JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

COMPARATIVE ANALYSIS OF DISCRETE FOURIER TRANSFORM AND FAST FOURIER TRANSFORM USING ANALYSIS OF SUNSPOT DATA.

[1] **Dr. Rohan Jadhav**, [1]**Machado Prasun Francis and** [1]**Tambare Aqsa Husain[1],**
[1]Assistant Professor, [2,3] Postgraduate Students
[1]Department of Physics,
[1]St. Xavier's College, Mumbai, India

Abstract: This work presents comparative analysis of Discrete Fourier transform (DFT) and Fast Fourier transform (FFT) through the study of the sunspot data. The power spectrum of sunspot data was obtained by DFT, FFT using in-built function of NumPy (numpy.fft.fft()), FFT using author written code based on radix-2 case of Cooley-Tukey algorithm (for dataset of 305 points), FFT using author written code based on radix-2 case of Cooley-Tukey algorithm for dataset with zero padding to make its size equal to power-of-two. This work indicates that the DFT and FFT using in-built function of NumPy gives identical results. On the other hand, radix-2 Cooley-Tukey algorithm gives some difference in the results when the dataset size was not of power-of-two. Addition of zero padding results in the considerable improvement of the results. FFT using in-built function of NumPy is very much efficient in the execution w.r.t. the authors code due to its optimized algorithm.

IndexTerms **- Fourier analysis, Wolf number, Fast Fourier transform (FFT), Discrete Fourier transform (DFT), power spectrum, radix-2 Cooley-Tukey algorithm, sunspots.**

I. INTRODUCTION

Fourier Transform is a powerful mathematical method that converts a function from its temporal/spatial domain to the frequency domain. Fourier transform is a natural extension of the Fourier Series when the time period of the function extends to infinity. It takes a function $f(t)$ or $f(x)$ and transforms it to another function $F(\omega)$ which is essentially decomposing the original function into its constituent frequency functions [1]. This insight into the temporal/spectral frequency components enables enhancement and modifications of the original function that is useful across a wide variety of fields like physics, engineering, digital signal and image processing, etc.

In all practical applications, the data is always in discrete form. As the Fourier transform can handle only continuous data, a numerical method is derived from the Fourier transform called Discrete Fourier transform (DFT). DFT takes a finite sequence of sampled function or experimental data, $f(x_n) = f_n$, and transforms it into the sequence of the complex-valued function of the frequency $F(\omega_k)$ of the same length. Even though it is considered one of the important computational tools, its direct applications are limited due to its complexity in handling long datasets as the number of calculations in the DFT increases as N^2, where N is the length of the dataset [2].

Fast Fourier transform is an optimized version of a DFT which drastically reduces the computational overhead using symmetry in the calculations. The number of calculations required in FFT to transform the dataset of length N is equal to $Nlog_2N$ [2]. This efficiency gain in the case of FFT has made not only processing of large datasets practical but also revolutionized numerous technologies by facilitating real-time processing and analysis. One of the main limitations of FFT is that it favors data sets whose length corresponds to powers of 2. The development of Fast Fourier Transform (FFT), primarily attributed to the work of James Cooley and John Tukey in 1965, marked a significant breakthrough in computational techniques for performing Fourier analyses.

A comparative analysis of DFT and FFT is essential in the nuanced understanding of both methods that can aid in selecting and applying the appropriate method for specific applications. In this work, we have done a comparative study of DFT and FFT by analysing sunspot data.

Sunspots are the regions (spots) on the surface of the sun that appear darker as compared to surrounding areas. This is associated with a local concentration of high magnetic flux that inhibits convection, effectively reducing the temperature of the region as compared to other parts of the surface of the sun. Sunspots generally appear as pairs of opposite magnetic polarity. In general, sunspots consist of a darker region called the umbra surrounded by a lighter region called the penumbra [3]. Typically, the

sunspot sizes vary from 16 km to 160,000 km, and their sizes contract or expand as they move over the surface of the sun [4]. Due to their correlation with other solar events, they can be used to predict space weather, the state of the ionosphere, and the factors affecting satellite communications. The observations of the sunspot data by the astronomers dates back to 1609. However, in 1848, Rudolf Wolf developed a method to characterize solar activity by counting the number of isolated spots and clusters of spots on the surface of the sun. The quantity that indicates the number of sunspots and the group of sunspots is called as the Wolf number or relative sunspot number [5].

