Signal Processing and Discrete Fourier Transform

Andre Adam
Scott Cai
Zibo Wang

March 2019

1 Introduction

Discrete Fourier Transforms (DFT's) are a widely used method of signal processing for various different applications. Basically, a signal collected needs to be filtered in order for the patterns to be more intrinsically studied. A convolution thus needs to be made, and a DFT is nothing but a method of cleverly mapping this convolution to a dot product. More so, with modern computing a DFT can be computed using a fast Fourier transform (FFT), which allows for a very fast mapping of a signal.

This allows to many applications in medicine, engineering, and sciences. In this project, the FFTs are used as a resource to further filter raw data from Solar flares and tides. An FFT is also used to filter music into low, medium, and high frequencies, and separate those, similar to what speakers do. In addition, the mathematics of an inverse DFT will be explored, as well as a 2D DFT to filters high frequency noise on a picture.

1.1 Inverse DFT

Suppose W is the DFT matrix so that the transformation of length N signal $x[n]$ is

$$\mathscr{F}(x) = X = \frac{1}{N} W x,$$

where $W_{kn} = w^{kn(mod\ N)}$.

Claim: The DFT matrix, W, is invertible, and

$$W_{kn}^{-1} = \frac{1}{N} w^{-kn(mod\ N)}.$$

Proof: The DFT matrix can be treated as a Vandermonde matrix, i.e.:

$$W = \begin{bmatrix} (w^0)^0 & (w^0)^1 & (w^0)^2 & (w^0)^3 & ... & (w^0)^N \\ (w^1)^0 & (w^1)^1 & (w^1)^2 & (w^1)^3 & ... & (w^1)^N \\ (w^2)^0 & (w^2)^1 & (w^2)^2 & (w^2)^3 & ... & (w^2)^N \\ ... & & & & & \end{bmatrix},$$

which has determinant

$$det(W) = \prod_{0<i<j<N} (w^i - w^j).$$

Since all w^i are distinct, i.e. $w^i - w^j \neq 0$ as long as $i \neq j$, the determinant of W thus is non-zero, and so it is invertible.

To verify its inverse has the form stated above:

$$\begin{aligned} x[n] &\stackrel{?}{=} \frac{1}{N} \sum_{k=0}^{N-1} X[k] w^{kn(mod\ N)} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} \left(\sum_{l=0}^{N-1} x[l] w^{kl(mod\ N)} \right) w^{kn(mod\ N)} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} x[l] w^{k(l-n)(mod\ N)} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} x[l] \delta((l-n)(mod\ N)) \\ &= \frac{1}{N} \sum_{k=0}^{N-1} x[n] \\ &= \frac{1}{N} N x[n] = x[n] \end{aligned}$$

□

1.2 Zero padding

A circular convolution of two signals, x and y, can be computed using FFT and IFFT:

```
ifft(fft(x).*fft(y))
```

If two signals are aperiodic with arbitrary length, padding them with zeros allow us to calculate their convolution. Suppose

$$x = [1, 3, -1, 3],\ y = [2, -1, 4].$$

First pad each signal with enough zeros to make the common length $N = N_x + N_y - 1$, which in this case would be 6:

$$x = [1, 3, -1, 3, 0, 0],\ y = [2, -1, 4, 0, 0, 0].$$

Then the convolution can be calculated by hand:

$$\begin{aligned} x * y = &(\ 1,\ 3, -1,\ 3,\ 0,\ 0) \times 2 \\ &+ (\ 0,\ 1,\ 3, -1,\ 3,\ 0) \times (-1) \\ &+ (\ 0,\ 0,\ 1,\ 3, -1,\ 3) \times 4 \\ &+ (\ 3,\ 0,\ 0,\ 1,\ 3, -1) \times 0 \\ &+ (-1,\ 3,\ 0,\ 0,\ 1,\ 3) \times 0 \\ &= (\ 2,\ 5, -1, 19, -7, 12) \end{aligned}$$

The command

```
ifft(fft([1 3 -1 3 0 0]).*fft([2 -1 4 0 0 0]))
```

returns the same result.

