

Compression of data using the FFT

In this experiment we are asked to compress discrete time data by storing only the most significant points of the data's FFT. If every point in the FFT is stored, it is possible to recover the original signal exactly. But by effectively zeroing out less significant points in the FFT, we elect to introduce some error into the reconstructed signal in order to reduce the amount of storage space.

The plot below shows the FFT of a given data signal—the sum of three *complex sinusoids*. Both the data and the FFT have 64 points. Each sinusoid has magnitude $A = 1$ and the following frequencies:

$$\omega_1 = 2\pi(2.3438 \times 10^{-2}) \text{ radians/sec}$$

$$\omega_2 = 2\pi(8.6458 \times 10^{-2}) \text{ radians/sec}$$

$$\omega_3 = 2\pi(1.3409 \times 10^{-1}) \text{ radians/sec}$$

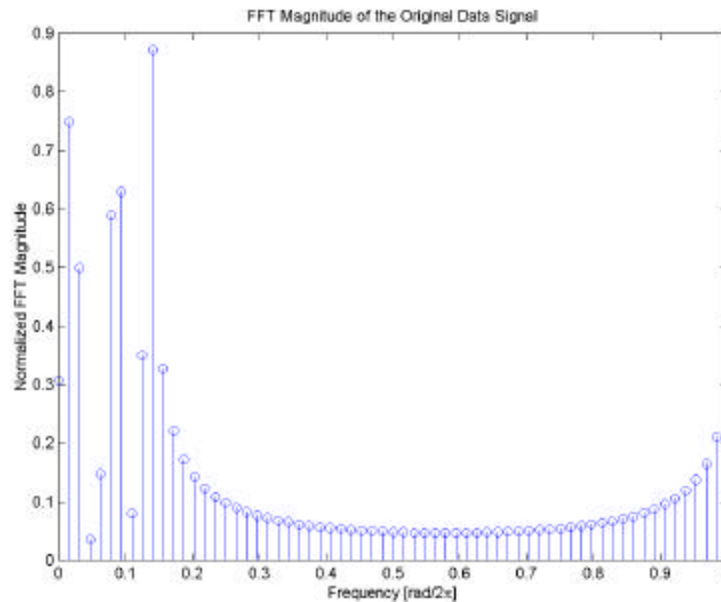


Figure 1: FFT magnitude of the original data signal, $d[n]$.

Noting that most of the signal's energy is in the low frequency region of the FFT, we arbitrarily decide to keep only the first 15 points. (The remaining points are assumed to be zero.) Taking the IFFT, the estimated signal is found to be a fairly close approximation of the original data signal. See the plots on the following page.

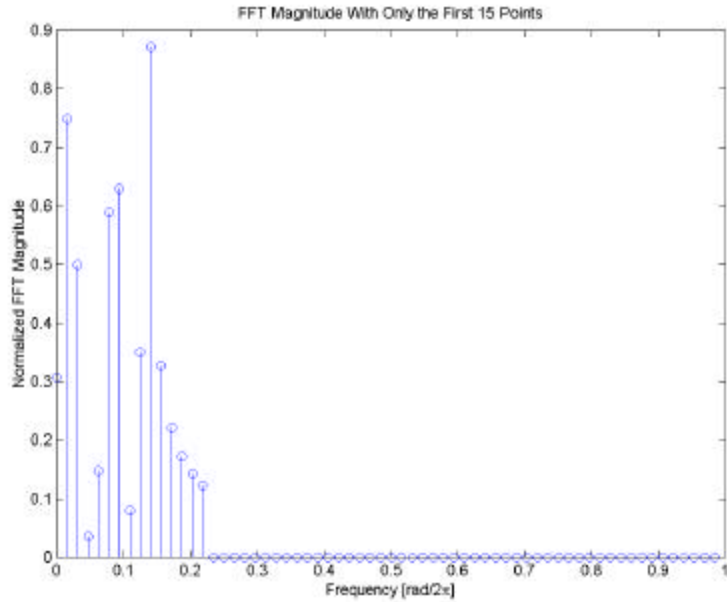


Figure 2: FFT magnitude of the original data signal, $d[n]$, with only the first 15 points retained.

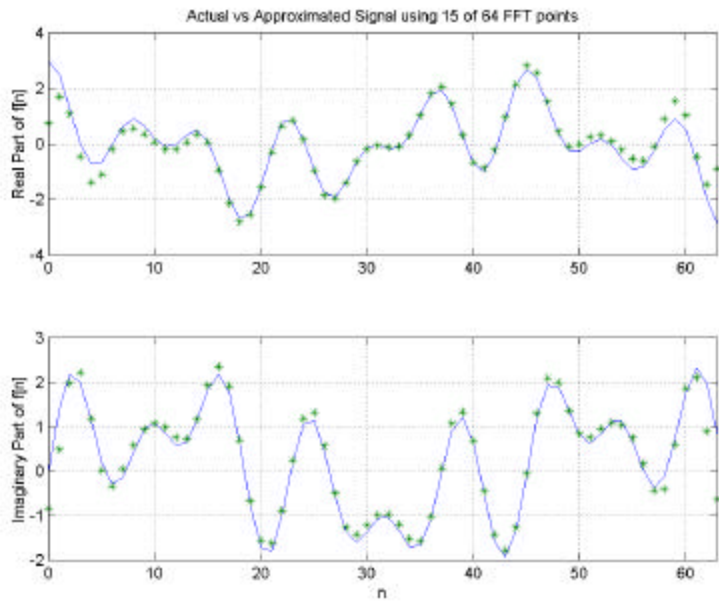


Figure 3: The real and imaginary parts of the real vs. the approximate signal. The solid (blue) signal is the original data, $d[n]$, while the discrete (green) stars represent the approximate signal, $a[n]$, which is the IFFT of signal in Figure 2.