II THEORY:

2.1 Fourier Transform:

Fourier Transform of the function $f(t)$ to the function of the frequency $F(\omega)$ is given by the following equation

$$F(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-i\omega t}dt \text{----------(1)}$$

Inverse Fourier transform which gives back the original function $f(t)$ is given by

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(\omega)e^{i\omega t}d\omega \text{ ----------(2)}$$

Both the complex-valued function F(ω) and the mathematical operation in equation 1 are denoted by the term "Fourier transform". $1/2\pi$ factor is used for normalization and can be shifted to equation 1 or can be divided in both the equations as $1/\sqrt{2\pi}$.

2.2 Discrete Fourier Transform:

In practical applications data is always collected as discrete datasets. Additionally, in many cases, functions are sampled at discrete values of independent variables. If the function/experimental data is sampled between interval 0 to T ($f(t)$: $f_0, f_1, \dots\dots, f_{N-1}$ for time t_n: $t_0, t_1, \dots\dots, t_{N-1}$ respectively) and the sampling is equidistant in this interval, then equation 1 can be written as

$$F_k = \sum_{n=0}^{N-1} f_n\, e^{-i\omega_k n\Delta t} \text{ for } k = 0 \text{ to } N-1 \text{----------(3)}$$

where N is the total number of sampled points, $\Delta t = T/N$ is the sampling interval for the independent variable t_n and the f_n is the value of the function sampled at point n i.e. at the value of the independent variable $t_n = n\Delta t$ [2, 6].

The first value of the ω_k is ω_o and it corresponds to the DC component of the signal. As data is only sampled between 0 to T, we can consider the data is periodic with time period T. So, the next possible value of the angular frequency is $\omega_1 = \frac{2\pi}{T} = \frac{2\pi}{N\Delta t}$. The next higher angular frequency will be $\omega_2 = 2\frac{2\pi}{T} = 2\frac{2\pi}{N\Delta t}$ with period T/2. We can generalize the relationship as $\omega_k = k\frac{2\pi}{N\Delta t}$ [6]. So, the equation 3 becomes

$$F_k = \sum_{n=0}^{N-1} f_n\, e^{-ik\omega n} \text{ for } k = 0 \text{ to } N-1 \text{ ----------(4)}$$

where $\omega = 2\pi/N$. Equation 4 is called Discrete Fourier transform (DFT) of the discrete function f_n. As it can be seen from equation 4, DFT requires N^2 complex operation to find F_k for all possible values of k.

2.3 Fast Fourier Transform:

One of the most widely used FFT is the Cooley-Tukey algorithm. The central idea of this algorithm is to use divide and conquer algorithm that recursively breaks down a DFT into smaller DFTs with multiplication of complex roots of unity ($e^{-ik\omega}$) called twiddle factor. One of the simplest ways to implement the Cooley-Tukey algorithm is to divide the transform into two smaller transforms (DFTs) of even position points and odd position points. The algorithm repeats this process of division until it reaches to the transforms (DFT) of individual data points. This implementation is called as radix-2 decimation in time (DIT) FFT.

