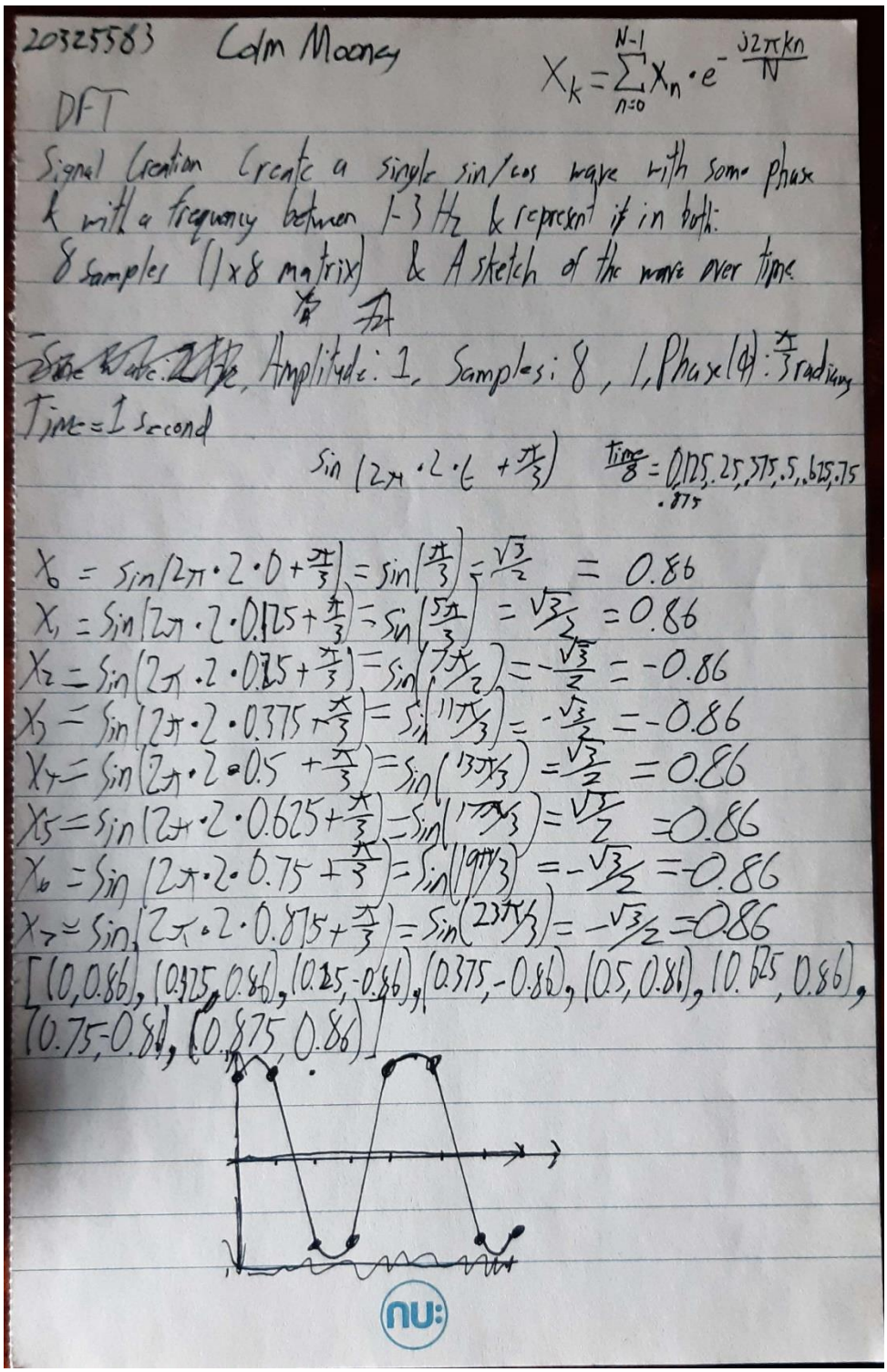
Colm Mooney -20325583

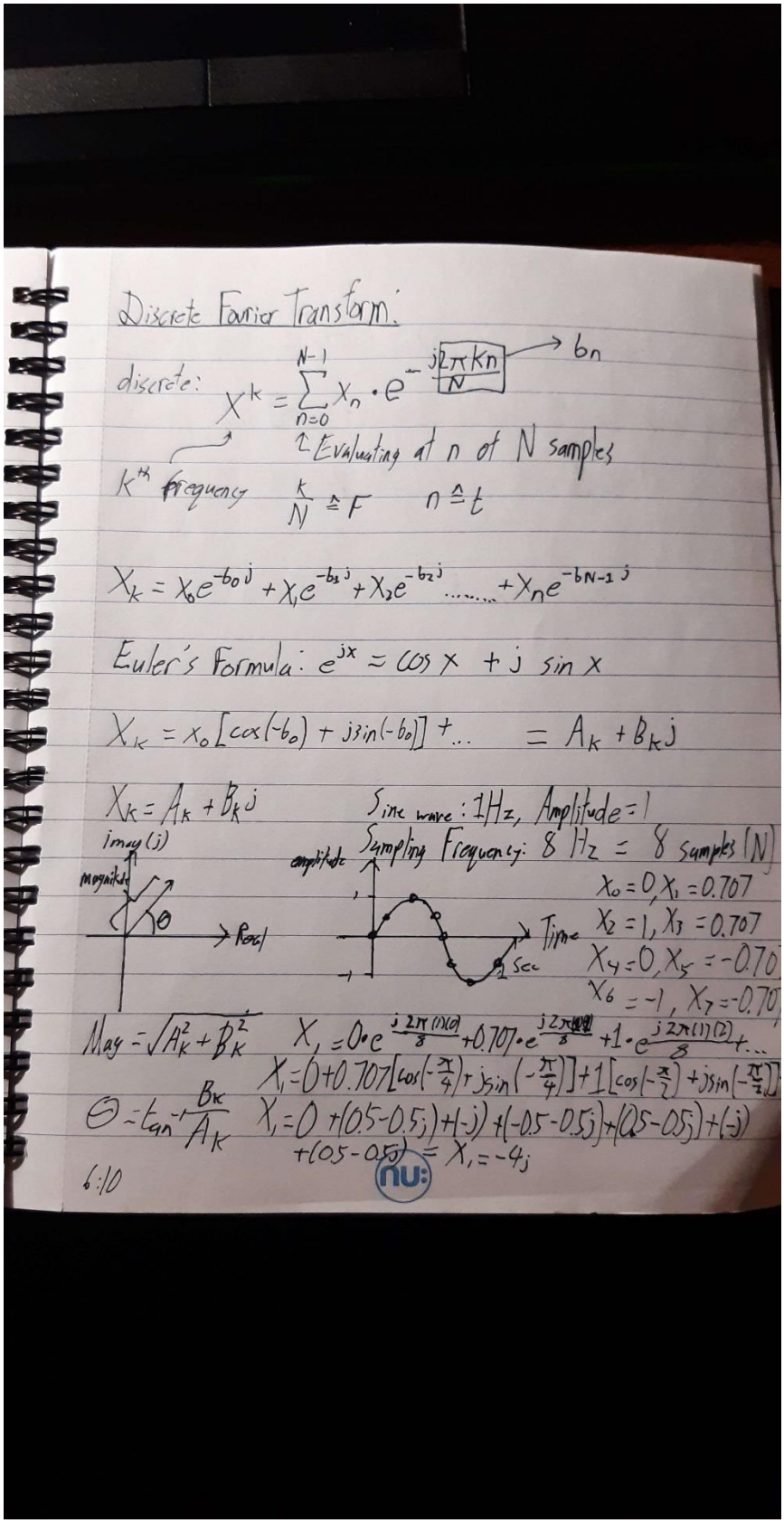
Task 1: Discrete Fourier Transform (DFT)

Signal Creation: Create a single sin/cos wave with some phase and with a frequency between 1-3 Hz and represent it in both: 8 samples (1x8 matrix). A sketch of the wave over time.

