Міністерство освіти і науки України
Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

ЗВІТ

з лабораторної роботи №2.1 з дисципліни
«Інтелектуальні вбудовані системи»

Виконав:
ІП-83
Сергійчук Н. С.

Київ 2020

1. Завдання

Для згенерованого випадкового сигналу з Лабораторної роботи N 1 відповідно до заданого варіантом побудувати його спектр, використовуючи процедурудискретного перетворення Фур'є. Розробити відповідну програму і вивести отримані значення і графіки відповідних параметрів.

2. Лістинг програми

```
use std::fs::File;
use std::io::Write;

use fourier::discrete_fourier;
use signal::gen_signal;
```

```
const HARMONICS: usize = 12;
const FREQUENCY: usize = 900;
const INTERVALS: usize = 256;
const DT: f64 = 1.0;
```

```
const USAGE: &str = "Usage: <signal.dat> <fourier.dat>";
```

```
fn save<T: Write>(mut dest: T, slice: &[f64], header: &str, mult: f64) {
writeln!(dest, "{}", header).unwrap();
for (i, s) in slice.iter().enumerate() {
writeln!(dest, "{}\t{}", i as f64 * mult, s).unwrap();
}
}
```

```
fn main() {
let signal_dest = std::env::args().nth(1).expect(USAGE);
let fourier_dest = std::env::args().nth(2).expect(USAGE);
```

```
let signal_dest = File::create(signal_dest).unwrap();
let fourier_dest = File::create(fourier_dest).unwrap();
```

```
let signal = gen_signal(HARMONICS, FREQUENCY, INTERVALS, DT);
save(signal_dest, &signal, "# t\tx(t)", DT);
```

```
let fourier = discrete_fourier(&signal);
save(fourier_dest, &fourier, "# p\tX(p)", DT);
}
```

```
use num::Complex;
use std::f64::consts::PI;
```

```rust
fn w_coeff(p: usize, k: usize, n: usize) -> Complex<f64> {
let (p, k, n) = (p as f64, k as f64, n as f64);
let x = 2.0 * PI / n * p * k;
Complex::new(x.cos(), 0.0) - Complex::new(0.0, -x.sin())
}
```

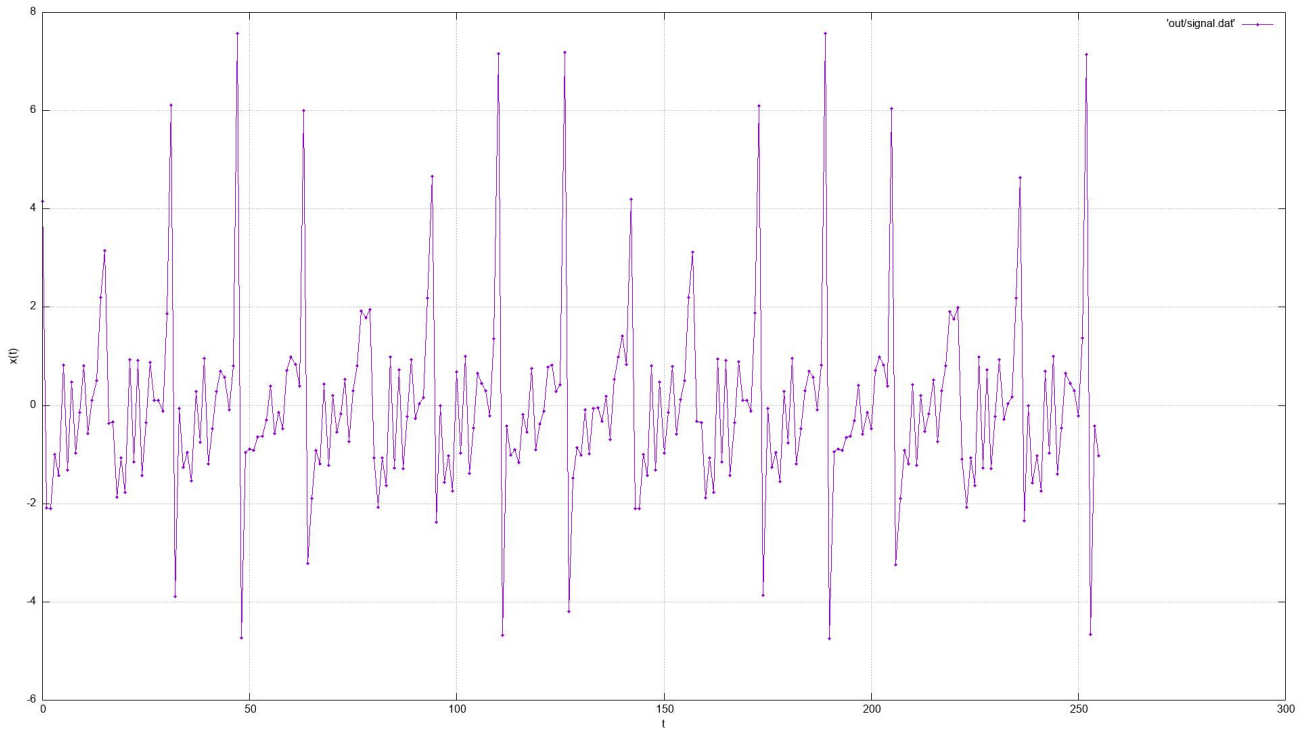
```rust
pub struct Wtable {
table: Vec<Complex<f64>>,
n: usize,
}
```