The radix-2 decimation in time (DIT) FFT is carried out as follows. The summation in equation 4 can be divided into two separate summations for even terms (2n) and odd terms (2n+1) of the dataset. This is given as

$$F_k = \sum_{n=0}^{N/2-1} f_{2n}\, e^{-ik\omega 2n} + \sum_{n=0}^{N/2-1} f_{2n+1}\, e^{-ik\omega(2n+1)} \text{ for } k = 0 \text{ to } N-1 \text{----------(5)}$$

$$F_k = \sum_{n=0}^{N/2-1} f_{2n}\, e^{-ik\omega 2n} + e^{-ik\omega}\sum_{n=0}^{N/2-1} f_{2n+1}\, e^{-ik\omega 2n} \text{ for } k = 0 \text{ to } N-1 \text{----------(6)}$$

Each summation in equation 6 is a DFT (the second term is a DFT multiplied by a factor $e^{-ik\omega}$called a twiddle factor). Because of the periodicity of the complex exponents ($e^{-i(k+N/2)\omega} = -e^{-ik\omega}$ and $e^{-i(k+N/2)\omega 2n} = e^{-ik\omega 2n}$), we can also calculate $F_{k+N/2}$ term from these smaller DFTs. So, we can summarize the evaluation of transform as

$$F_k = \sum_{n=0}^{N/2-1} f_{2n}\, e^{-ik\omega 2n} + e^{-ik\omega}\sum_{n=0}^{N/2-1} f_{2n+1}\, e^{-ik\omega 2n} \text{ for } k = 0 \text{ to } N/2-1 \text{ ----------(7a)}$$

$$F_{k+N/2} = \sum_{n=0}^{N/2-1} f_{2n}\, e^{-ik\omega 2n} - e^{-ik\omega}\sum_{n=0}^{N/2-1} f_{2n+1}\, e^{-ik\omega 2n} \text{ for } k = 0 \text{ to } N/2-1 \text{ ----------(7b)}$$

This effectively results in decomposing the single DFT with N^2 operations to two smaller DFTs with $(N/2)^2$ operations. This process is applied repeatedly/recursively till we reach DFT of a single point. This decomposition results in limiting the total operations to $Nlog_2N$ [2, 7]. As in every recursion the data set is divided into half, radix-2 DIT FFT is most accurate when dataset has a size of power-of-two. One of the simplest ways to handle data with other sizes is by zero padding the data to get the power-of-two size.

2.4 Power spectrum:

As Fourier coefficients are difficult to interpret directly, the power spectrum of the Fourier transform is used to understand the results. It symmetrically distributes the energy associated with the data across its various frequency components. The power spectrum is plotted as a graph depicting how much of the signal's power falls within the specific frequencies. The power in the k^{th} real harmonic of the dataset f_n is given by power spectral density $p_k = 2|F_k|^2$ [2]. The Power spectrum is a plot of p_k as a function of frequencies $k\omega$.

2.5 Sunspots:

Approaching the sunspot number is not a straightforward task as sunspots greatly vary in size and also many are coalesced together at their edges to form clusters. According to the method devised by Rudolf Wolf, a Wolf sunspot number (R) depends on the number of isolated spots (S) and clusters of spots (G) on the sun's surface [5].

$$R = K(10G + S) \text{ ----------(8)}$$

where K is a scaling factor that varies with the observer and is also called the observatory factor or the personal reduction coefficient. This factor takes into account the differences in recorded values due to differences in experimental setups, personal experience and other factors amongst the observers. The sunspot data is provided by many prominent organizations around the world like National Oceanic and Atmospheric Administration (NOAA), Solar Influences Data Analysis Center (SIDC) at the Royal Observatory of Belgium, NASA, the National Solar Observatory, the Solar Heliospheric Observatory (SOHO), etc.

III METHODOLOGY

3.1 Sunspot

In this work we have taken sunspot number data from the reference [8]. Data at this source is available from the year 1700 to the year 2004 in the yearly sunspot number form. This data was obtained by the source from the National Geophysical Data Center - NOAA Satellite and Information Service. This data is plotted using the python-3 programming language in Google Colab. The resultant plot is shown in the figure (1).