Attempt 1



Attempt 2:



Discrete Fourier Transform:

discrete: $X^k = \sum_{n=0}^{N-1} X_n \cdot e^{-\frac{j2\pi kn}{N}}$ → b_n

↑ Evaluating at n of N samples

k^{th} frequency $\frac{k}{N} \triangleq F \quad n \triangleq t$

$X_k = X_0 e^{-b_0 j} + X_1 e^{-b_1 j} + X_2 e^{-b_2 j} \ldots\ldots + X_n e^{-b_{N-1} j}$

Euler's Formula: $e^{jx} = \cos x + j \sin x$

$X_k = X_0[\cos(-b_0) + j\sin(-b_0)] + \ldots = A_k + B_k j$

$X_k = A_k + B_k j$

imag (j)

magnitude

θ

Real

Sine wave: 1Hz, Amplitude = 1

Sampling Frequency: 8 Hz = 8 samples (N)

amplitude

Time

Sec

$X_0 = 0, X_1 = 0.707$

$X_2 = 1, X_3 = 0.707$

$X_4 = 0, X_5 = -0.70$

$X_6 = -1, X_7 = -0.70$

$Mag = \sqrt{A_k^2 + B_k^2}$

$\theta = \tan^{-1}\frac{B_k}{A_k}$

$X_1 = 0 \cdot e^{\frac{j2\pi(0)(0)}{8}} + 0.707 \cdot e^{\frac{j2\pi(1)(1)}{8}} + 1 \cdot e^{\frac{j2\pi(1)(2)}{8}} + \ldots$

$X_1 = 0 + 0.707[\cos(-\frac{\pi}{4}) + j\sin(-\frac{\pi}{4})] + 1[\cos(-\frac{\pi}{2}) + j\sin(-\frac{\pi}{2})]$

$X_1 = 0 + (0.5 - 0.5j) + (-j) + (-0.5 - 0.5j) + (0.5 - 0.5j) + (-j)$

$+(0.5 - 0.5j) = X_1 = -4j$

6:10

$$X_2 = 0 \cdot e^{\frac{j2\pi(2)(0)}{8}} + 0.707 \cdot e^{\frac{j2\pi(2)(1)}{8}} + 1 \cdot e^{\frac{j2\pi(2)(2)}{8}} + \dots$$

$$X_2 = 0 + 0.707\left[\cos\left(-\tfrac{\pi}{2}\right) + j\sin\left[-\tfrac{\pi}{2}\right]\right] + 1\left[\cos[-\pi] + j\sin[-\pi]\right] + \dots$$

$$X_2 = 0 + (-0.707) + (-1) + (0.707)j + (0.707j) + 1 + (-0.707j) = 0$$

$X_1 = 0 - 4j$ $\quad X_0, X_2 - X_8 = 0$

$X_7 = 0 + 4j$

$$M_{ay} = \sqrt{A_K^2 + B_K^2} = \sqrt{0^2 + (-8)^2} = 8$$

Task 2: Fast Fourier Transform (FFT)

Write pseudocode for how an FFT algorithm, explaining your understanding of how FFT works in 200 words. Calculate the FFT for the same signal you used for the DFT and compare the results.

How FFT works:

The Fast Fourier Transform is an algorithm that determines the Discrete Fourier Transform of a sequence faster than computing it. The idea is to decompose an N-point DFT into transformations of smaller length. This is done by reducing the number of real multiplications that is done.

FFTs are used for image, video & audio processing, communications, and can even be used to speed up the training of Convolutional Neural Networks. The FFT's ability to factorize the DFT matrix into a produce of mostly zero factors allows it to reduce the Computational Complexity from O(N^2) to O(N log N), where n = sequence length.

FFT provides a means of evaluating the DFT and IDFT equations with a number of operations proportional to N log2 N rather than N^2. This algorithm works best when N is a power of two. However, no savings are made if N is a prime number.

PseudoCode for an FFT algorithm:

```
SamplingFrequency = 1000
Time Interval = 1 / SamplingFrequency
SignalFrequency = 100
Samples = SamplingFrequency / SignalFrequency
FrequencyInterval = SamplingFrequency / Samples
Timesteps = CalculateTimesteps
Frequencysteps = Calculate Frequency


Sinodial wave = 1 * sin(2 * pi * SignalFrequency * TimeSteps)


numpy.fft.fft(SINEwave) = Answer
```

OR:

