# CS365 Final Project

Oguz Kaan Elgin
email: okelgin@bu.edu

April 23, 2024

## 1 Introduction

In this project, I analyze time series data of daily observed suns spots from 1818 to 2019 to determine the characteristics of the sun sun spot numbers in time. I find this problem important as I believe the sunspot dataset is a good representation of a time series with periodic behaviour. My interest in the topic stems both from personal interest and my curiosity regarding the analysis of a dataset with nonlinear time dependence. I began the project with the intention of finding periodic behaviour in the number of sun spots with respect to time. After initial analysis of the dataset, I used the discrete Fourier transform in order to shift the data into the frequency domain and find predominant frequencies and their respective coefficients. The resulting analysis yielded information about prime frequencies of the Sun's activities, such as the approximate solar cycle. All code written and datasets used can be referenced in my GitHub[1].

## 2 Related Work

Although my analysis primarily focuses on Fourier series, there have been other methods of analysing time series data as well. There are forecasting methods such as exponential smoothing[2] and ARIMA[2] which also deal with nonlinear time series and are primarily used for predicting future trends. Although these methods also fall in line with my analysis of sunspots, they are more general purpose methods while the sunspot dataset seems to have clear periodic behaviour. A more simpler model, which is also a part of ARIMA, is the Auto regressive (AR) model. Since the data I use is not stationary thought (having fluctuating statistical properties) I calculate auto-correlation after my Fourier transform and filtering as a tool of assessing my previous choices. Overall, My focus on this project will be Fourier transforms and what what their outputs show about a time series dataset.

# 3 Resources

The language I used for data analysis was python3. Specifically, the pandas, numpy, scipy, matplotlib.pyplot, seaborn, statsmodels libraries were utilized. The data set I am using is from kaggle[3], and contains information such as an index for each day since 01/01/1818, the date of each index, the number of sun spots observed each day, the standard deviation of the observed sun spots each day, and the number of observations made to find the sun spot number each day. All of the computations and analysis was done on my personal laptop with, 16GB of RAM and a AMD Ryzen 9 4900HS CPU.

# 4 Method

## 4.1 Fourier Series and Transform

The theory of Fourier transforms is derived from the Jean Baptiste Fourier's fundamental proof that any periodic function $f(t)$ with a period of $\tau$ can be expressed as the linear combination of sinusoidal functions[4] as follows:

$$f(t) = \sum_{0}^{\infty} [a_n cos(n\omega t) + b_n sin(n\omega t)] \;\; with \;\; \omega = \frac{2\pi}{\tau}$$

Using this definition, we can then calculate the $a_n$ and $b_n$ coefficients using the fact that sin and cos are orthogonal functions to be[4]:

$$a_n = \frac{2}{\tau} \int_{-\frac{2}{\tau}}^{\frac{2}{\tau}} f(t) cos(n\omega t) \;\; and \;\; b_n = \frac{2}{\tau} \int_{-\frac{2}{\tau}}^{\frac{2}{\tau}} f(t) sin(n\omega t)$$

It is important however, to remember Euler's formula: $e^{i\theta} = cos(\theta) + isin(\theta)$

Using this relation and the formula's for $f(t)$, $a_n$ and $b_n$, we can get an exponential form of Fourier transforms as follows[5]:

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{i\omega t} \;\; where \;\; c_n = \frac{2}{\tau} \int_{-\frac{2}{\tau}}^{\frac{2}{\tau}} f(t) e^{-i\omega t}$$

Using these two relationships and taking our the limit $n- > 0$ (This step is not straightforward but I have skipped most of it so as to not surpass the assignment page limit) we get[5]:

$$f(t) = \int_{-\infty}^{\infty} f'(k) e^{ikt} dk \;\; and \;\; f'(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} d\omega$$

The overall meaning of the function $f'(\omega)$ is that it represents how commonly a frequency occurs in the function $f(t)$. $f(t)$ is said to be in the time domain while $f'(\omega)$ is in the frequency. For sunspots, this would allow us to find underlying

frequencies. The issue, is though, that I have no knowledge of f(t), and my data is not continuous.

## 4.2 Discrete Fourier Transforms

Given the fact that the sunspot data I have is a finite set of discreet values, the approach to integrate it with over infinity multiplied by a complex exponential fails. This is where the Discrete Fourier Transform (DFT) comes to play. The DFT is an approximation of Fourier transforms for discrete cases[6]. Given a sequence of real inputs $x_n$, analogous to f(t), it returns a sequence of complex outputs $X_k$, analogous to $f'(\omega)$, representing the frequency domain of the input $x_n$.