3.2 DFT and FFT

The programs were written in python-3 programming language and executed in Google Colab. The computation part was carried out in four parts. In the first part, equation 4 was used to get the DFT of the sunspot data. In the second part, the Cooley-Tukey FFT algorithm for the radix-2 DIT case was implemented [7]. In this process, data was divided into even and odd position points recursively and the Fourier transform was calculated with the help of equations 7a and 7b. As the sunspot data was of the length 305 points, in the third part, the calculations were repeated with zero padding to increase the dataset size to 512. In the last part, the FFT was carried out by the in-built function of numpy.fft.fft() of the NumPy module in the python-3 [9]. For all four cases, first the results were plotted for the imaginary part of F_k against the real part of F_k and then power spectra (p_k vs $k\omega$) were plotted.

timeit() module was used for comparison of the execution time for DFT, in-built FFT and FFT [10]. *timeit* is a useful tool for measuring the execution time for small pieces of code. It minimizes the effect of related background processes and system tasks to provide more accurate results. It is especially useful when we need to compare different snippets of code.

IV RESULTS AND DISCUSSION:

Data taken from the source [8] is plotted in the figure (1).

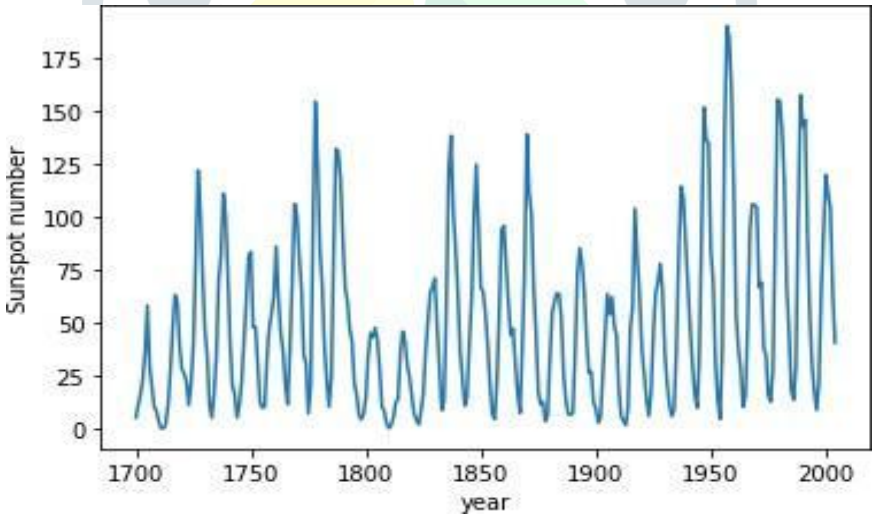

Figure 1: The yearly Wolf sunspot numbers from 1700 to 2004 [8]

Fourier Coefficients of the sunspots number data were calculated using in-built FFT function of the NumPy module of Python- 3, DFT, FFT (radix-2 Cooley–Tukey algorithm) and FFT (radix-2 Cooley–Tukey algorithm) with zero padding. The figure 2 shows the plots of the imaginary parts of Fourier Coefficients against the corresponding real parts for all the four cases.

The plot of the power spectra for all the four cases were also calculated and plotted as a function of frequency (measured in cycles per year) (see Figure 3)

Average time taken to count a single loop in the Fourier transformation by different methods is as follows:

- By numpy.fft.fft() = 16.4 µs ± 3.04 µs/loop
- By DFT = 14.8 ms ± 2.58 ms/loop
- By FFT (radix-2 Cooley-Tukey Algorithm) = 7.98 ± 2.74 ms/loop

