HIGH PERFORMANCE COMPUTING

Design Activity - 1

Topic - Using Discrete Fourier Transform for finding the normalized frequency of a random wave and parallelizing the code using Openmp.

CONTENTS

Bhaavanaa Thumu - CED17I021

1. Introduction

About discrete fourier transform :

The discrete fourier transform converts a finite sequence of equally-spaced samples of a function into a same-length sequence of equally-spaced samples of the discrete-time Fourier transform (DTFT), which is a complex valued function of frequency. The DFT is therefore said to be a frequency domain representation of the original input sequence.

The DFT is the most important discrete transform, used to perform Fourier analysis in many practical applications. In digital signal processing, the function is any quantity or signal that varies over time, such as the pressure of a sound wave, a radio signal, or daily temperature readings, sampled over a finite time interval. In image processing, the samples can be the values of pixels along a row or column of a raster image. The DFT is also used to efficiently solve partial differential equations, and to perform other operations such as convolutions or multiplying large integers.

Since it deals with a finite amount of data, it can be implemented in computers by numerical algorithms or even dedicated hardware. The advantage of DFT over FFT is the restriction of 2^n samples and fixed/discrete sample rates for the FFT.

About power spectrum :

For a given signal, the power spectrum gives a plot of the portion of a signal's power (energy per unit time) falling within given frequency bins. The most common way of generating a power spectrum is by using the discrete fourier transform, and other techniques such as the maximum entropy method can also be used.

About normalized frequency :

Normalised frequency is frequency in Hz (or more generically cycles/second or some other unit) divided by the sample frequency of your signal in Hz (or the same units as your original frequency).

2. Formulae used

Speed-up = T(1) / T(P)

Parallel fraction, f = (1 - T(P)/T(1)) / (1 - (1/P))

where,

T(1) - time taken for serial execution

T(P) - time taken for parallel execution

P - number of processes/threads/processors

Formula for finding DFT of a sample -

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}$$

3. Procedure

The steps for performing the task of finding the normalized frequency of a random wave using discrete fourier transform are -

a. Produce an array of N random numbers within a range, e.g. -1 to 1 using the rand() function. This would be our array of samples.
b. Define a struct which contains real and imaginary double precision numbers, so that it can store the real and imaginary coefficients of the output of DFT. Initialize the real and imaginary values to 0.
c. Find the DFT of all the samples present in the array.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}$$

d. For every element in the array of structures containing the coefficients of the DFT obtained above, find the power spectrum of it. It is the sum of the square of the real coefficient and the square of the imaginary coefficient of the DFT.
e. Find the sum of the product of the frequency and the power spectrum value of all the elements. Here,

   frequency of an element = 2*PI*k/N

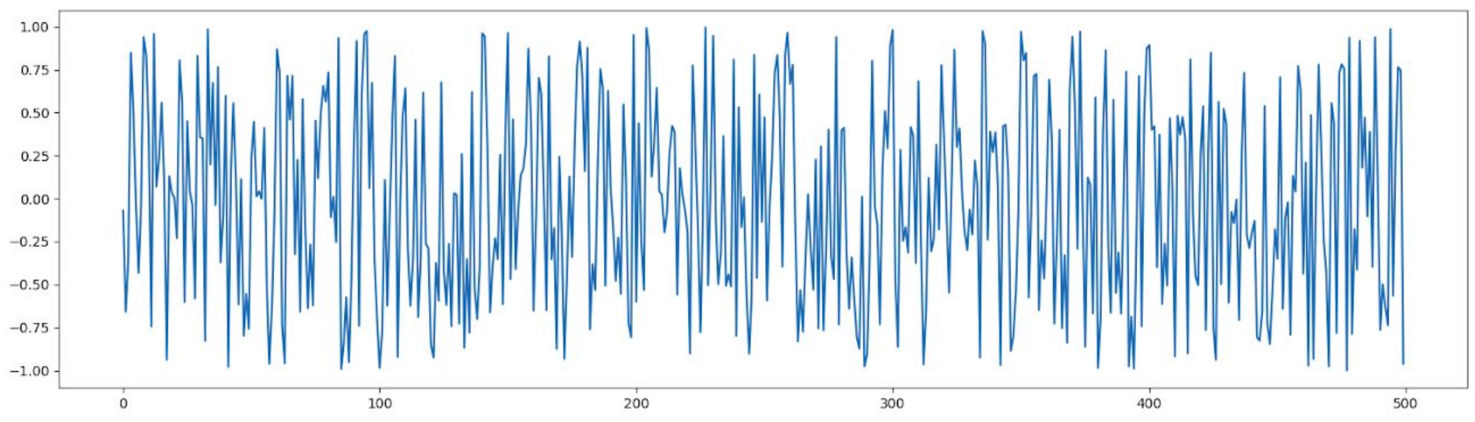
f. Find the sum of the power spectrum values.
g. The normalized frequency of the random wave generated above is the sum of the product of the frequency and the power spectrum value of all the elements (obtained in step e) divided by the sum of the power spectrum values (obtained in step f).