However, if not enough padding zeros, the result would not be a convolution. Suppose now the padded signals are:

$$x = [1, 3, -1, 3],\ y = [2, -1, 4, 0].$$

the result becomes:

$$\begin{aligned} &(\ 1,\ 3, -1,\ 3) \times 2 \\ &+(\ 3,\ 1,\ 3, -1) \times (-1) \\ &+(-1,\ 3,\ 1,\ 3) \times 4 \\ &+(\ 3, -1,\ 3,\ 1) \times 0 \\ &=(-5, 17, -1, 19) \end{aligned}$$

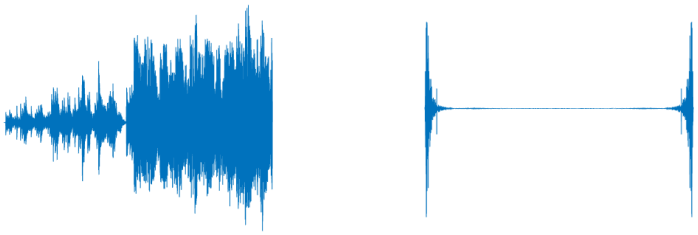
Again, the same result can be get using command:

```
ifft(fft([1 3 -1 3]).*fft([2 -1 4 0])).
```

The reason, from the point of view of how we preform the convolution by hand, it the common length or the period of signal is wrong. From the point of view of using FFT and IFFT is that, the FFT turn the signal from time domain to frequency domain, with same sample number, i.e. in discrete case, length of vector is same in both domain. When not enough zeros are padded, the resolution in frequency domain is too low so that it cannot represent the whole picture of the signal; therefore, after performing the dot product in frequency domain, and then turning the result of dot product back into time domain using IFFT, what comes out is a distorted convolution.

2 Audio filter

We can use FFT help us filter signals. Since filter noise with certain frequency in time domain in practical can be hard to conduct, we can use FFT to turn the signals to frequency domain, eliminate certain frequency, and then inverse FFT to give filtered signals. For example here is an audio clip:

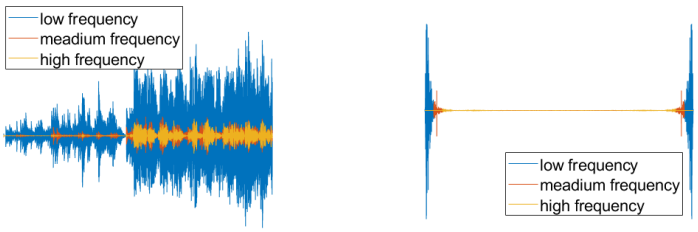


(a) Time domain plot (b) Frequency domain plot

Figure 1: An audio clip

Notice, because of the nature of the Fourier transform, in frequency domain plot, low frequency is at two ends, and high frequency is at the center. From the frequency domain plot, the dominant frequency is less than about 1/15 highest frequency. Therefore, a reasonable low, medium, and high frequency ranges would be $0 < f_{low} < 1/30 f_{max}$, $1/30 f_{max} < f_{medium} < 2/30 f_{max}$, $2/30 f_{max} < f_{high}$.

A MATLAB script that splits the audio file into these frequency ranges is shown in Appendix I, and it outputs audio files:



(a) Time domain plot (b) Frequency domain plot

Figure 2: An audio clip split into three frequency ranges

3 Tidal Frequency

Historically, FFT's and DFT's were first used by Lord Kelvin to analyze and predict tides in the ocean. With much more computing power nowadays, such an analysis can be easily made using MatLab's function for FFT.

