k^{th} frequency bin for each transform. Which transform does a better job of concentrating the energy into a fewer number of bins?

Solution of Problem-3:

The plot of $|X_f[k]|$ for $k \in \{0, 1, \dots 7\}$ and another plot of $|X_c[k]|$ for $k \in \{0, 1, \dots 7\}$ is shown in below for the given DFT and DCT equation respectively with $x[n] = 0.1n, \quad n = \{0, 1, \dots, 7\}$. The python code is added in the Teams assignment submission window and also in the appendix of this solution.

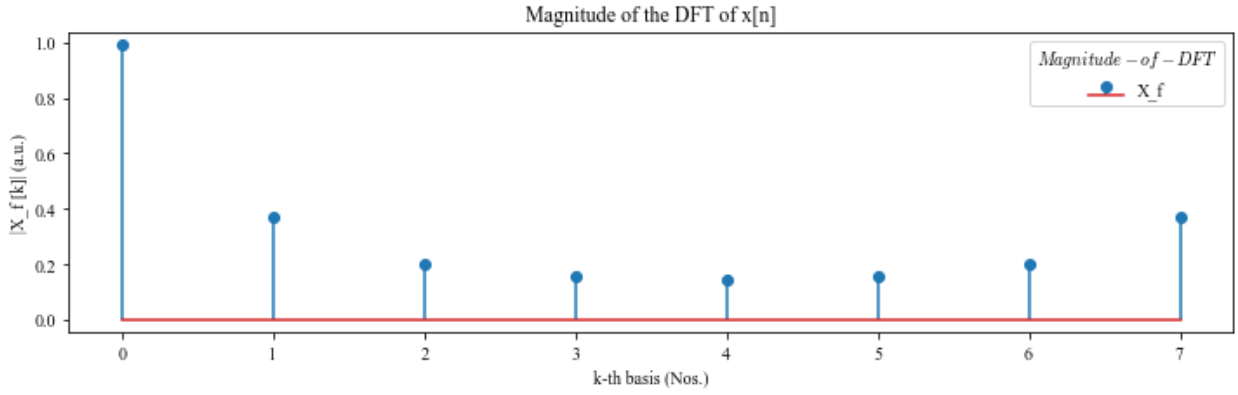

Figure 1: Plot of Discrete Fourier Transform (DFT): $|X_f[k]|$ for $k \in \{0, 1, \dots 7\}$

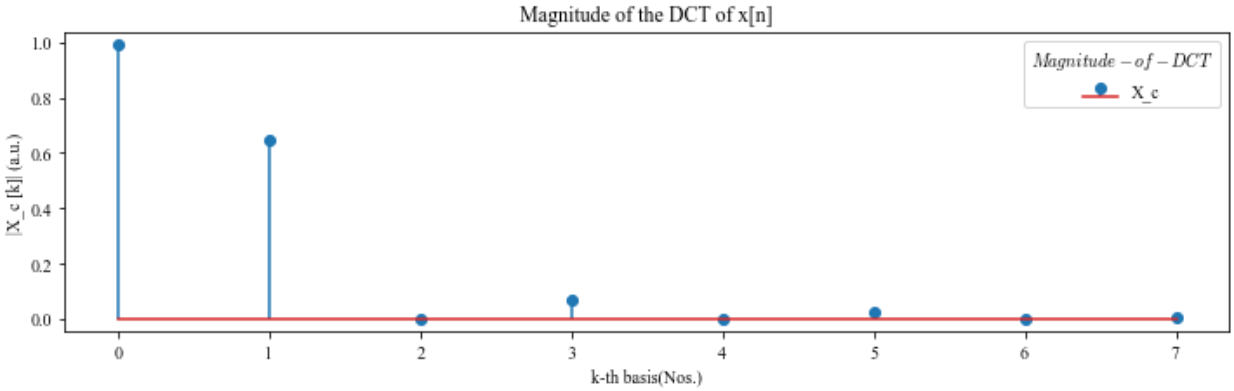

Figure 2: Plot of Cosine Fourier Transform (DCT): $|X_c[k]|$ for $k \in \{0, 1, \ldots 7\}$

Which transform does a better job of concentrating the energy into a fewer number of bins?

Regarding the question of which transform does a better job of concentrating the energy into a fewer number of bins, it depends on the specific signal being transformed and the desired properties of the transform. In general, the DCT is more often used for signals that have energy concentrated in a small number of bins (e.g. signals that are sparse in the frequency domain), while the DFT is more often used for signals that have energy distributed evenly across all bins. This is also shown in Figure 1 (plot for DFT) and Figure 2 (plot for DCT).

In general, the DCT tends to have better energy compaction properties compared to the DFT. The DCT represents signals with most of their energy concentrated in a few low-frequency coefficients, making it more suitable for image and audio compression applications. On the other hand, the DFT provides a full frequency analysis of the signal, making it more suitable for applications such as frequency-domain filtering or spectrum analysis.