The DFT of a list of x can be defined as[6]:

$$x(n) = \frac{1}{N}\sum_{n=0}^{N-1} X(\omega_k)e^{i\omega_k t_n} \;\; and \;\; X(k) = \sum_{k=0}^{N-1} x(n)e^{-i\omega_k t_n}$$

Letting $e^{\frac{-i2\pi}{N}} = W_k$ we can see:

$$X(0) = x(0)W_N^0 + x(1)W_N^0 + ...x(n)W_N^0$$

$$X(1) = x(0)W_N^2 + x(1)W_N^3 + ...x(n)W_N^{(N-1)}$$

$$...$$

$$X(N-1) = x(0)W_N^0 + x(1)W_N^{(N-1)} + ...x(n)W_N^{(N-1)^2}$$

This information encodes a matrix operation where X is a vector of $X_k$, x is a vector of $x_n$ and the multiplying factor is a matrix $W_N^{nk}$ as:

$$X = W_N^{nk}x$$

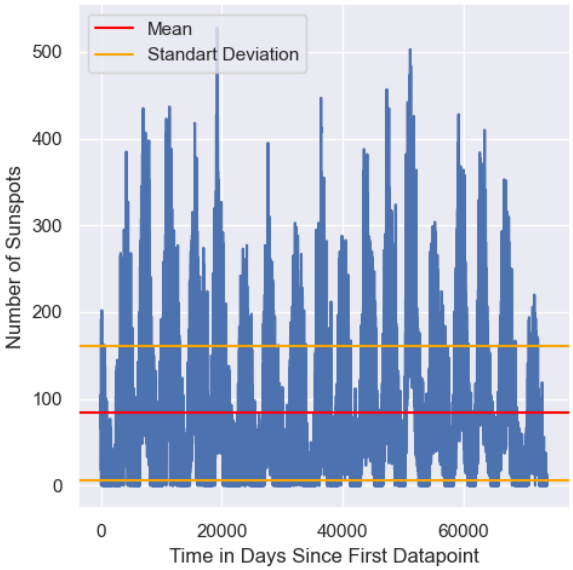
This matrix operation can actually be calculated by a computer. The operation does grow with large number of elements N, and so more efficient algorithms (such as the Fast Fourier Transform) to compute values for $X(k)$ have been created. In my project I will be using the numpy.fft.fft() function in order to find the Fourier transform of my sunspot dataset.

# 5 Experimental Results

## 5.1 Preliminary Adjustments

An initial issue to consider before any analysis of the sun spot dataset was that of missing sun spot observations (denoted by a 'Number of Sunspots' value of -1). These unobserved values exist up to index 11313 out of 73717, from which point on no unobserved data points remain. This implies that the missing values are not random and are more likely to be found in the first timestamps of the data. However, as these points are spread out among the first 11313 points, approximating their sun spot values to be similar to their neighbors would allow us to retain some information on the trends of the graph.

Since the dataset starts out with some missing values, I resolved to use the "Next Observation Carried Back"[7], method of imputation. So the missing data values were fitted with their closest next neighbors. After filling missing values, the plot of the sunspots against time is as follows:




Looking at the data, there seems to be some periodic behaviour, however due to the high amount of noise that is present throughout, it is difficult to make it into a clear periodic function.

In order to analyze the distribution of a random variable in the time series, the skew and kurtosis of the dataset can be calculated. The skew $\tau$ of a random variable is the third central moment of the random variable divided by its standard deviation cubed[8]:

$$\tau = \frac{\mu^3}{\sigma^3} = \frac{E[(x-\mu)^3]}{\sigma^3}$$

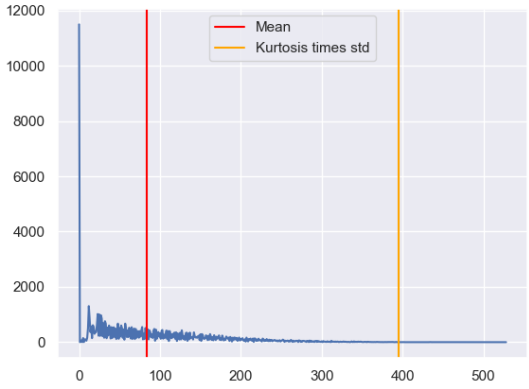
The skew of a random variable determines how symmetric it is with respect to its mean. Symmetric functions (such as the standard normal distribution) have a skew of 0, while positive and negative values imply the distribution is tilted toward one side.

The kurtosis $\kappa$ is related to the fourth central moment of the variable and is "informative about the tail behaviour of the series."[8] This is due to the fact that values close to the mean hardly effect the fourth power of the data, and therefore it is primarily dominated by far values. The definition is[8]:

$$\kappa = \frac{\mu^4}{\sigma^4} = \frac{E[(x-\mu)^4]}{\sigma^4}$$

This tail behaviour generally relates with outliers in the distribution, meaning how spread out from the peak a distribution is. The kurtosis of a Gaussian distribution is 3[9], which is why values outside 3 standard deviations of the mean are considered outliers in a Gaussian distribution. The kurtosis of the sunspot data will be important in determining what kind outliers exist.

