# Math 104A Homework 5

Rad Mallari
8360828

**1.) The Discrete Fourier Transform (DFT)** is a periodic array $f_j$, for $j = 0, \ldots, N-1$ (corresponding to data at equally spaced point of the interval of periodicity) is evaluated via the Fast Fourier Transform (FFT) algorithm ($N$ power of $2$). Use FFT package, i.e. an already coded FFT (e.g scipy.fftpack or numpy.fft or fft() in matlab).

**a.)** Which of the following expressions define the Fourier coefficients (DFT) that fits your package (name it) returns for $k = 0, \ldots, N-1$?

(a) $c_k = \sum_{j=1}^{N} f_j e^{-\frac{i2\pi kj}{N}}$
(b) $c_k = \sum_{j=1}^{N} f_j e^{-\frac{i2\pi k(j-1}{N}}$
(c) $c_k = \sum_{j=1}^{N-1} f_j e^{-\frac{i2\pi kj}{N}}$

**b.)** Which of the following expressions define the inverse DFT computed by fft package?

(a) $c_k = \sum_{j=1}^{N} f_j e^{-\frac{i2\pi kj}{N}}$
(b) $c_k = \sum_{j=1}^{N} f_j e^{-\frac{i2\pi k(j-1}{N}}$
(c) $c_k = \sum_{j=1}^{N-1} f_j e^{-\frac{i2\pi kj}{N}}$

In [1]:

```python
import numpy as np
from math import e, pi

f_j = np.random.rand(
    16,
)
N = len(f_j)

package_fft_coeffs = list(np.fft.fft(f_j))

a_expression_coeffs = [complex(0, 0) for n in range(N)]
b_expression_coeffs = [complex(0, 0) for n in range(N)]
c_expression_coeffs = [complex(0, 0) for n in range(N)]

_2ipi = complex(0, -1) * 2 * pi

# checking (a) c_k=SUM_j=1^N f_je^(-i2pikj/N)
for k in range(N):
    for j in range(1, N):
        a_expression_coeffs[k] += f_j[j - 1] * e ** (_2ipi * k * j / N)
# checking (b) c_k=SUM_j=0^N f_je^(-i2pik(j-1)/N)
for k in range(N):
    for j in range(N):
        b_expression_coeffs[k] += f_j[j] * e ** (_2ipi * k * (j - 1) / N)
# checking (c) c_k=SUM_j=0^N-1 f_je^(-i2pikj/N)
for k in range(N):
    for j in range(N - 1):
        c_expression_coeffs[k] += f_j[j] * e ** (_2ipi * k * j / N)

print("Distances between expression (a) points and numpy fft method:")
print(
    [
        np.round(np.abs(a - t), 3)
        for a, t in zip(a_expression_coeffs, package_fft_coeffs)
    ]
)
print("Distances between expression (b) points and numpy fft method:")
print(
    [
        np.round(np.abs(b - t), 3)
        for b, t in zip(b_expression_coeffs, package_fft_coeffs)
    ]
)
print("Distances between expression (c) points and numpy fft method:")
print(
    [
        np.round(np.abs(c - t), 3)
        for c, t in zip(c_expression_coeffs, package_fft_coeffs)
    ]
)
```

```
Distances between expression (a) points and numpy fft method:
[0.307, 0.439, 0.738, 1.452, 0.673, 2.085, 1.254, 2.881, 0.573, 2.881, 1.254, 2.085, 0.673, 1.452, 0.738, 0.439]
Distances between expression (b) points and numpy fft method:
[0.0, 0.144, 0.971, 1.259, 0.785, 2.21, 1.554, 2.649, 0.266, 2.649, 1.554, 2.21, 0.785, 1.259, 0.971, 0.144]
Distances between expression (c) points and numpy fft method:
[0.307, 0.307, 0.307, 0.307, 0.307, 0.307, 0.307, 0.307, 0.307, 0.307, 0.307, 0.307, 0.307, 0.307, 0.307, 0.307]
```

**Rerunning the code multiple times, we see that the closest distance between the coefficients is consistently expression (c)**

b.) Which of the following expressions define the inverse DFT computed by fft package?

