Національний технічний університет України
«Київський політехнічний інститут»
Факультет інформатики і обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №1
з курсу: «Системи реального часу»

Виконав:
студент групи ІО-71
Муравйов І.П.
Номер у списку: 17
Перевірив:
Регіда П.Г.

Київ 2020 р.

Тема: Дослідження і розробка моделей випадкових сигналів. Аналіз їх характеристик.
Мета: ознайомлення з принципами генерації випадкових сигналів, вивчення та дослідження їх основних параметрів з використанням засобів моделювання і сучасних програмних оболонок.

Теоретичні відомості:

$$M_x = \lim_{N\to\infty} \frac{1}{N} \cdot \sum_{i=1}^{N} x_i(t_k) = \lim_{n\to\infty} \frac{1}{N} \sum_{k=0}^{n} x_i(t_k)$$

$$D_x = \lim_{N\to 0} \cdot \frac{1}{N-1} \sum_{i=1}^{N} x_i(t_k) - M_x)^2 = \lim_{n\to\infty} \frac{1}{n-1} \cdot \sum_{k=0}^{n} (x_i(t_k) - M_x)^2 \geq 0$$

Варіант:

Варіант	Число гармонік в сигналі n	Гранична частота, $\omega_{гр}$	Кількість дискретних відліків, N
18	10	1500	256