Tides are well studied and documented phenomena, with years worth of data available at some of the major scientific databases. In particular, the National Oceanic and Atmospheric Administration (NOAA for short) has data on tides available since the year of 1964, and closely tracks changes. Predictions are made every 6 minutes, and although the verification schedule may change depending on the location, it is done at least 2 times a day.

A sample data ranging from June 2018 to March 2019 was taken from the NOAA station in Anchorage, AK. The information was gathered hourly, so a total of 5567 hours of data are available. The raw data is plotted versus time below, as seen in figure 3.

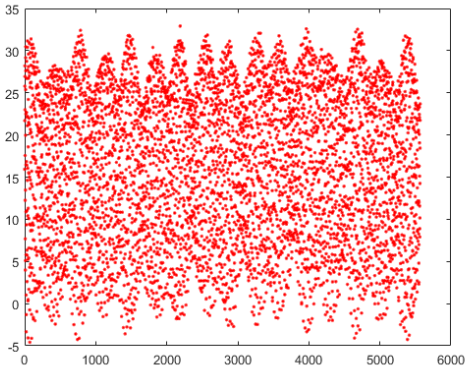


Figure 3: Hourly tidal data for 9 months

As seen in the picture, there is a great deal of periodicity on the data, so an FFT can be done as post-processing to further analyze the frequencies in which it oscillates. In figure 4 below, a filtered one sided FFT is shown.

As made evident by the plot, there is a very distinct peak at 0.08 (unit of 1/hour), which physically refers to the 24 hours and 50 minutes lunar cycle. The period is thus 12 hours and 25 minutes, or half of that cycle, because that's the time the moon takes to go from one side of the planet to the other, and then back. In other words, the maximum and minimum tides are 12 hours and 25 minutes from each other. Other smaller frequencies can be found in figure 4, but are mostly harmonic oscillations from the main solar cycle.

Another notable point happens with a period of about 14.7 days, which can be seen as one of the major points closer to the y-axis. This point in particular refers to the lunar phase cycle, which also has some correspondence with the moon cycles, which are 29.5 days approximately. Although not very evident in

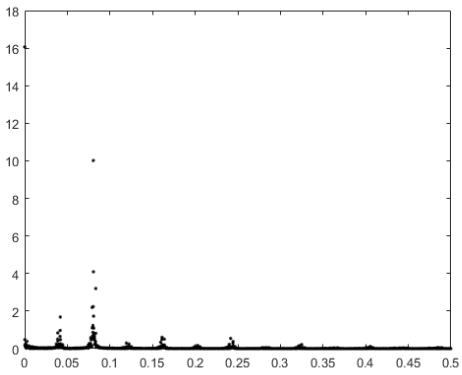


Figure 4: FFT of tidal data for 9 months

figure 4, in other data sets the point stands out more, but also has a somewhat negligible effect when compared to the Lunar orbit around the Earth.

There are, nonetheless, countless factors that may influence tides, such as the distance from the Earth to the Sun, or maybe even solar cycles. Those almost look like noise on the left end of the plot, and that is because the data only comprises a year. Therefore, such effects are too small to actually get observed, and not consistent enough to be noticed. Storms do affect tides, but are not very periodic. However, in some locations, periodic winds can have a noticeable periodic effect on the tides., but it doesn't shown in this data.

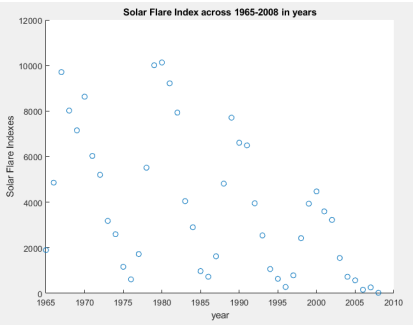
4 Solar Cycle

The solar cycle has been a natural event helping explaining the solar flares frequency. It is a short-lived sudden radiation emission intensity increase neighboring the sunspots. In this part, We obtained a wide range of sample data from NOAA National Centers for environmental information (NCEI), trying to identify the fundamental frequencies of a historical period of flare activities and appearances in the sun. They were first discovered in 1843 by Samuel Schwabe and have been observed and recorded for centuries to study the effects in space caused by the sun. Back to 1800s, scientists monitored solar flares in H-apla wavelength in the chromosphere. Nowadays, their wavelengths can easily be recorded via satellites.

