Experiment No.6

**Aim:** To implement computationally fast algorithms.

**Theory:**

A fast Fourier transform (FFT) algorithm computes the discrete Fourier transform (DFT) of a sequence, or its inverse. Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa. An FFT rapidly computes such transformations by factorizing the DFT matrix into a product of sparse (mostly zero) factors. As a result, it manages to reduce the complexity of computing the DFT from O(n2), which arises if one simply applies the definition of DFT, to O(nlog n), where n is the data size.

Fast Fourier transforms are widely used for many applications in engineering, science, and mathematics. The basic ideas were popularized in 1965, but some algorithms had been derived as early as 1805. In 1994, Gilbert Strang described the FFT as "the most important numerical algorithm of our lifetime" and it was included in Top 10 Algorithms of 20th Century by the IEEE journal Computing in Science & Engineering.

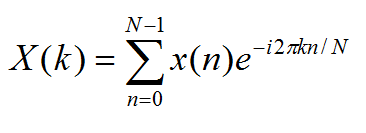
***Overview***

A fast Fourier transform (FFT) algorithm computes the discrete Fourier transform (DFT) of a sequence, or its inverse. Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa. An FFT rapidly computes such transformations by factorizing the DFT matrix into a product of sparse (mostly zero) factors. As a result, it manages to reduce the complexity of computing the DFT from Fast Fourier transforms are widely used for many applications in engineering, science, and mathematics.

***Definition and Speed***

An FFT computes the DFT and produces exactly the same result as evaluating the DFT definition directly; the most important difference is that an FFT is much faster.

Let *x*0, ...., *xN*−1 be complex numbers. The DFT is defined by the formula



**Program:**

#include<stdio.h>

#include<math.h>

#define PI 3.14159265

int main() {

int N = 0;

int n;

float x[10];

int k;

float x\_real[10], x\_img[10];

printf("Enter the length of x(n): ");

scanf("%d", &N);

printf("Enter the values of x(n): ");

for (n = 0; n < N; n++) {

scanf("%f", &x[n]);

}

for (k = 0; k < N; k++)

{ x\_real[k] = 0;

x\_img[k] = 0;

for (n = 0; n < N; n++) {

x\_real[k] = (x[n] \* cos(2 \* PI \* k \* n / (N)) ) + x\_real[k];

x\_img[k] = (x[n] \* sin(2 \* PI \* k \* n / (N)) ) + x\_img[k];

}

}

printf("The Coefficients are:\n");

for (k = 0; k < N; k++) {

printf("x[%d] = ( %f )-( %f i ) \n", k, x\_real[k], x\_img[k]);

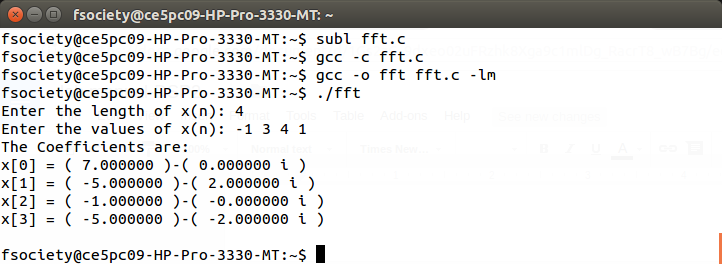
}

printf("\n");

return 0;

}

**Output:**

****

**Conclusion:**

Hence we performed Fast Fourier Transform & implement computationally fast algorithms.