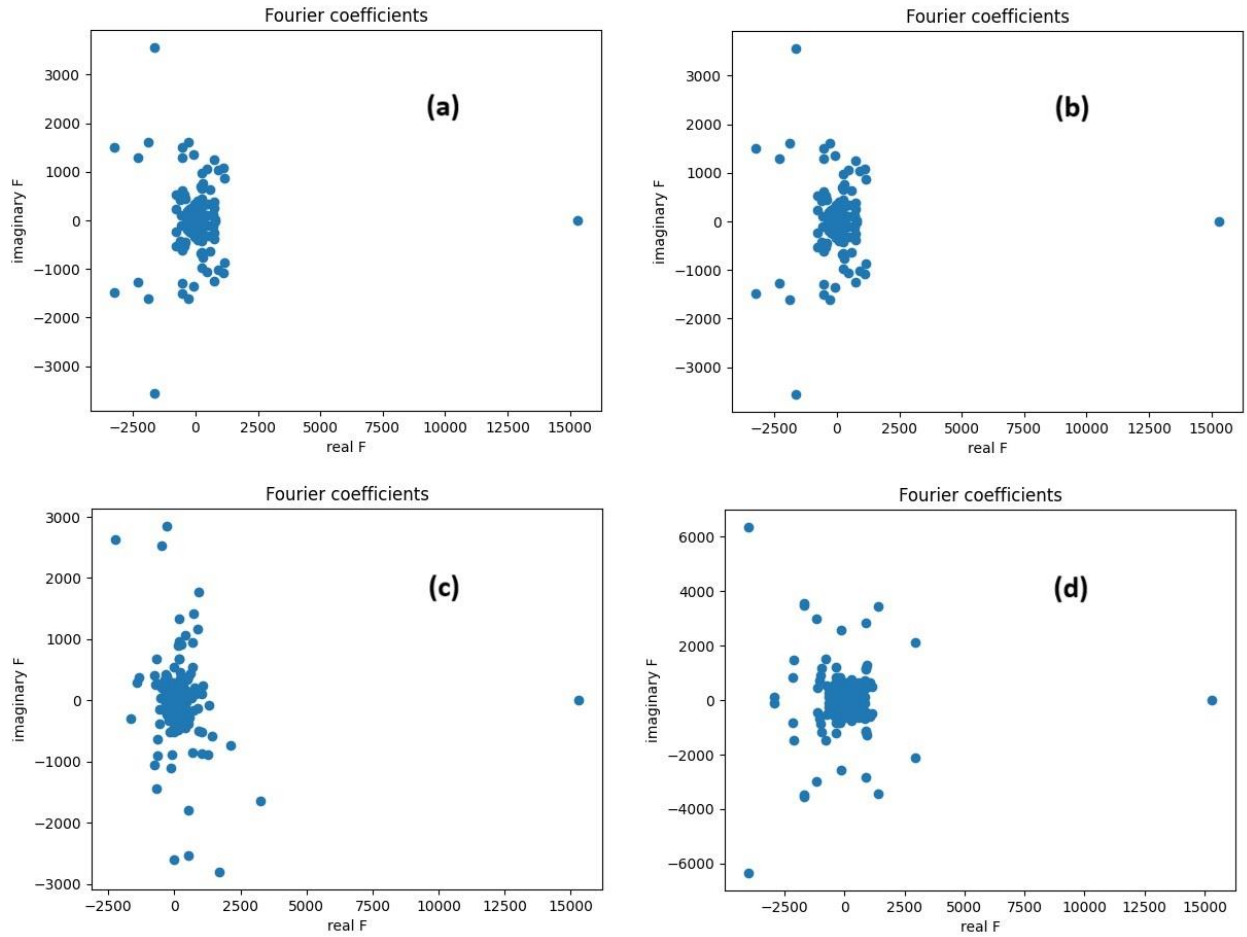

Figure 2: Fourier coefficients plotted using (a) in-built FFT function (b) DFT function (c) FFT (radix-2 Cooley–Tukey algorithm) (d) FFT (radix-2 Cooley – Tukey algorithm) with zero padding.

The table 1 shows the time periods in years for solar cycle and position of the peak corresponding to highest power for all the four cases.