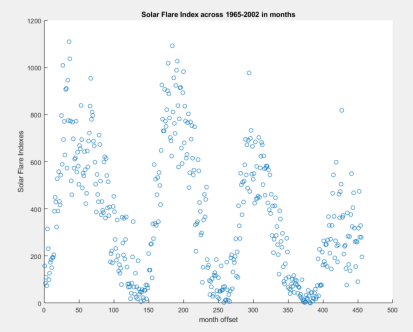
Solar flares are characterized by a rise time of minutes and a decay of tens of minutes. With NCEI archiving approximately 80 stations' solar flare data, from 1938 to the present, we can pick the most representative data from their databases.

The Comprehensive Flare Index (cfi) was developed by Helen W. Dodson and E. Ruth Hedeman, McMath-Hulbert Solar Observatory. They added five measures of flare importance into account and obtained the cfi scale, including

sudden Ionospheric Disturbance importance, H-alpha flare importance, 10.7 cm solar radio flux magnitude, Solar radio spectral type, and Magnitude of 200 MHz flux, etc.

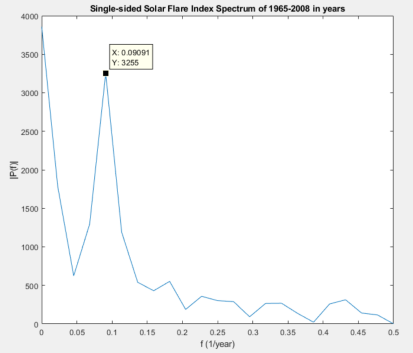


(a) Annual solar flares index

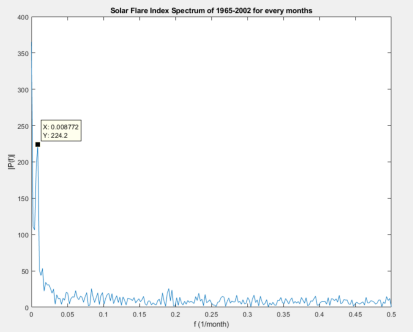


(b) Monthly solar flares index

Figure 5: This figure shows counts of grouped solar flares from Jan 1965 - Dec 2008



(a) Annual solar flare spectrum



(b) Monthly solar flare spectrum

Figure 6: This figure shows single-sided solar flare Fourier Transform Spectrum from Jan 1965 - Dec 2002

After gathering solar flares data from 1965 to 2008 in both monthly and annually counts, With respectively maximum and minimum sunspot counts, we can visually tell in Figure 5 that there is a period around 10 years for the solar

flares. To figure out the exact solar flare index cycle, we can try to perform a single-sided Fast Fourier transform to calculate DFT with MATLAB.

As we can see from Figure 6, after performing fast Fourier transform on the discrete solar flares, we can see the spectrum starting with positive x-axis. Then we need to look for the first several peaks from 0. In fact, for both figures, we can only find one obvious peak except the point at x=0. On the left figure, x=0.09091 claims the peak with y=3255, which indicates for annual solar flare indexes, solar flare has a period of 1/0.09091 = 10.998 years. On the right figure, x=0.008772 stands on the peak y=224.2, which turns out for monthly solar flare indexes, solar flare has a period of 1/0.008772 = 113.99 months, almost nine and a half years. Compared to the scientists' result of around 11 years, we are getting close cycles due to different historical period changes and other measurement influences not taking into consideration.