Clearly there is some error between the actual and approximated signal. The normalized summed squared error is

$$E = \frac{\sum_{n=0}^{N-1} |d[n] - a[n]|^2}{\sum_{n=0}^{N-1} |d[n]|^2}$$

where $d[n]$ is the data signal and $a[n]$ is the approximated signal. The error in dB is $10 \log_{10}(E)$. Plotting the error as function of the number of points used (where we use the first M points of the FFT for $M = 0, 1, \dots, N$), we find the error decreases as M gets larger. For $M = 0$ the normalized error is just 1 (0 dB) and for $M = N$ there is no error.

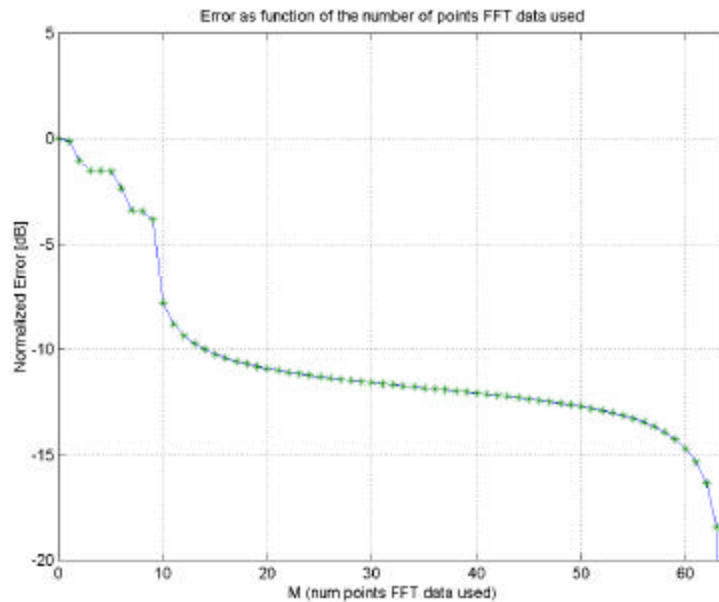


Figure 4: The error, E , as function of the number of points used, where we use the first M points of the FFT for $M = 0, 1, \dots, N$.

Note that for the example on the previous page, when we used the first 15 points of the FFT to approximate the data (i.e. $M = 15$), we find that $E = -10.23$ dB.

Further Thoughts

Windowing

We can improve the performance of this type of compression by windowing the data segment before performing the FFT. The plot below shows the error, E , when the original data is multiplied by a triangular window before taking the FFT. The windowing is undone after the IFFT by dividing by the same triangular window. In this case, the error for $M = 15$ is much better, $E = -19.05$ dB.

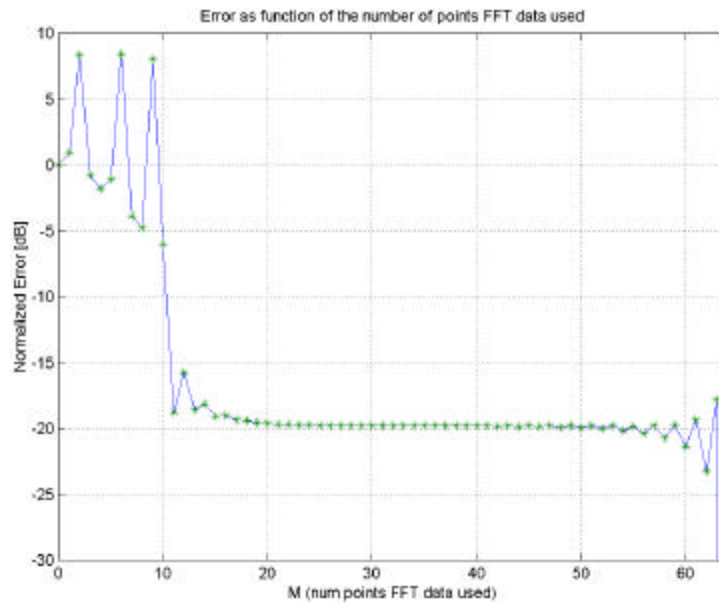


Figure 5: The error, E , as function of the number of points used, where we use the first M points of the FFT for $M = 0, 1, \dots, N$. In this case we multiplied by a *triangular* window before taking the FFT.

This method may not work as well when applied to highly quantized data such as an eight-bit grayscale image.

Conclusions

The example presented here shows how to compress data, data that is a sum of complex sinusoids. However most real-world signals (e.g. images and sounds) are real valued. Images, furthermore, are two-dimensional. Thus, the results from this example would have to be elaborated upon in order to actual apply this idea in a meaningful way.