Using the stats.skew() and stats.kurtosis() functions of the scipy library on the sunspot data yields: $\tau = 1.112933530808$, and $\kappa = 4.004001103833$. This indicates that the distribution of a single sunspot data point is skewed to the left by a small amount, and the tails reach further than a regular Gaussian distribution. A graph of the sunspot number against its occurrences is given below:



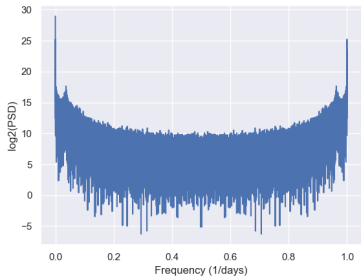

As expected, the distribution is clearly not symmetric around its mean, (especially due to the large number of 0 sunspot observations), and the kurtosis multiplied by the standard deviation seems to include most of the graph's information.

It is now possible to move on to Fourier analysis to get an understanding of the periodic structure of the time series.

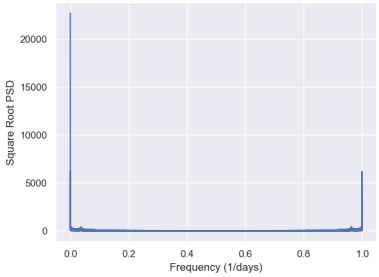
## 5.2 Sunspot data in Frequency domain

With this information, it is possible to find the Fourier transform of the dataset with the numpy function fft.fft(). The code for the Fourier transformation is given in my GitHub link[1].

After the initial transformation, the transformed datapoints are complex number in with the same dimensions as the original input data. The transformed datapoints are then multiplied by their complex conjugates to yield a power spectral density (PSD). This is done to obtain non-complex frequency density values. Due to this multiplication, the density norms are effectively squared, and so PSD values explode at common frequencies. The plot of the PSD is given below:




(a) PSD Log




(b) PSD Root

As I previously stated, the amplitudes of dense frequencies explode in the PSD (as expected, given that we are attempting to find dominating frequency peaks). Due to this, the graph (a) above shows $log_2(PSD)$ to make visualization easier. Graph (b) is given for a more clear view of the frequency density peaks.

With all of this at hand, and the use of the Fourier series approximation using the coefficients obtained after the transformation, it is possible to eliminate noise frequencies in order to arrive at a cleaner function with a handful of sinusoidal elements depending on a few frequencies.

## 5.3 Noise removal

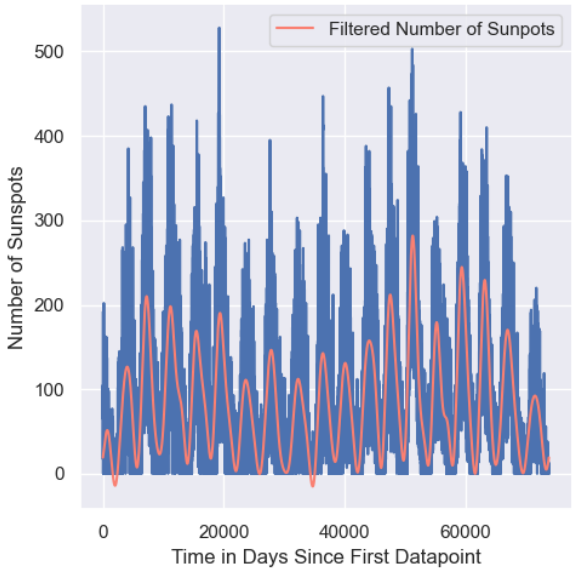
There are many methods of removing noise from data, but in the context of this project, amplitude thresholding is a viable option, given that we already have the frequency density series of the time series. There are again, many methods of setting the amplitude threshold[10] itself. However, since the only knowledge I have at this point is about the distribution of a random element within the set, the approach I will take is eliminating outliers according to the original data.

The kurtosis value previously calculated can serve as a base to estimate to how many standard deviations away from the norm should be excluded in the filtered data. Since the original data is slightly leptokurtic, more data must be included in the threshold than 3 standard deviations. To account for this, I have calculated a threshold based on the standard deviation $\sigma_{freq}$ and mean $\mu_{freq}$ of the frequency domain time series, and the kurtosis value $\kappa$ of the time series. I set the threshold as:

$$\mu_{freq} + \kappa \cdot \sigma_{freq}$$

The filtering function is given in code[1].