4. Sample inputs and outputs

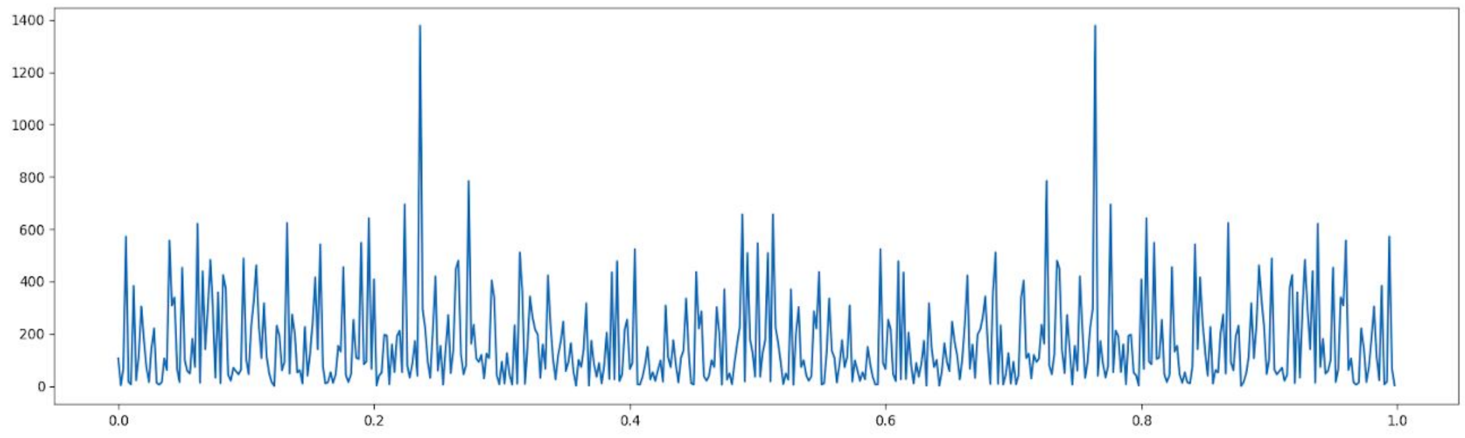
Example 1:

Input :- No. of random samples : 500

Range of the random samples : -1 to 1



Random input signal



Power spectrum vs k/N of the random input signal

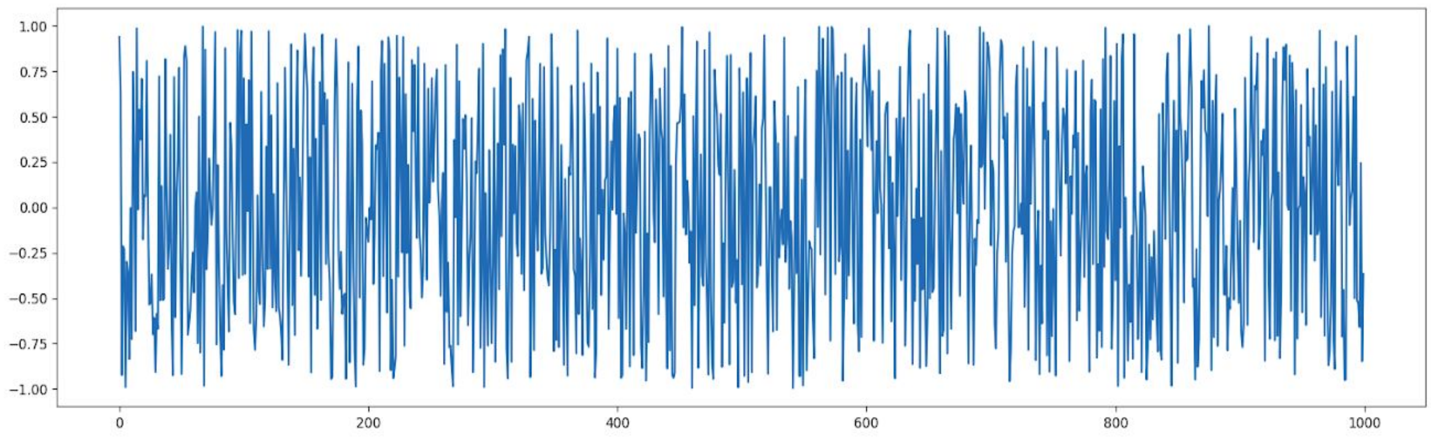
Output :- Normalized frequency of the wave : 3.137617 Hz