5 Two dimensional DFT and image denoise

The DFT can be extends to two dimensions, by acting on rows then columns. Apply the idea similar to audio filter in Section 2, we can have a low pass filter to remove high frequency noise. A MATLAB script that accept a gray scale image and remove noise is shown in Appendix III.

With different amounts of noise and different cutoff frequencies, the result is shown below:

(a) Image with noise (b) $f < 0.6f_{max}$ (c) $f < 0.4f_{max}$ (d) $f < 0.1f_{max}$

Figure 7: Noise amount=0.08

(a) Image with noise (b) $f < 0.6f_{max}$ (c) $f < 0.4f_{max}$ (d) $f < 0.1f_{max}$

Figure 8: Noise amount=0.2

References

https://www.ngdc.noaa.gov/stp/solar/solarflares.html
https://tidesandcurrents.noaa.gov/waterlevels.html
https://oceanservice.noaa.gov/education/tutorial_tides/tides05_lunarday.html
https://oceanservice.noaa.gov/education/tutorial_tides/tides06_variations.html

Appendix I

A MATLAB script split audio file into three frequency ranges:

```
%import file and trim a sample
L=2^21;
[x,Fs]=audioread('test.flac',[1,L]);
audiowrite('sample.wav',x,Fs);
y=fft(x(:,1));
%lowpass filter
y1=y;
y1(floor(L/30):L-floor(L/30))=zeros(L-2*floor(L/30)+1,1);
x1=real(ifft(y1));
audiowrite('low.wav',x1,Fs);
%highpass filter
y2=y;
y2(1:2*floor(L/30))=zeros(2*floor(L/30)-1+1,1);
y2(L-2*floor(L/30):L)=zeros(L-(L-2*floor(L/30))+1,1);
x2=real(ifft(y2));
audiowrite('high.wav',x2,Fs);
%mediumpass filter
y3=y;
y3(1:floor(L/30))=zeros(floor(L/30),1);
y3(2*floor(L/30):L-2*floor(L/30))=zeros(L-4*floor(L/30)+1,1);
y3(L-floor(L/30):L)=zeros(L-(L-floor(L/30))+1,1);
x3=real(ifft(y3));
audiowrite('medium.wav',x3,Fs);
```

An example audio clip before and after being processed can be acessed through the link below:
https://drive.google.com/open?id=1OnMcULEYXAqYgl6xyMqfZtESXEXe4KR9

Appendix II

A MATLAB script to read and process tidal data from a xlsxfile.

```
clc
clear variables
close all

%% Read CSV file

data = xlsread('tidal_data_anchorage_061218_031219.xlsx');

%% Hourly data
t = 0:1:5567;
```

```
L = 5567;

Fs = 1;            % Sampling frequency in hours

f = Fs*(0:(L/2))/L;

Y = fft(data(:,1));

P2 = abs(Y/L);

P1 = P2(1:(L/2)+1);

P1(2:end-1) = 2*P1(2:end-1);
figure(1);

plot(f,P1, 'k.')

figure(2);

plot(t,data(:,1),'r.');
```

Appendix III

A MATLAB script add noise to an image and then remove the noise by using FFT:

```
%load an image
M = imread('test.jpg');
n = size(M);
%add noise to the image
M = imnoise(M,'salt & pepper',0.08);
imwrite(M,'noise.jpg')
%denoise in one direction
for i=1:n(1)
    x = fft(M(i,:));
        for j = 1:length(x)
        if (j>floor(length(x)/6))
        && (j<length(x)-floor(length(x)/6))
            x(j) = 0;
        end
    end
M(i,:) = real(ifft((x),n(2)));
clear x
end
%denoise in the other direction
```

```
for i=1:n(2)
    y = fft(M(:,i));
    for j = 1:length(y)
        if (j>floor(length(y)/6))
        && (j<length(y)-floor(length(y)/6))
            y(j) = 0;
        end
    end
M(:,i) = real(ifft((y),n(1)));
clear y
end
%save denoised image
imwrite(M,'denoise.jpg')
```