Appendix

Python code for Solving Problem-2 of HW2:

```
# Name: Sohag Kumar Saha
# Homework 2, Problem 2
# ECE 6040 (Signal Analysis), Spring 2023
# Tennessee Tech University

#import numpy library

import numpy as np

# declaring user define function for inner product calculation
def inner_product(N):
    # Create the basis set b_k [n]
    n = np.arange(N)
    b_k = np.zeros((N, N))
    b_l = np.zeros((N,N))

    #calculation of b_k[n]
    for k in range(N):
        if k == 0:
            b_k[k, :] = 1 / np.sqrt(N)
        else:
            b_k[k, :] = np.sqrt(2 / N) * np.cos(((np.pi) / N) * (n + 0.5) * k)

    # calculation of conjugate of bk[n] which is termed as bl[n]
    for l in range(N):
        if l == 0:
            b_l[l, :] = np.conj (1 / np.sqrt(N))
        else:
            b_l[l, :] = np.conj(np.sqrt(2 / N) * np.cos(((np.pi) / N)  * (n + 0.5) * l))

    # Calculate the inner product between each element of the basis set
    inner_products = np.zeros((N, N))
    for k in range(N):
        for l in range(N):

            # the inner product values (no rounding here)
            #inner_products[k, l] = round(np.inner(b_k[k, :], b_l[l, :]))
```

```
            # rounding the inner product values so that we get zero or one rather than very sm
all decimal values
            inner_products[k, l] = round(np.inner(b_k[k, :], b_l[l, :]))

    return inner_products

# define main function
def main():

    # Calculate the inner product for N = 16
    N = 16
    inner_products = inner_product(N)
    print("N =", N)
    for k in range(N):
        for l in range(N):
            # printing inner product
            print("k =", k, ", l =", l, ", inner_product = ", inner_products[k, l])

    # Calculate the inner product for N = 64
    N = 64
    inner_products = inner_product(N)
    print("N =", N)
    for k in range(N):
        for l in range(N):
            # printing inner product
            print("k =", k, ", l =", l, ", inner_product = ", inner_products[k, l])

    # Calculate the inner product for N = 128
    N = 128
    inner_products = inner_product(N)
    print("N =", N)
    for k in range(N):
        for l in range(N):
            # printing inner product
            print("k =", k, ", l =", l, ", inner_product = ", inner_products[k, l])


if __name__ == '__main__':
    main()
```

Python code for Solving Problem-3 of HW2:

```
# Name: Sohag Kumar Saha
# Assignment-2 (Problem 3)
# ECE 6040 (Signal Analysis), Spring 2023
# Tennessee Tech University

#import numpy, matplotlib library

import numpy as np
import matplotlib.pyplot as plt

# initialize values
N = 8
n = np.arange(N)
x = 0.1 * n

# Define Function for Discrete Fourier Transform  - DFT
def DFT_Function(x):
    X = np.zeros(N, dtype=complex)
    for k in range(N):
        for n in range(N):
            X[k] += x[n] * (1/(np.sqrt(N)))  * np.exp((1j *2* np.pi * k * n) / N)
    return X

# Define Function for Discrete Consine Transform - DCT
def DCT_Function(x):
    X = np.zeros(N)
    for k in range(N):
        if k == 0:
            c = 1 / np.sqrt(N)
        else:
            c = np.sqrt(2 / N)
            #c = np.sqrt(2 / N) * np.cos(((np.pi)/N) * (n + 0.5) * k )

        for n in range(N):
            X[k] += x[n] * c * np.cos(((np.pi)/N) * (n + 0.5) * k )
            #X[k] += x[n] * c

    return X
```

```
# Define Main function
def main():

    X_f = DFT_Function(x) # calling DFT function
    X_c = DCT_Function(x) # calling DCT function


    # For pretty looking fonts
    plt.rcParams['font.family'] = 'serif'
    plt.rcParams['font.serif'] = 'Times New Roman'
    plt.rcParams["mathtext.fontset"] = "cm"

    # Plot of DFT and DCT
    plt.figure()
    plt.stem(np.abs(X_f), label='X_f')
    plt.title("DFT Magnitude")
    plt.xlabel(r'k-th basis (Nos.)')
    plt.ylabel(r'|X_f [k]| (a.u.)')
    plt.title('Magnitude of the DFT of x[n]')
    plt.legend(title='$Magnitude-of-DFT$')

    plt.figure()
    plt.stem(np.abs(X_c), label='X_c')
    plt.title("DCT Magnitude")
    plt.xlabel(r'k-th basis(Nos.)')
    plt.ylabel(r'|X_c [k]| (a.u.)')
    plt.title('Magnitude of the DCT of x[n]')
    plt.legend(title='$Magnitude-of-DCT$')

    plt.show()

    # Verification of Parsevals theorem
    Sq_DFT = np.sum(abs(X_f)**2)
    Sq_DCT = np.sum(abs(X_c)**2)

    if Sq_DFT == Sq_DCT:
        print ('Comments on the satisfactory of the parseval theorem: The Parseval-
s theorem is found: True!')
    else:
        print ('Comments on the satisfactory of the parseval theorem: The parseval-
s theorem is found: False!' )


if __name__ == '__main__':
    main()
```