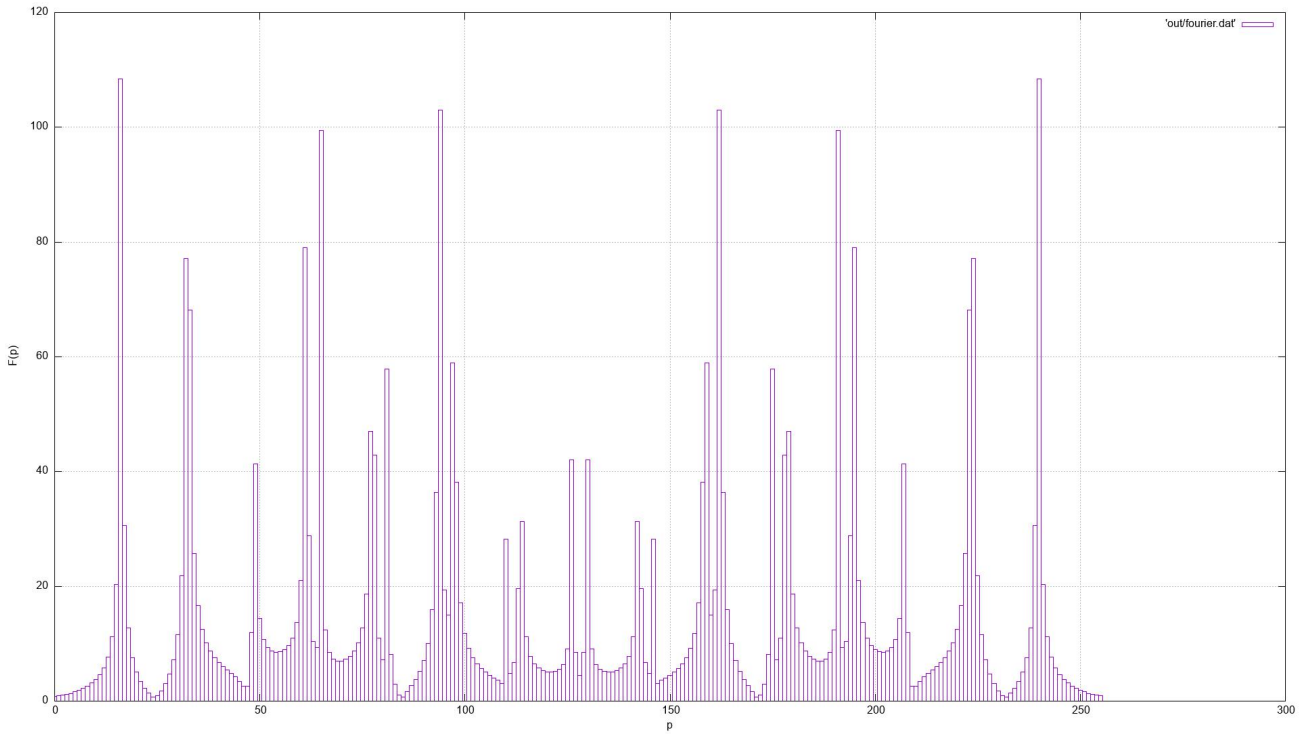
```rust
impl Wtable {
fn new(n: usize) -> Self {
let mut table = Vec::with_capacity(n * n);
for i in 0..n {
for j in 0..n {
table.push(w_coeff(i, j, n));
}
}
Self { table, n }
}
```

```rust
fn get(&self, p: usize, k: usize) -> Complex<f64> {
let index = p * self.n + k;
self.table[index]
}
}
```

```rust
pub fn discrete_fourier(signal: &[f64]) -> Vec<f64> {
let wcoeffs = Wtable::new(signal.len());
(0..signal.len())
.map(|p| {
signal
.iter()
.cloned()
.enumerate()
.map(|(k, x)| x * wcoeffs.get(p, k))
.sum::<Complex<f64>>()
})
.map(Complex::norm)
.collect()
}
```

3. Результати виконання кожної програми.


'out/signal.dat'
x(t)
t
8
6
4
2
0
-2
-4
-6
0
50
100
150
200
250
300


'out/fourier.dat'
F(p)
p
120
100
80
60
40
20
0
0
50
100
150
200
250
300