(a) $c_k = \sum_{j=1}^{N} f_j e^{-\frac{i2\pi kj}{N}}$
(b) $c_k = \sum_{j=1}^{N} f_j e^{-\frac{i2\pi k(j-1}{N}}$
(c) $c_k = \sum_{j=1}^{N-1} f_j e^{-\frac{i2\pi kj}{N}}$

In [2]:

```python
package_fft_coeffs = list(np.fft.ifft(f_j))
a_expression_f_j = [complex(0, 0) for n in range(N)]
b_expression_f_j = [complex(0, 0) for n in range(N)]
c_expression_f_j = [complex(0, 0) for n in range(N)]


# checking (a) f_j=2/N SUM_k=0^N-1 c_ke^i2pikj/N for j=0,1,...,N-1
for j in range(N):
    for k in range(N):
        a_expression_f_j[j] += package_fft_coeffs[j] * e ** (-_2ipi * k * j / N)
        a_expression_f_j[j] /= 2 * N
# checking (b) f_j=1/N SUM_k=0^N-1 c_ke^i2pikj/N for j=0,1,...,N-1
for j in range(N):
    for k in range(N):
        b_expression_f_j[j] += package_fft_coeffs[j] * e ** (-_2ipi * k * j
                                                             / N)
        b_expression_f_j[j] /= N
# checking (c) f_j=1/N SUM_k=1^N c_ke^i2pikj/N for j=0,1,...,N-1
for j in range(N):
    for k in range(1, N):
        c_expression_f_j[j] += package_fft_coeffs[j] * e ** (-_2ipi * k * j / N)
        c_expression_f_j[j] /= N

print("Original f_j:")
print(f_j)
print("Distances between expression (a) points and original f_j")
print(
    [
        np.round(np.abs(a - t), 3)
        for a, t in zip(a_expression_f_j, f_j)
    ]
)
print("Distances between expression (b) points and original f_j")
print(
    [
        np.round(np.abs(b - t), 3)
        for b, t in zip(b_expression_f_j, f_j)
    ]
)
print("Distances between expression (c) points and original f_j")
print(
    [
        np.round(np.abs(c - t), 3)
        for c, t in zip(c_expression_f_j, f_j)
    ]
)
```

```
Original f_j:
[0.19647159 0.03475025 0.52094032 0.50341066 0.14813285 0.49364657
 0.34428479 0.55751014 0.01505941 0.14826342 0.01221182 0.07814163
 0.94361547 0.2282745  0.30350729 0.3073653 ]
Distances between expression (a) points and original f_j
[0.187, 0.034, 0.523, 0.502, 0.149, 0.496, 0.344, 0.558, 0.015, 0.149, 0.012, 0.081, 0.945, 0.227, 0.306, 0.307]
Distances between expression (b) points and original f_j
[0.176, 0.033, 0.526, 0.5, 0.15, 0.498, 0.344, 0.559, 0.016, 0.15, 0.012, 0.083, 0.946, 0.225, 0.309, 0.306]
Distances between expression (c) points and original f_j
[0.176, 0.033, 0.526, 0.5, 0.15, 0.498, 0.344, 0.559, 0.016, 0.15, 0.012, 0.083, 0.946, 0.225, 0.309, 0.306]
```

**Rerunning the code multiple times, we see that differences that expression (b) and (c) are consistenly similar, so it could be either.**

**2.)** Prove that if $f_j$ for $j = 0, 1, \ldots, N-1$ are real numbers, then $c_0$ is real and $c_{N-k} = \overline{c_k}$ where $\overline{c_k}$ is the complex conjugate.