Table 1: Time period of sunspot cycles as measured by DFT, in-built FFT module of NumPy, FFT (Cooley-Tukey Algorithm) with and without zero padding

Method	Highest power corresponds to frequency (cycle/year)	Time period (years)
DFT	0.0885245	11.296
FFT (NumPy module)	0.0885245	11.296
FFT (radix-2 Cooley-Tukey Algorithm)	0.0819672	12.200
FFT (radix-2 Cooley-Tukey Algorithm) with zero padding	0.0859375	11.636

As most dominant peak (figure 3) is located around 0.08 in all four cases, these results are consistent with Wolf's estimate of 11 years of solar cycle [2, 11] and also the sunspot activity peaks about once every 11 years. On comparing DFT, FFT (in-built) and FFT (radix-2 Cooley-Tukey Algorithm), we observed that DFT is the slowest as compared to the rest.

The Fourier coefficients and the power spectrum obtained from DFT and in-built FFT module (figure 2 (a) and (b), figure 3 (a) and (b)) are quite identical to each other. Values of solar cycle time period and frequency related to the highest power as calculated by these methods (table 1) are also identical. In the case of the FFT method using the radix-2 Cooley-Tukey algorithm (without zero padding) the Fourier coefficients and the power spectrum deviate from the previous methods. Also values of solar cycle time period (12.2 years) and frequency related to highest power (0.0819672) as calculated by this method deviates from the results obtained by the previous methods. The main reason for this deviation is attributed to the size of the dataset (305 data points) which is not a power-of-two. The issue is addressed by increasing the data size to power-of-two by adding zeros in the dataset (zero padding). Figures 2(d) and 3(d) show the results obtained by zero padding. The improvement in the results can be seen clearly. The values of solar cycle time period and frequency related to the highest power as calculated by FFT (radix-2 Cooley-Tukey algorithm) with zero padding is in better match with the DFT and in-built FFT as compared to FFT (radix-2 Cooley-Tukey algorithm). The in-built FFT module uses different algorithms like zero padding, Mixed-Radix FFT algorithms and Bluestein's algorithm (Chirp Z-Transform). It generally handles non-power-of-two sizes by automatically choosing the most suitable and efficient algorithm, often using mixed-radix FFT for many practical sizes.