Time taken for the program for the given number of processes.

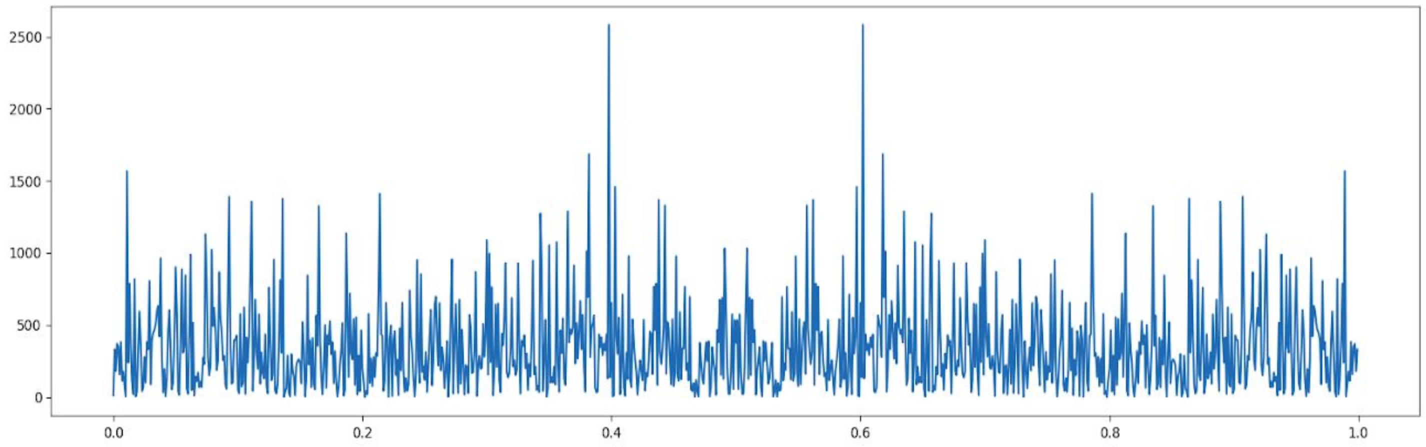
Example 2:

Input :- No. of random samples : 1,000

Range of the random samples : -1 to 1



Random input signal



Power spectrum vs k/N of the random input signal

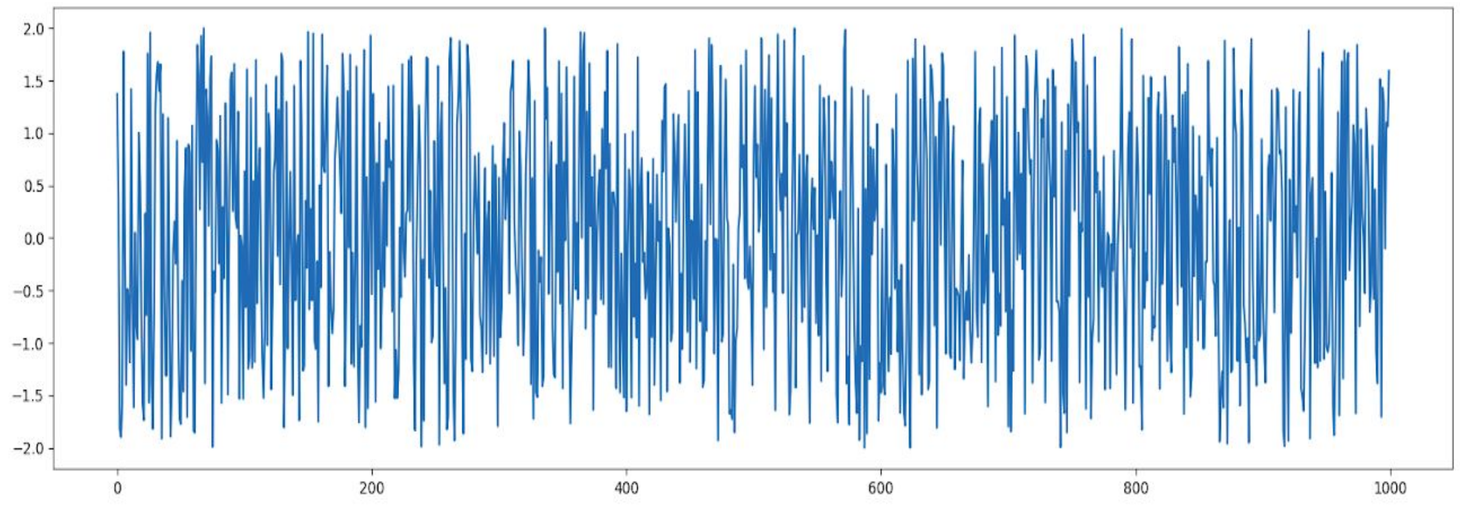
Output :- Normalized frequency of the wave : 3.141485 Hz