**Proof:**

We know that $c_k$ is given by:

$$c_k = \sum_{j=0}^{N-1} f_j e^{-2\pi ikj/N}$$

Therefore, letting $f_j \in \mathbb{R}$, then

$$c_0 = \sum_{j=0}^{N-1} f_j e^{-2\pi i \cdot 0 \cdot j/N} = \sum_{j=0}^{N-1} f_j e^0$$

Here we know that $e^0 = 1$ which is a real number, and since $f_j$ is assumed to be real, $c_0$ must be real.
Furthermore, by the definition, we know:

$$c_{N-k} = \sum_{j=0}^{N-1} f_j e^{-2\pi ij(N-k)/N} = \sum_{j=0}^{N-1} f_j e^{-2\pi ij} \cdot e^{2\pi ijk/N}$$

In class we proved that $e^{-2\pi in}$ where $n \in \mathbb{Z}$ is $1$, therefore, we are left with:

$$c_{N-k} = \sum_{j=0}^{N-1} f_j e^{2\pi ijk/N}$$

Meanwhile, $\overline{c_k}$ is defined as:

$$\overline{c_k} = \sum_{j=0}^{N-1} \overline{f_j e^{-2\pi ijk/N}}$$

Complex conjugate of $e^{-iz}$ is $e^{iz}$. Moreover, since $f_j$ is a real number then it stays the same, therefore:

$$\overline{c_k} = \sum_{j=0}^{N-1} f_j e^{2\pi ijk/N}$$

Which is exactly $c_{N-k}$ thereby proving $c_{N-k} = \overline{c_k}$

**3.)** Recall the inner product axioms

- $\langle f, h\rangle = \langle h, f\rangle$
- $\langle f, \alpha h + \beta g\rangle = \alpha\langle f, h\rangle + \beta\langle f, g\rangle$
- $\langle f, f\rangle > 0$ if $f \neq 0$
- $\|f\| = \sqrt{\langle f, f\rangle}$

Suppose $\{\phi_0(x), \phi_1(x), \ldots, \phi_n(x)\}$ is an orthogonal set of functions with respect to the $L^2$ inner product, i.e.,

$$\langle \phi_j, \phi_k\rangle = \int_a^b \phi_j(x)\phi_k(x)dx = 0, \quad j \neq k$$

Using axioms of inner product to prove the Pythagorean theorem

$$\|\phi_0 + \phi_1 + \ldots + \phi_n\|^2 = \|\phi_0\|^2 + \|\phi_1\|^2 + \ldots + \|\phi_n\|^2$$

**Proof:**

By the last axiom, the left side of the equation is given by:

$$\|\phi_0 + \phi_1 + \ldots + \phi_n\|^2 = \langle \phi_0 + \phi_1 + \ldots + \phi_n, \phi_0 + \phi_1 + \ldots + \phi_n\rangle$$

Then by the second axiom:

$$\langle \phi_0 + \phi_1 + \ldots + \phi_n, \phi_0 + \phi_1 + \ldots + \phi_n\rangle = \langle \phi_0 + \phi_1 + \ldots + \phi_n, \phi_0\rangle + \langle \phi_0 + \phi_1 + \ldots + \phi_n, \phi_1\rangle + \ldots + \langle \phi_0 + \phi_1 + \ldots + \phi_n, \phi_n\rangle$$

And by the first axiom, we rewrite this as:

$$\langle \phi_0 + \phi_1 + \ldots + \phi_n, \phi_0\rangle + \langle \phi_0 + \phi_1 + \ldots + \phi_n, \phi_1\rangle + \ldots + \langle \phi_0 + \phi_1 + \ldots + \phi_n, \phi_n\rangle = \langle \phi_0, \phi_0 + \phi_1 + \ldots + \phi_n\rangle + \langle \phi_1, \phi_0 + \phi_1 + \ldots + \phi_n\rangle + \ldots + \langle \phi_n, \phi_0 + \phi_1 + \ldots + \phi_n\rangle$$

Repeating these steps yields:

$$\langle \phi_0, \phi_0\rangle + \langle \phi_0, \phi_1\rangle + \ldots + \langle \phi_0, \phi_n\rangle + \langle \phi_1, \phi_0\rangle + \langle \phi_1, \phi_1\rangle + \ldots + \langle \phi_1, \phi_n\rangle + \langle \phi_n, \phi_0\rangle + \langle \phi_n, \phi_1\rangle + \ldots + \langle \phi_n, \phi_n\rangle$$