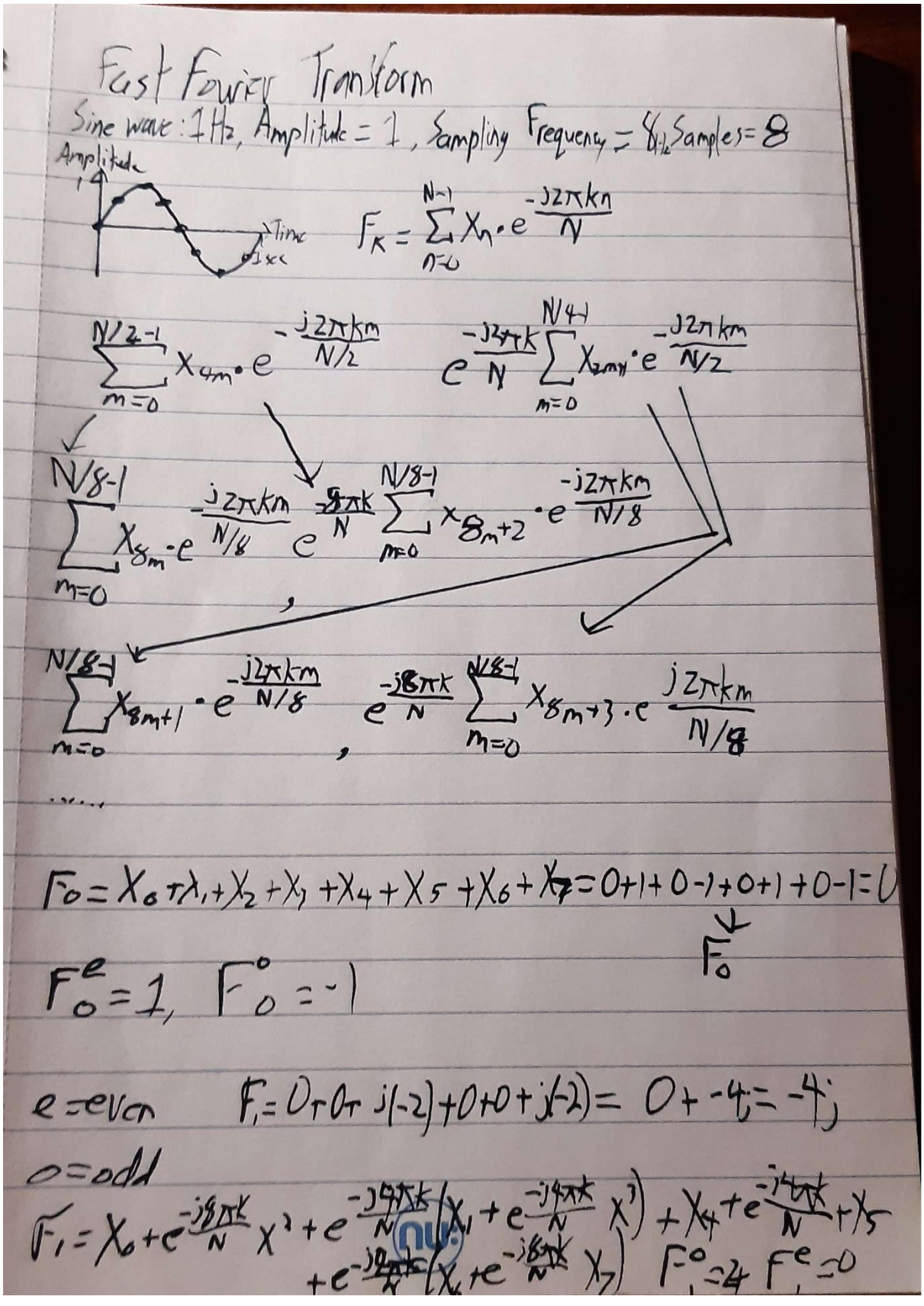
```
function fft(x):
n = length of x
 if n <= 1:
 return x
X_even = fft(even-index elements of x)
X_odd = fft(odd-index elements of x)
combined_X = new array of size n
for k from 0 to n/2 - 1:

        SinodialWave= exp(-2*pi*i*k/n)

        combined_X[k] = X_even[k] + twiddle_factor * X_odd[k]

        combined_X[k + n/2] = X_even[k] - twiddle_factor * X_odd[k]
```

return combined_X



The end results for this are the same. Both have -4j. Fast Fourier Transform however took significantly longer to understand. To be fair, FFT is mainly used for larger sample sizes, but they are both very similar in terms of mathematics. FFT is more efficient in computing, though can be very annoying to work with since you are splitting up the algorithms with odds and evens and can get lost in the process