Time taken for the program for the given number of processes.

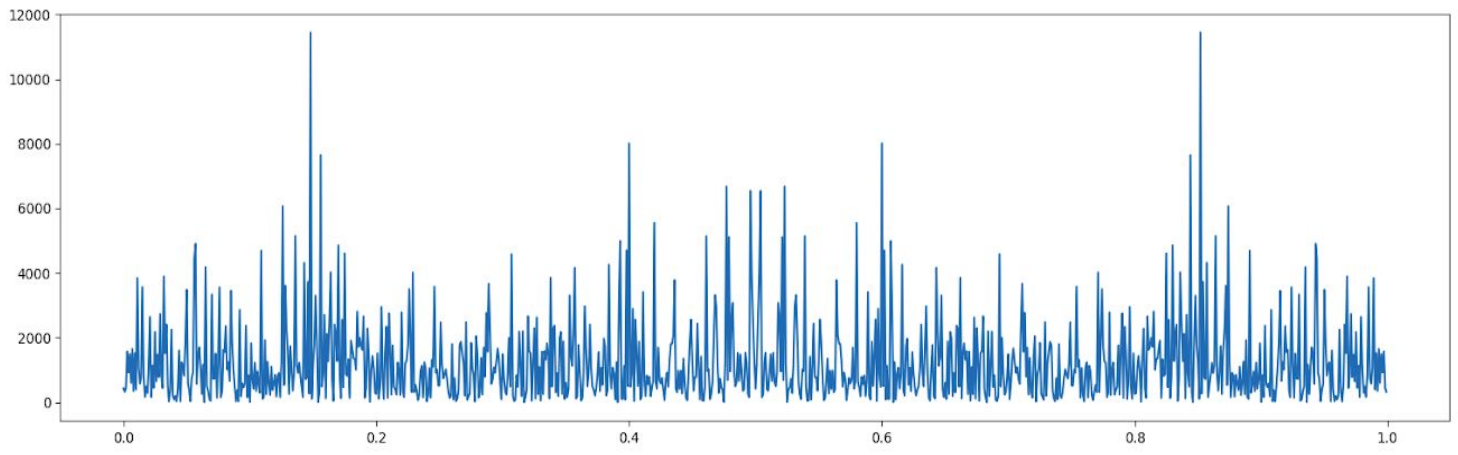
Example 3:

Input :- No. of random samples : 1000

Range of the random samples : -2 to 2



Random input signal



Power spectrum vs k/N of the random input signal

Output :- Normalized frequency of the wave : 3.140520 Hz

Time taken for the program for the given number of processes.

5. Execution time, speed-up and parallel fraction

For the below analysis, the program contains 10,000 sample points in the random wave. The range for these random points, considered here, is -1 to 1.

The number of processors initialized and the corresponding execution time, speed-up and parallel fraction is as shown in the table below.