Since our $\phi$s are orthogonal functions, the inner products between $\phi$'s that do not have the same index will be $0$ and we are left with:

$$\langle \phi_0, \phi_0\rangle + \langle \phi_1, \phi_1\rangle + \ldots + \langle \phi_n, \phi_n\rangle$$

Which by the last axiom is equivalent to:

$$\langle \phi_0, \phi_0\rangle + \langle \phi_1, \phi_1\rangle + \ldots + \langle \phi_n, \phi_n\rangle = \|\phi_0\|^2 + \|\phi_1\|^2 + \ldots + \|\phi_n\|^2$$

Proving that $\|\phi_0 + \phi_1 + \ldots + \phi_n\|^2 = \|\phi_0\|^2 + \|\phi_1\|^2 + \ldots + \|\phi_n\|^2$

**4.)** Consider the inner product $\int_{-1}^{1} f(x)g(x)dx$. Recall Legendre polynomials are obtained by orthogonalization of $\{1, x, \ldots, x^n\}$.

**a.)** Compute the first $4$ Legendre polynomials in $[-1, 1]$.
**b.)** Find the least squares polynomial approximations of degrees $1$, $2$, and $3$ for the function $f(x) = e^x$ on $[-1, 1]$.
**c.)** What is the least squares approximation of degree $3$ for $f(x) = x^2 - x - 1$ on $[-1, 1]$? Explain.

**Proof:**

**a.)** We obtain this by the follow:

$$\xi_0(x) = 1$$

$$\xi_1(x) = x - \alpha_0, \quad \alpha_0 = \frac{\langle x\xi_0, \xi_0\rangle}{\langle \xi_0, \xi_0\rangle} = 0 \Rightarrow \xi_1(x) = x$$

$$\xi_2(x) = (x - \alpha_1)\xi_1(x) - \beta_1\xi_0(x), \quad \alpha_1 = \frac{\langle x\xi_0, \xi_0\rangle}{\langle \xi_0, \xi_0\rangle} = 0, \quad \beta_1 = \frac{\langle x\xi_1, \xi_0\rangle}{\langle \xi_0, \xi_0\rangle} = \frac{1}{3} \Rightarrow \xi_2(x) = x^2 - \frac{1}{3}$$

$$\xi_3(x) = (x - \alpha_2)\xi_2(x) - \beta_2\xi_1(x), \quad \alpha_2 = \frac{\langle x\xi_1, \xi_1\rangle}{\langle \xi_1, \xi_1\rangle} = 0, \quad \beta_2 = \frac{\langle x\xi_2, \xi_1\rangle}{\langle \xi_1, \xi_1\rangle} = \frac{4}{15} \Rightarrow \xi_3(x) = x^3 - \frac{3}{5}x$$

**b.)** Since we have an orthonormal system using the orthogonalization of $\{1, x, \ldots, x^n\}$, our least squares polynomial approximation can be obtained using:

$$\sum_{i=1}^{n} c_i g_i$$

Where $n$ is our degree and $c_i = \langle f, g\rangle$. Therefore, we have:

$$c_0 = \int_{-1}^{1} e^x dx = e^{-x}\Big|_{-1}^{1} = e - \frac{1}{e}$$

$$c_1 = \int_{-1}^{1} x \cdot e^x dx = x \cdot e^{-x}\Big|_{-1}^{1} - \int_{-1}^{1} e^x dx = x \cdot e^{-x}\Big|_{-1}^{1} - e^x\Big|_{-1}^{1} = \frac{2}{e}$$

$$c_2 = \int_{-1}^{1}\left(x^2 - \frac{1}{3}e^x\right)dx = x^2e^x\Big|_{-1}^{1} - 2\left[xe^x\Big|_{-1}^{1} - e^x\Big|_{-1}^{1}\right] - \frac{1}{3}(e^x)\Big|_{-1}^{1} = \frac{1}{3}\left(2e - \frac{14}{e}\right)$$

$$c_3 = \int_{-1}^{1}\left(x^3 - \frac{3}{5}x\right)e^x dx = x^3e^x\Big|_{-1}^{1} - 3\left[\int_{-1}^{1} x^2e^x dx\right] - \frac{3}{5}\int_{-1}^{1} xe^x dx = -2e + \frac{74}{5e}$$

