Thomas Dormart – 260910500 Enzo-Benoit Jeannin – 260969262


## Abstract

To compare the efficiency of the 2D Discrete Fourier Transform (DFT) and the 2D Fast Fourier Transform (FFT), we implemented two image compression algorithms, each using one of the two functions. The compression removes Fourier coefficients in the Fourier domain of the image and uses inverse Fourier transform to generate a new compressed image from it. We compared both compress functions by monitoring how long each compression took depending on the size of the image to compress. The results showed the compression using the FFT was significantly faster than the one using the DFT especially as the image size increased.


## 1 Problem/Mathematical Setup

Our implementation of the 2D FFT relies on the 1D FFT as it first applies a standard FFT independently to all rows of the matrix before doing the same to all columns [1]. The Colley-Tukey recursion formula to compute the 1D FFT $\hat{f}$ of $f$ is:

$$\hat{f}[u] = \sum_{k=0}^{\frac{N}{2}-1} \hat{f}_e[k] + e^{\left(\frac{-2\pi u j}{N}\right)} \sum_{k=0}^{\frac{N}{2}-1} \hat{f}_o[k]$$

With $\hat{f}_e$ and $\hat{f}_o$ the FFT of the even and odd indexed values of $f$ and N the length of the signal $f$. Using this formula allows us to compute the Fourier Transform of a signal in time complexity $O(Nlog(N))$ compared to $O(N^2)$ with the DFT[1].

## 2 Numerical Methods and Algorithms

Our first implementation of the 1D FFT computed a length 1 FFT as base case, where the function would return the signal. We decided to explore another alternative by changing this base case from which the FFT algorithm would rely on the naïve DFT. We implemented a different function that takes as input the base case to use. We then wrote a test to compute the average time taken by the FFT for different array lengths and different base cases, which revealed that $2^{32}$ was the best base case to use. The comparison between base cases is shown in figure 1.

| Base Case Value | Average time |
|---|---|
| 1.0 | 0.4199945586206886 |
| 2.0 | 0.21892275172414322 |
| 4.0 | 0.11470188275861978 |
| 8.0 | 0.06714633448276064 |
| 16.0 | 0.041416337931044885 |
| 32.0 | 0.03629985862069663 |
| 64.0 | 0.04132486896551376 |
| 128.0 | 0.08175714827587435 |
| 256.0 | 0.2215677137931062 |
| 512.0 | 0.42696878275862493 |

**Figure 1: Table comparing the average time of the FFT algorithm depending on the base case used over signals of different length.**

Our compression application uses this new base case to compress images. The FFT is required for compression to access the frequency domain where all the data from the original image is preserved and where we can modify the Fourier coefficients [2]. Low coefficients affect the overall shape or pattern in the image, while the high frequencies are more important and provide more details [2]. Our compression algorithm takes a second input "p" after the image parameter, which is the percentage of the largest Fourier coefficients to keep in the Fourier domain. The algorithm sets to 0 all the frequencies lower than the minimum coefficient in the top "p%" in the 2D FFT of the image before returning its inverse FFT [2]. As we tested this algorithm, we noticed that for high values of "p", the compression sometimes increased the weight of the image instead.

## 3 Experiments and Results

To compare the difference in speed between the two algorithms we performed both synthetic and applicational benchmarks using the "timeit" library. To reuse our code, we decided to implement each test in different methods.

We compared timing performances of the DFT against both FFT functions on synthetic signals which showed the FFT with the base case of 32 was the fastest compare to the DFT and "1" base case FFT.

The last set of tests benchmarked our application and confirmed the images were correctly compressed. We applied our compression algorithms with different compression levels on different images. Lastly, we benchmarked the 2D FFT compression against DFT compression, which showed that using the FFT with a base case of $2^{32}$ significantly decrease the time to compress the image as shown in Figure 2.

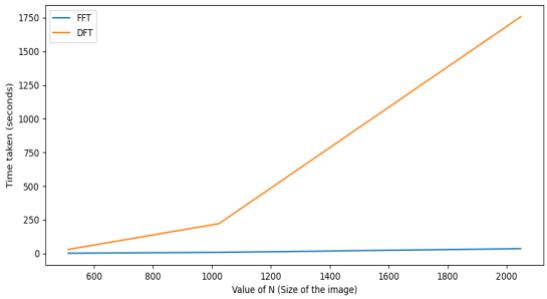


**Figure 2: Graph comparing the time taken to compress images using DFT and FFT.**

## Bibliography


[1] Transformation de Fourier discrète. (2021, octobre 21). Wikipédia, l'encyclopédie libre. Page consultée le 15:14, octobre 21, 2021 à partir de http://fr.wikipedia.org/w/index.php?title=Transformation_de_Fourier_discr%C3%A8te&oldid=187324609.

[2] S. A. A. Karim, M. H. Kamarudin, B. A. Karim, M. K. Hasan and J. Sulaiman, "Wavelet Transform and Fast Fourier Transform for signal compression: A comparative study," 2011 International Conference on Electronic Devices, Systems and Applications (ICEDSA), 2011, pp. 280-285, doi: 10.1109/ICEDSA.2011.5959031.