Now that the data is filtered, an inverse Fourier transform gives back a time series in the time domain once again. The graph of the new and old versions of the plots are superposed in the figure below:




This new graph does seem to capture the essence of the fluctuations dominating the initial data. Also, while the initial function looks very chaotic, the filtered version seems to be a more clear linear combination of sinusoidal functions.

# 6 Conclusion

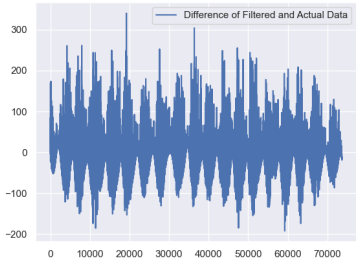
## 6.1 Solar Cycle Estimation

Although I was able to use the Fourier coefficients in order to reconstruct the graph as a filtered function of time, since one of the main assumptions we made while computing the Fourier transform was that the function was periodic, the

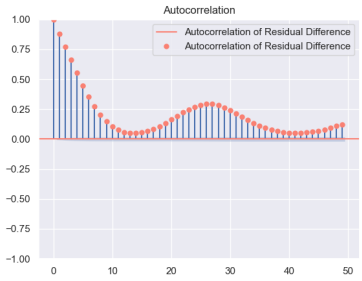
actual function calculated has a period of 73717 days. Despite this, the transform did yield dominant frequencies of the aggregate wave. Using the np.argmax function on our Fourier coefficient array and taking the corresponding frequency $\omega$, we get the most dominant frequency in the wave[1]. This turns out to be 0, which makes sense given that the function is shifted up, having a min of close to zero, so this term acts as a $\alpha cos(0) = \alpha$ vertical raise. Then to find oscillatory behaviour the second most dominant frequency is $w_1 = 2.57739 \cdot 10^{-4}/day$. This is surprisingly close to the accepted solar cycle period[11] of 11 years. Given that 11 years is $365 \cdot 11 = 4015$ days, converting into $1/days$ we get: $2.49066 \cdot 10^{-4}/days$. Given that the calculated value has an error of 3.48%, it can be said that although the transform is made up of thousands of frequencies, the leading Fourier coefficients determine long term behaviour of the sunspots.

## 6.2 Potential Issues in Noise Reduction

Since my goal was not only to fit the sunspot curve but also filter out the noise in this analysis, the difference between the fitted sunspot curve and the original data should ideally yield some form of uncorrelated white noise. It is possible to see if my filter threshold was effective by finding the auto-correlation relationship of the difference data. This merely relates each item $y_t$ to some other lagged item $y_{t-n}$. I then plot the lagged correlation of the difference data with itself up to a lag of 50. To achieve this I simply used the statsmodels acf function on the difference data[1]. The results are displayed below.




(a) Diff




(b) ACF

Given that the auto-correlation graph displays a relationship between different lags, instead of falling off quickly after the first lags, it seems as though the remaining difference is not wholly noise. This implies that the kurtosis strategy I previously employed was not a strict enough criteria for determining the amplitude threshold I set for the frequency domain of the dataset. In the future more analysis could be done on setting a better threshold for the domain by employing other noise reduction techniques.

# References


[1] O. K. Elgin, "Github link to project code." `https://github.com/okelgin/CS365-Project-Code/`.

[2] D. Gooijer., J. G, Hyndman., and R. J, "25 years of time series forecasting," *International journal of forecasting*, vol. 22, no. 3, 2006-1.

[3] Abhinand05, "Daily sun spot data (1818 to 2019)." `https://www.kaggle.com/datasets/abhinand05/daily-sun-spot-data-1818-to-2019/`, 2020.

[4] J. Taylor, *Classical Mechanics*. G - Reference,Information and Interdisciplinary Subjects Series, University Science Books, 2005.

[5] D. Morin, "Waves(draft) chaper 3: Fourier analysis." November 2005.

[6] A. V. Anand, "A breif study of discrete and fast fourier transforms.," *University of Chicago Mathematics*, 2010.

[7] A. Abulkhair, "Data imputation demystified | time series data." `https://medium.com/@aaabulkhair/data-imputation-demystified-time-series-data-69bc9c798cb7`, June 2023.

[8] J. Bai and S. Ng, "Tests for skewness, kurtosis, and normality for time series data," *Journal of Business  Economic Statistics*, vol. 23, no. 1, pp. 49–60, 2005.

[9] K. Najim., E. Ikonen., and A.-K. Daoud, *Stochastic Processes: Estimation, Optimization  Analysis*. Butterworth-Heinemann, 2004.

[10] Z. Zhang. and T. G. Constandinou, "Selecting an effective amplitude threshold for neural spike detection," *IEEE Xplore*, July 2022.

[11] D. H. Hathaway, "The solar cyle," *Living Reviews in Solar Physics*, September 2015.