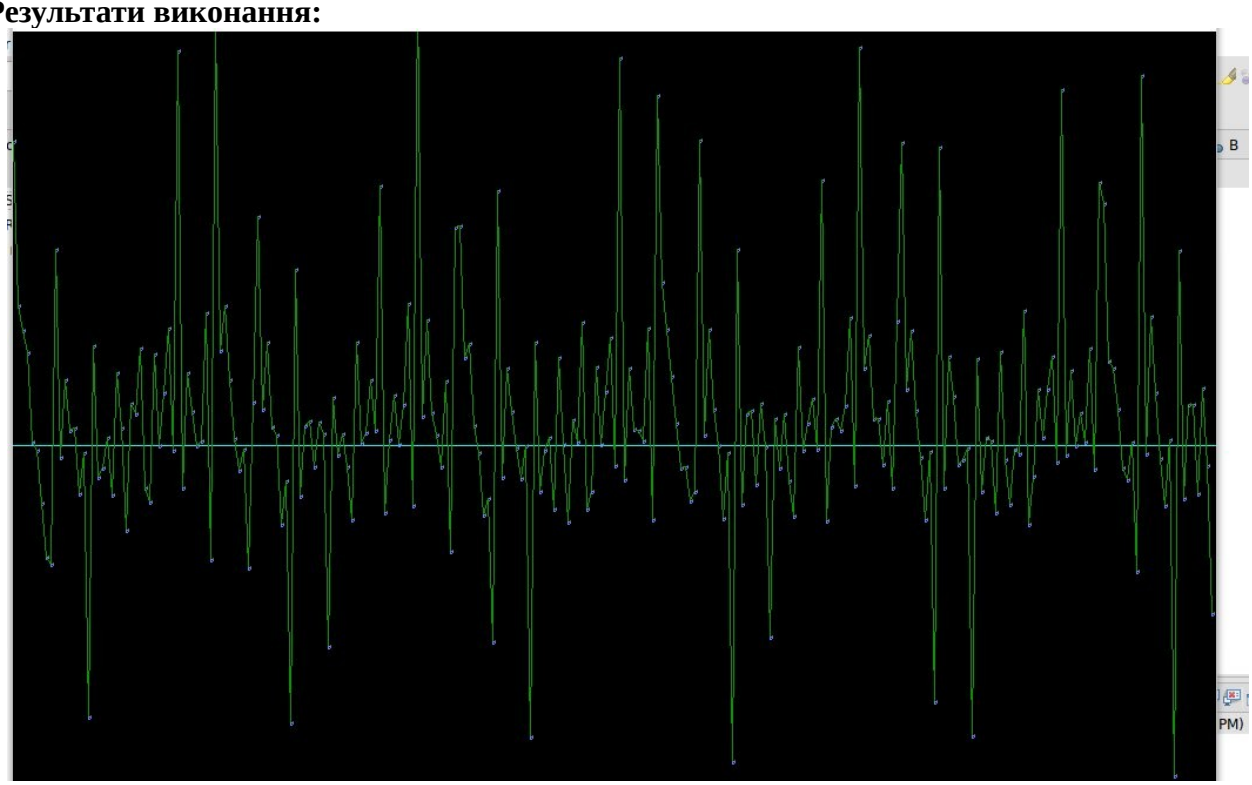
Лістинг програми:

```
#include <cstdio>
#include <cmath>
#include <SDL2/SDL.h>
#include <algorithm>
#include <iostream>
#include <fstream>
#include <ctime>
#include "defs.h"
#include "statistics.hpp"
#include "RNG.h"
#include "app.hpp"
using namespace std;
#define VAR_DISCR 256
#define VAR_W 1500
#define VAR_HARM 10

double x_arr[VAR_DISCR];
int t_arr[VAR_DISCR];

double harmonic(int t, int w, double ampl, double phi){
	return ampl*sin(((double)w)*t+phi);
}
int main(int argc, char **argv) {
	RNG_init();
	App app;
	app.init(1024, 768, 1);
```

```
        SDL_SetRenderDrawColor(app.ren, 0,0,0,0);
        SDL_RenderClear(app.ren);
        SDL_SetRenderDrawColor(app.ren, 0,0,0,0);
        double ampl = RNG.get_float(0, 1);
        double phi = RNG.get_float(0, 1);
        printf("ampl=%lf, phi=%lf\n", ampl, phi);
        // here draw line t=0
        SDL_SetRenderDrawColor(app.ren, 10, 250, 240, 250);
        SDL_RenderDrawLine(app.ren, app.middle_x(), app.middle_y(), app.end_x(),
app.middle_y());
                double x, x_prev;
        for(int t=0; t<VAR_DISCR; t++) {
                x_prev = x;
                x=0;
                for(int harm=0; harm<VAR_HARM; harm++) {
                        x += harmonic(t, harm*VAR_DISCR/VAR_HARM, ampl, phi);
                //SDL_RenderDrawPoint(app.ren, t*3, -x*30+app.height/2);
                }
                x_arr[t] = x;
                t_arr[t] = t;
                printf("x=%lf\n", x);
        }
        // draw x(t)
        std::pair<double*, double*> minmaxx = std::minmax_element(std::begin(x_arr),
std::end(x_arr));
        std::pair<int*, int*> minmaxt = std::minmax_element(std::begin(t_arr), std::end(t_arr));
        // conv
        double x_offs = (abs(*(minmaxx.first))>abs(*(minmaxx.second)))?
abs(*(minmaxx.first)):abs(*(minmaxx.second));
        int t_offs = (abs(*(minmaxt.first))>abs(*(minmaxt.second)))?
abs(*(minmaxt.first)):abs(*(minmaxt.second));
        double x_coef = (app.end_y() - app.middle_y()) / x_offs;
        double t_coef = (app.end_x() - app.middle_x()) / t_offs;
        for(int i=0; i<VAR_DISCR; i++) {
                app.out(t_arr[i]*t_coef, app.real_y(x_arr[i]*x_coef));
                SDL_SetRenderDrawColor(app.ren, 10, 150, 0, 0);
                if(i+1<VAR_DISCR)
                        SDL_RenderDrawLine(app.ren, t_arr[i]*t_coef,
app.real_y(x_arr[i]*x_coef), (t_arr[i+1]*t_coef), app.real_y(x_arr[i+1]*x_coef));
        }
        SDL_UpdateWindowSurface(app.win);
        SDL_RenderPresent(app.ren);
        clock_t start_mx, end_mx, start_dx, end_dx;
        start_mx = clock();
        double Mx = Expected(x_arr, VAR_DISCR);
        end_mx = clock();
        double mx_timeused = ((double) (end_mx - start_mx)) / CLOCKS_PER_SEC;
        start_dx = clock();
        double Dx = Dispersion(x_arr, VAR_DISCR);
        end_dx = clock();
        double dx_timeused = ((double) (end_dx - start_dx)) / CLOCKS_PER_SEC;
        printf("dx time=%f\n\r", dx_timeused);
```

```
        printf("mx time=%f\n\r", mx_timeused);
        // write results to file
        ofstream file;
        file.open("lab_res.txt");
        file << "dx time=" << dx_timeused << endl;
        file << "mx time=" << mx_timeused << endl;
        file << "Mx=" << Mx << endl;
        file << "Dx=" << Dx << endl;
        file.close();
        // get it into google drive
        extern int gdrive_out();
        gdrive_out();
        while (1){
                SDL_Event event;
                SDL_PollEvent(&event);
                if(event.type == SDL_QUIT || event.key.keysym.sym == 'q') {
                        SDL_DestroyRenderer(app.ren);
                        SDL_DestroyWindow(app.win);
                        SDL_Quit();
                        exit(0);
                }
        }
        return 0;
}
```

Результати виконання:



Висновок: під час виконання даної лабораторної роботи була написана програма, яка генерує випадковий сигнал та рахує математичне очікування та дисперсію. Математичне очікування, як і очікувалось, завжди близько нуля, але дисперсія приймає різні значення в залежності від границь значення A та випадкових значень A та фі, через що не може бути приблизно спрогнозоване.