Therefore, our polynomial is given by:

$$e^x \approx e - \frac{1}{e} + \frac{2}{e}(x) + \frac{1}{3}\left(2e - \frac{14}{e}\right)\left(x^2 - \frac{1}{3}\right) + \left(-2e + \frac{74}{5e}\right)\left(x^3 - \frac{3}{5}x\right)$$

**(c)** The least squares approximation of degree $3$ for $f(x) = x^2 - x - 1$ on $[-1, 1]$ is exactly itself. This is because the Least-Squares Theory approximates a continuous function $f$ using some polynomial $p$ with degree at most $n$ that deviates as little as possible from $f$. This deviation is measured by $|f(x) - p(x)|$, and here it's clear that a polynomial with degree $3$ approximating a polynomial with degree $2$ that deviates a little as possible would be the degree $2$ polynomial itself.

**5.)** Let $T_n(x)$ denote the Chebyshev polynomial on $[-1, 1]$. Prove that

$$\frac{2}{\pi}\int_{-1}^{1}\frac{T_n^2(x)}{\sqrt{1-x^2}}dx = 1$$

**Proof:**

We define the Chebyshev polynomial for $x \in [-1, 1]$ to be:

$$T_n(x) = \cos(n\cos^{-1}(x)) \quad (n \geq 0)$$

We have the property:

$$T_n(\cos\theta) = \cos(n\theta)$$

where $\theta \in [0, \pi]$, and $n \in \mathbb{N}$ Therefore, letting $x = \cos\theta \Rightarrow dx = -sin\theta d\theta$, we rewrite the left side of our equation as:

$$\frac{2}{\pi}\int_{-1}^{1}\frac{T_n^2(x)}{\sqrt{1-x^2}}dx = \frac{2}{\pi}\int_{\pi}^{0}\frac{(\cos(n\theta))^2}{\sqrt{1-\cos^2\theta}}\sin\theta d\theta$$

By trignometric identities, we know that $1 - \cos^2(\theta) = \sin^2(\theta)$, which yields:

$$\frac{2}{\pi}\int_{\pi}^{0}\frac{(\cos(n\theta))^2}{\sqrt{1-\cos^2\theta}}\sin\theta d\theta = -\frac{2}{\pi}\int_{\pi}^{0}(\cos(n\theta))^2 d\theta$$

Using the double angle formula, we expand $\cos^2(n\theta)$ as:

$$-\frac{2}{\pi}\int_{\pi}^{0}(\cos(n\theta))^2 d\theta = -\frac{2}{\pi}\left(\int_{\pi}^{0}\frac{1}{2}d\theta + \int_{\pi}^{0}\frac{\cos(2n\theta)}{2}d\theta\right)$$

$$-\frac{2}{\pi}\left(\int_{\pi}^{0}\frac{1}{2} + \int_{\pi}^{0}\frac{\cos(2n\theta)}{2}\right) = -\frac{2}{\pi}\left(\frac{\theta}{2}\Big|_{\pi}^{0} + \frac{1}{2}\sin(2n\theta)\Big|_{\pi}^{0}\right)$$

$$-\frac{2}{\pi}\left(\frac{\theta}{2}\Big|_{\pi}^{0} + \frac{1}{2}\sin(2n\theta)\Big|_{\pi}^{0}\right) = -\frac{2}{\pi}\left(-\frac{\pi}{2}\right) = 1$$