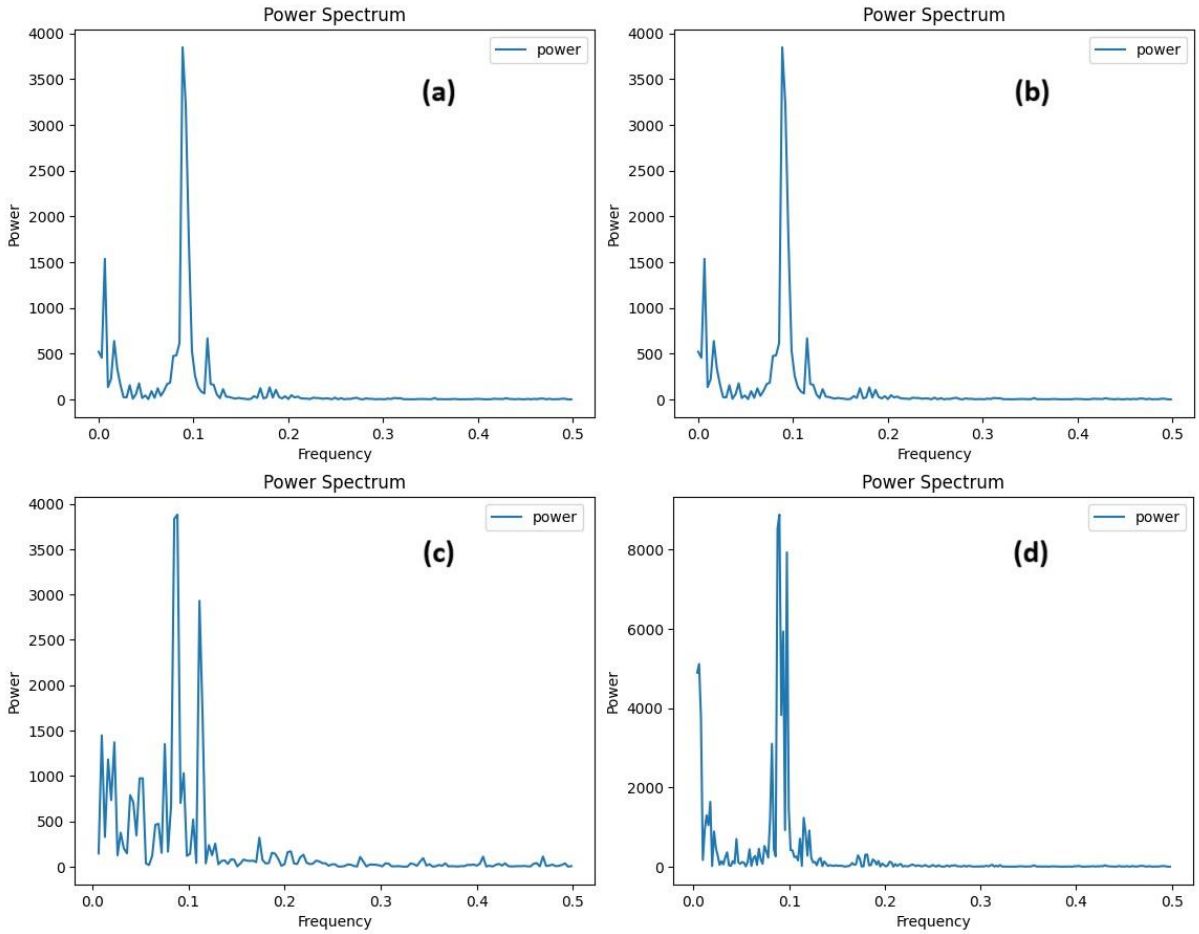

Fig. 3: Power spectrum plotted using (a) in-built fft function (b) DFT function (c) Cooley – Tukey algorithm (d) Cooley – Tukey algorithm with zero padding.

It can be seen clearly that DFT is the slowest method as it takes 14.8 ms per loop as compared to FFT (radix-2 Cooley-Tukey algorithm) (7.98 ms per loop) and in-built FFT module (16.4 µs per loop). The in-built FFT module is the fastest one of all. This efficient implementation is attributed to many factors. The NumPy functions are implemented in C and Fortran, which are compiled directly into the machine code. This makes them significantly faster as compared to our programs written in python-3. 'NumPy' uses vectorized operations that use architecture capabilities of modern processors, that allow many operations to be performed simultaneously [12]. The NumPy uses an iterative FFT algorithm that reduces the computational overhead as compared to the recursive FFT algorithm used by us. NumPy uses in-place algorithms that reduce memory usage and improves speed as compared to array slicing operations (to separate even and odd data points) used by us.

V REFERENCES:

1. https://en.wikipedia.org/wiki/Fourier_transform
2. Chapra, S. and Canale, R. (2006), Numerical Methods for Engineers. McGraw-Hill higher education, McGraw-Hill.
3. Ostlie, D. A., Carroll, B. W. (2017), An Introduction to Modern Astrophysics, United Kingdom: Cambridge University Press.
4. https://en.wikipedia.org/wiki/Sunspot
5. https://en.wikipedia.org/wiki/Wolf_number
6. https://phys.libretexts.org/Learning_Objects/Demos_Techniques_and_Experiments/Fourier_Transform_A_Brief_Introduction
7. https://en.wikipedia.org/wiki/Cooley%E2%80%93Tukey_FFT_algorithm
8. https://linuxgazette.net/115/andreasen.html
9. https://numpy.org/doc/stable/reference/generated/numpy.fft.fft.html
10. https://docs.python.org/3/library/timeit.html
11. https://en.wikipedia.org/wiki/Solar_cycle
12. https://jakevdp.github.io/blog/2013/08/28/understanding-the-fft/