No. of processes	Execution time	Speed-up	Parallel fraction
1	4.532341	1	#DIV/0!
2	2.270333	1.996333137	0.9981632009
4	1.707884	2.653775666	0.8309045296
6	1.70222	2.662605891	0.749313699
8	**1.697047**	**2.670722143**	**0.7149364975**
10	1.699981	2.666112739	0.6943578752
12	1.709311	2.65156019	0.6794875079
14	1.715441	2.642085038	0.669319589
16	1.743051	2.600234302	0.6564472238
20	1.800295	2.517554623	0.6345148996
24	1.836132	2.468417848	0.6207466469

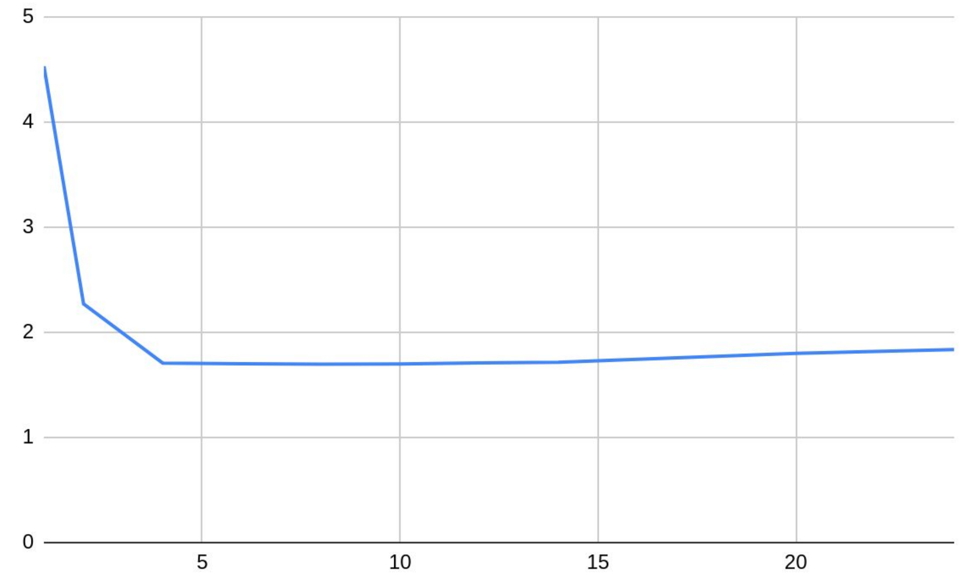
Hence, in this case, the optimal number of processes is 8. In this case-

Execution time = 1.697047

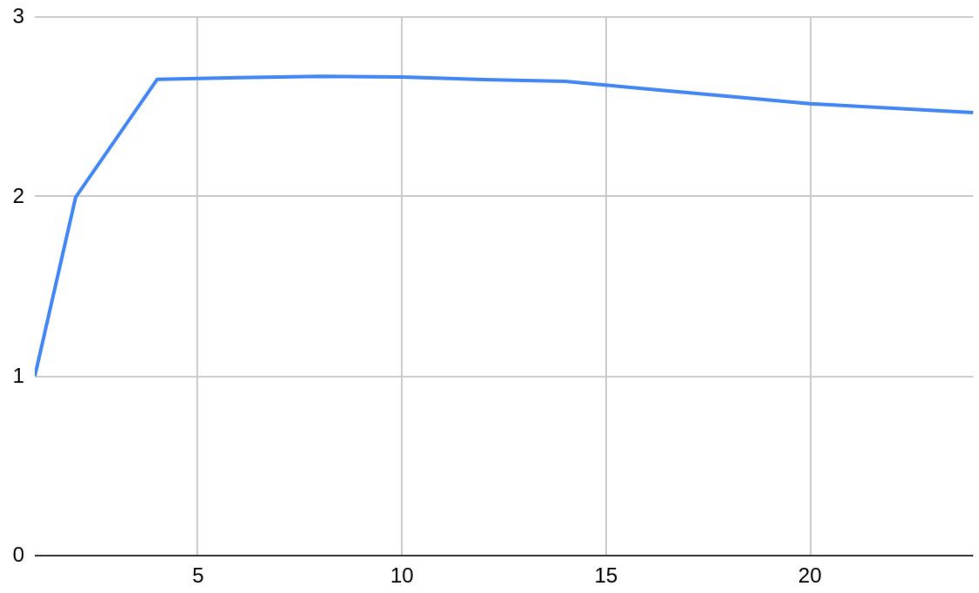
Speed-up = 2.670722143

Parallel fraction = 0.7149364975

The plot below is the graph between the number of processes and execution time.



The plot below is the graph between the number of processes and speed-up.



In this case, the loop iterations are parallelized.