**6.)** Given a collection of data points $\{(x_i, y_i)\}_{i=1}^{m}$

**a.)** Find the best least squares of the form $y = ax + bx^2$
**b.)** Use the approximation to fit the data in Table 2 and find the error in the least square approximation. |$x_j$|$y_j$| |---|---| |1|$3.1$| |2|$9.8$| |3|$21.2$| |4|$36.1$| **Proof:**
**a.)** We use the general formula to minimize the least square of $a_0, a_1, \ldots, a_n$, which is given by:

$$E(a_0, a_1, \ldots, a_n) = \sum_{i=1}^{m}(y_i - P_n(x_i))^2$$

Where $P_n$ is the polynomial of interest. Therefore, for our case, we have:

$$E(a_0, a_1, a_2) = \sum_{i=1}^{m}(y_i - P_2(x_i))^2$$

$$\sum_{i=1}^{m}(y_i - P_2(x_i))^2 = \sum_{i=1}^{m}\left(y_i - \sum_{k=0}^{2}a_kx_i^k\right)^2$$

$$\sum_{i=1}^{m}\left(y_i - \sum_{k=0}^{2}a_kx_i^k\right)^2 = \sum_{i=1}^{m}\left(y_i - a_0 - a_1x_i - a_2x_i^2\right)^2$$

Since $a_0 = 0$ in our case we are left with as our best least squares approximation:

$$\sum_{i=1}^{m}\left(y_i - a_1x_i - a_2x_i^2\right)^2$$

To solve this, we first take the partial derivatives with respect to each $a_n$:

$$0 = \frac{\partial E}{\partial a_1} = 2\sum_{i=1}^{m}(y_i - a_1x_i - a_2x_i^2)(-x_i)$$

$$0 = \frac{\partial E}{\partial a_2} = 2\sum_{i=1}^{m}(y_i - a_1x_i - a_2x_i^2)(-x_i^2)$$

This gives us our normal equations:

$$\sum_{i=1}^{m}a_1x_i^2 + \sum_{i=1}^{m}a_2x_i^3 = \sum_{i=1}^{m}x_iy_i$$

$$\sum_{i=1}^{m}a_1x_i^3 + \sum_{i=1}^{m}a_1x_i^4 = \sum_{i=1}^{m}x_i^2y_i$$

**b.)** Now to use this approximation to fit the data, we represent our normal equations in matrix form given by $A^TA\vec{a} = A^T\vec{b}$, where:

$$A = \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^n \\ 1 & x_2 & x_2^2 & \ldots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \ldots & x_m^n \end{bmatrix} \quad \text{and} \quad \vec{b} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

For our problem we have:

$$A = \begin{bmatrix} 1 & 1 & 1^2 \\ 1 & 2 & 2^2 \\ 1 & 3 & 3^2 \\ 1 & 4 & 4^2 \end{bmatrix}, \quad A^T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1^2 & 2^2 & 3^2 & 4^2 \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} 3.1 \\ 9.8 \\ 21.2 \\ 36.1 \end{bmatrix}$$

Computing the matrices:

$$A^TA = \begin{bmatrix} 4 & 10 & 30 \\ 10 & 30 & 100 \\ 30 & 100 & 354 \end{bmatrix}, \quad A^Tb = \begin{bmatrix} 70.2 \\ 230.7 \\ 810.7 \end{bmatrix}$$

This gives us $3$ equations:

$$4a_0 + 10a_1 + 30a_2 = 70.2$$
$$10a_0 + 30a_1 + 100a_2 = 230.7$$
$$30a_0 + 100a_1 + 354a_2 = 810.7$$

Solving for our coefficients, gives us $a_1 = 0.96, a_2 = 2.017 \Rightarrow y = (0.96)x + (2.017)x^2$. For the errors, we have:

- $x = 1$ the error is: $3.1 - (0.96 + 2.017) = 0.123$
- $x = 2$ the error is: $9.8 - (1.92 + 8.068) = -0.188$
- $x = 3$ the error is: $21.2 - (2.88 + 18.153) = 0.167$
- $x = 4$ the error is: $36.1 - (3.84 + 32.272) = -0.012$

In [ ]: