МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И. ВЕРНАДСКОГО»

ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра компьютерной инженерии и моделирования

Спектральный анализ периодических сигналов

Отчет по лабораторной работе №2

по дисциплине «**Обработка сигналов**»

студента 3 курса группы ИВТ-б-о-222(1)

Гоголева Виктора Григорьевича

Направления подготовки 09.03.01«Информатика и вычислительная техника»

Симферополь, 2025

Лабораторная работа №2

Тема: Спектральный анализ периодических сигналов

Цель работы: разработать программное обеспечение, осуществляющее спектральный анализ периодической последовательности импульсов. Получить аналитические выражения для коэффициентов разложения. Найти амплитуду и фазу гармоник и построить амплитудные и фазовые спектральные диаграммы. Провести цикл вычислительных экспериментов, в котором определить количество гармоник исходя из потери относительной мощности сигнала (10 %, 5 %, 2 %, 1 %, 0,1 %). Графически изобразить исходный и аппроксимированный периодические сигналы для различного количества гармоник при разложении в спектр.

Теоретические сведения

Спектральный анализ периодических сигналов – это процесс анализа, который разбивает периодический сигнал на сумму частотных компонентов. Частотные компоненты – отдельное изменение амплитуды сигнала на заданной частоте.

Периодическая последовательность импульсов может быть записана в виде ряда Фурье для периодического сигнала:

$$s(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cdot \cos(n \cdot \omega_1 \cdot t) + b_n \cdot \sin(n \cdot \omega_1 \cdot t)$$

где $\omega 1 = 2 \cdot \pi \cdot f$ – основная частота последовательности; a_0, a_n, b_n – коэффициенты разложения в ряд Фурье.

Коэффициенты разложения рассчитываются по следующим формулам:

$$a_0 = \frac{2}{T} \cdot \int_{-T/2}^{T/2} s(t) \cdot dt;\ a_n = \frac{2}{T} \cdot \int_{-T/2}^{T/2} s(t) \cdot \cos(n \cdot \omega_1 \cdot t) dt;\ b_n = \frac{2}{T} \cdot \int_{-T/2}^{T/2} s(t) \cdot \sin(n \cdot \omega_1 \cdot t) dt$$

где T=1/f – период последовательности импульсов.

Эквивалентная форма ряда Фурье:

$$s(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} A_n \cdot \cos(n \cdot \omega_1 \cdot t - \varphi_n)$$

где A_n – амплитуда, φ_n – начальная фаза n-ой гармоники. Амплитуда и начальная фаза гармоник определяются через коэффициенты ряда Фурье:

$$A_0 = \sqrt{a_n^2 + b_n^2};\ tg\varphi_n = \frac{b_n}{a_n}$$

Средняя мощность для периодической последовательности импульсов определяется следующим выражением:

$$P_c = \frac{1}{T} \cdot \int_{-T/2}^{T/2} s^2(t) \cdot dt$$

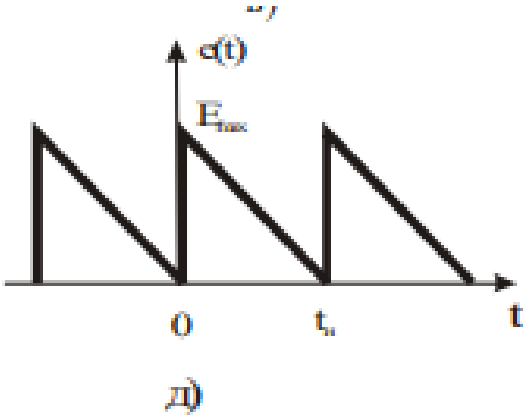
Мощность, заключенная в сложном периодическом сигнале, может быть рассчитана через коэффициенты ряда Фурье:

$$P_k = \left(a_0 / 2 \right)^2 + \frac{1}{2} \cdot \sum_{n=1}^{\infty} (A_n)^2$$

Ход работы

E = 280

t = 4




```
public delegate double fooHandler(double x);
public fooHandler current_foo ;

Ссылок: 6
public static double tri(double x, double t_imp, double e_max)
{
    return (2 * e_max / Math.PI) * Math.Atan(1/Math.Tan(x*Math.PI/t_imp));
}
Ссылок: 4
```

Рисунок 1 - Функция исходного сигнала

```
ссылка: 1
public double fourierSeries(double x,double t_imp, List<double> array_a, List<double> array_b)
{
    double sum = 0;
    for (int k = 1; k <= array_a.Count; k++)
    { sum += array_a[k-1] * Math.Cos(k * x * 2d*Math.PI / t_imp) + array_b[k-1] * Math.Sin(k * 2d  * x * Math.PI / t_imp); }
    return sum;
}
ссылка: 1
```

Рисунок 2 – функция для построения ряда Фурье

Вычисление частотных компонент

Вычисление частотных компонент сигнала выполняется при помощи преобразования Фурье. Путём интегрирования временной функции сигнала на бесконечном интервале времени, при различных значениях частоты. Таким

образом получается комплексное число, которые представляет амплитуду и фазу частотной компоненты.

```
ссылка: 1
public void createChartAmplitude(List<double> array_a, List<double> array_b)
{
    chart_A.Series[0].Points.Clear();
    for (int i = 0; i < array_a.Count; i++)
    {
        chart_A.Series[0].Points.AddXY(i+1, Math.Sqrt(Math.Pow(array_a[i], 2) + Math.Pow(array_b[i], 2)));
    }
}
ссылка: 1
public void createChartPhase(List<double> array_a, List<double> array_b)
{
    chart_F.Series[0].Points.Clear();
    for (int i = 0; i < array_a.Count; i++)
    {
        chart_F.Series[0].Points.AddXY(i+1, Math.Abs(Math.Atan(array_b[i] / array_a[i])));
    }
}
```

Рисунок 3 – функции вычисления частотных компонент

Коэффициент разложения – вычисляется с помощью быстрого преобразования Фурье. Он показывает, насколько каждая частотная компонента сигнала вносит свой вклад в общую сумму сигнала.

```
double Pc, Pk;
List<double> array_a = new List<double>();
List< double > array_b = new List<double>();
double a0 = 0;
math.current_foo = (x) => { return MathFunctions.tri(x,t_imp,e_max); };
a0 = (2d / t_imp) * math.integration(0, t_imp,10000);
```

Рисунок 4 – расчет коэффициента разложения

Восстановление сигнала обратным преобразованием

Для восстановления сигнала из его частотных компонент необходимо выполнить обратное преобразование Фурье, которое обратно преобразует спектральное представление сигнала во временную функцию. К каждой частотной компоненте сигнала применяется обратное преобразование Фурье и результаты складываться.

```
ссылка: 1
public void createChartFinale(double a, double b, fooHandler foo)
{
    chart_finale.Series[0].Points.Clear();
    double h = (b - a) / tochek;
    for (int i = 0; i < tochek; i++)
    {
        chart_finale.Series[0].Points.AddXY(a + i * h, foo(a + i * h));
    }
}
Ссылок: 2
public void createChart(double a, double b,fooHandler foo)
{
    chart.Series[0].Points.Clear();
    double h = (b - a) / tochek;
    for (int i = 0; i < tochek; i++)
    {
        chart.Series[0].Points.AddXY(a + i * h, foo(a + i * h));
    }
}
```

Рисунок 4 – функции востонавления сигнала обратным преобразованием

Расчёт мощности

```
double Pc, Pk;
List<double> array_a = new List<double>();
List< double > array_b = new List<double>();
double a0 = 0;
math.current_foo = (x) => { return MathFunctions.tri(x,t_imp,e_max); };
a0 = (2d / t_imp) * math.integration(0, t_imp,10000);
math.current_foo = (x) => { return Math.Pow(MathFunctions.tri(x, t_imp, e_max), 2); };
Pc = math.integration(0,t_imp,10000)/t_imp;
Pk = Math.Pow(a0 / 2, 2);
while ((Pc-Pk)/Pc >loses)
{
    //if (N % 2 == 1)
    //    array_a.Add(4 * e_max / Math.Pow(Math.PI * N, 2));
    //else
    //    array_a.Add(0);
    //array_b.Add(0);
    math.current_foo = (double x) =>
    {
        return MathFunctions.tri(x,t_imp,e_max)*Math.Cos(x*(double)N*2d*Math.PI/12d);
    };
    array_a.Add((2d / t_imp) * math.integration(-t_imp, t_imp, 10000));

    math.current_foo = (double x) =>
    {
        return (MathFunctions.tri(x, t_imp, e_max) * Math.Sin(2d*(double)N * x * ((2d * Math.PI) / t_imp)));
    };
    array_b.Add((2d / t_imp) * math.integration(-t_imp, t_imp, 10000));
    Pk += 0.5 * (Math.Pow(array_a[array_a.Count-1], 2)+ Math.Pow(array_b[array_b.Count-1],2));
    N++;
}
labelPc.Text = "Pc=" + Math.Round(Pc,5).ToString();
labelPk.Text = "Pk=" + Math.Round(Pk,5).ToString();
labelloses.Text = "(Pc-Pk)/Pc=\n" + ((Pc - Pk) / Pc).ToString();
```

Функция интегрирования

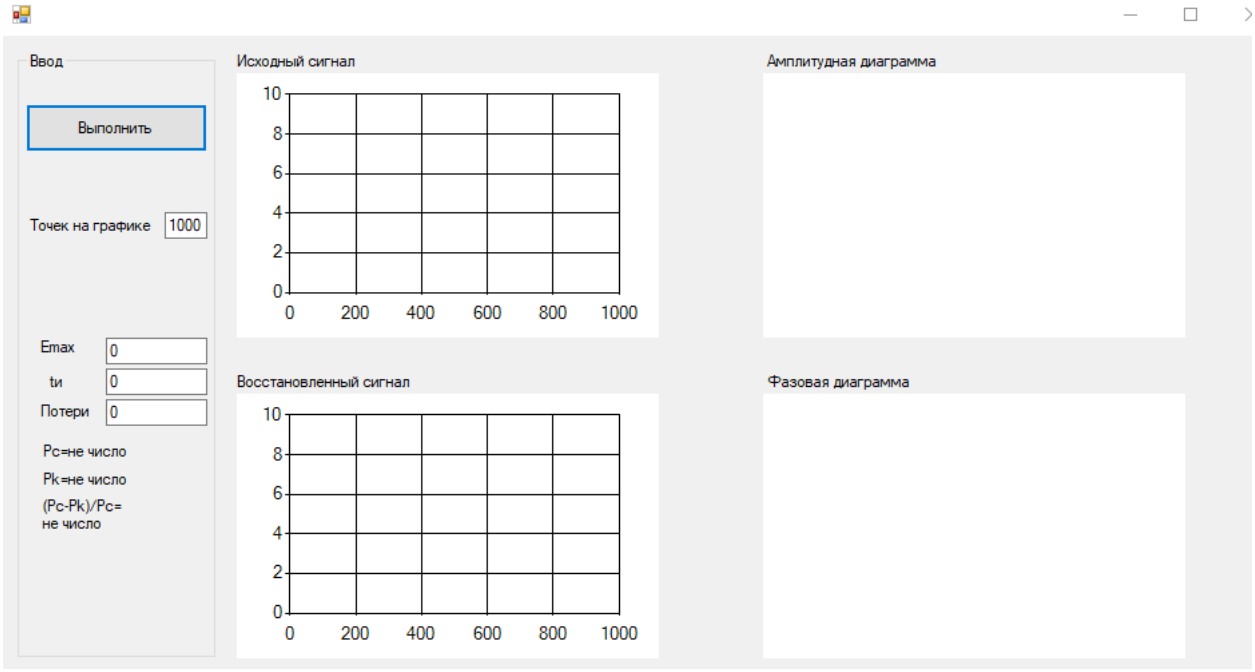
```
Ссылок: 8
public class MathFunctions
{
    public delegate double fooHandler(double x);
    public fooHandler current_foo ;

    Ссылок: 6
    public static double tri(double x, double t_imp, double e_max)//напильник
    {
        return (2 * e_max / Math.PI) * Math.Atan(1/Math.Tan(x*Math.PI/t_imp));
    }
    Ссылок: 4
    public double integration(double a,double b ,int n)
    {
        double h = (b - a) / n;
        double sum1 = 0d;
        double sum2 = 0d;
        for (int k = 1; k <= n; k++)
        {
            double xk = a + k * h;
            if (k <= n - 1)
            {
                sum1 += current_foo(xk);
            }

            double xk_1 = a + (k - 1) * h;
            sum2 += current_foo((xk + xk_1) / 2);
        }

        double result = h / 3d * (1d / 2d * current_foo(a) + sum1 + 2 * sum2 + 1d / 2d * current_foo(b));
        return result;
    }
}
```

Интерфейс



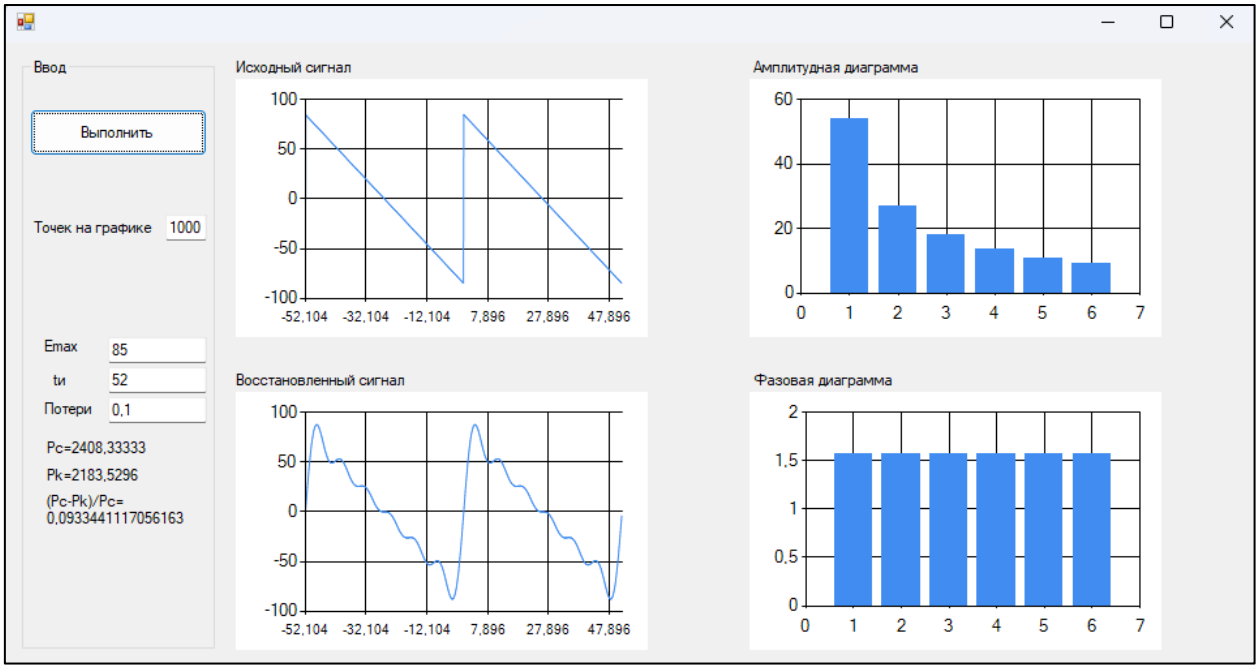


Рисунок – статистика и графики сигналов при потерях 10%

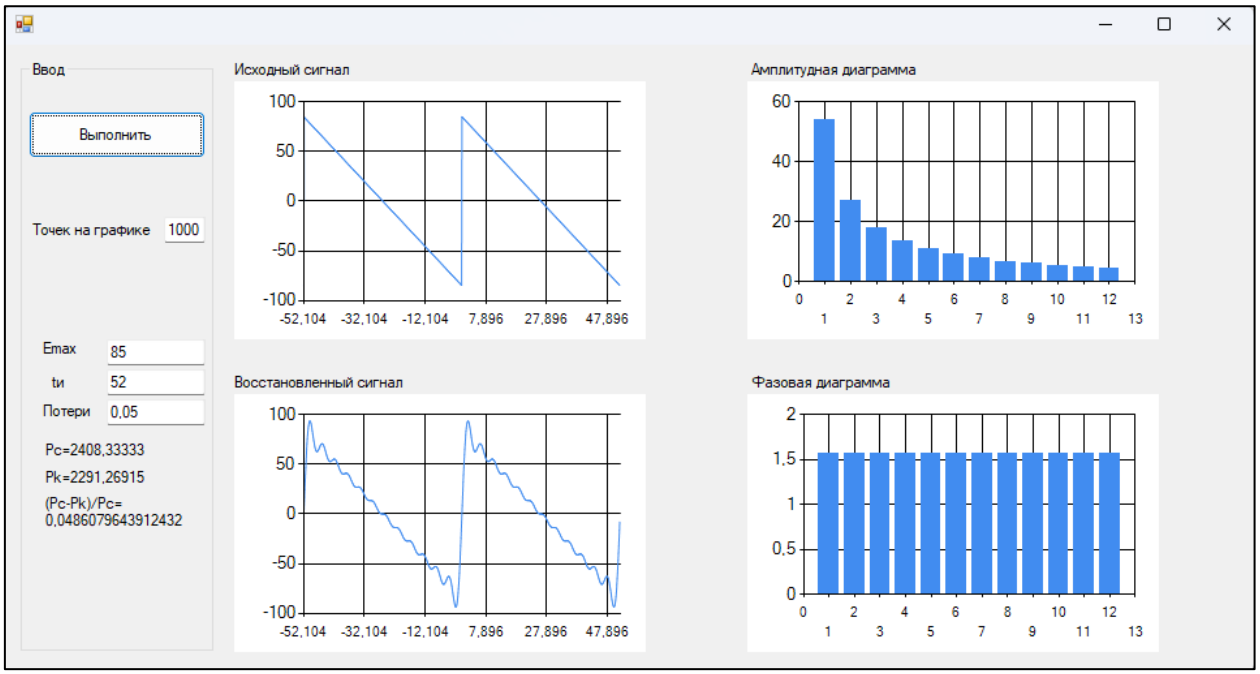


Рисунок – статистика и графики сигналов при потерях 5%

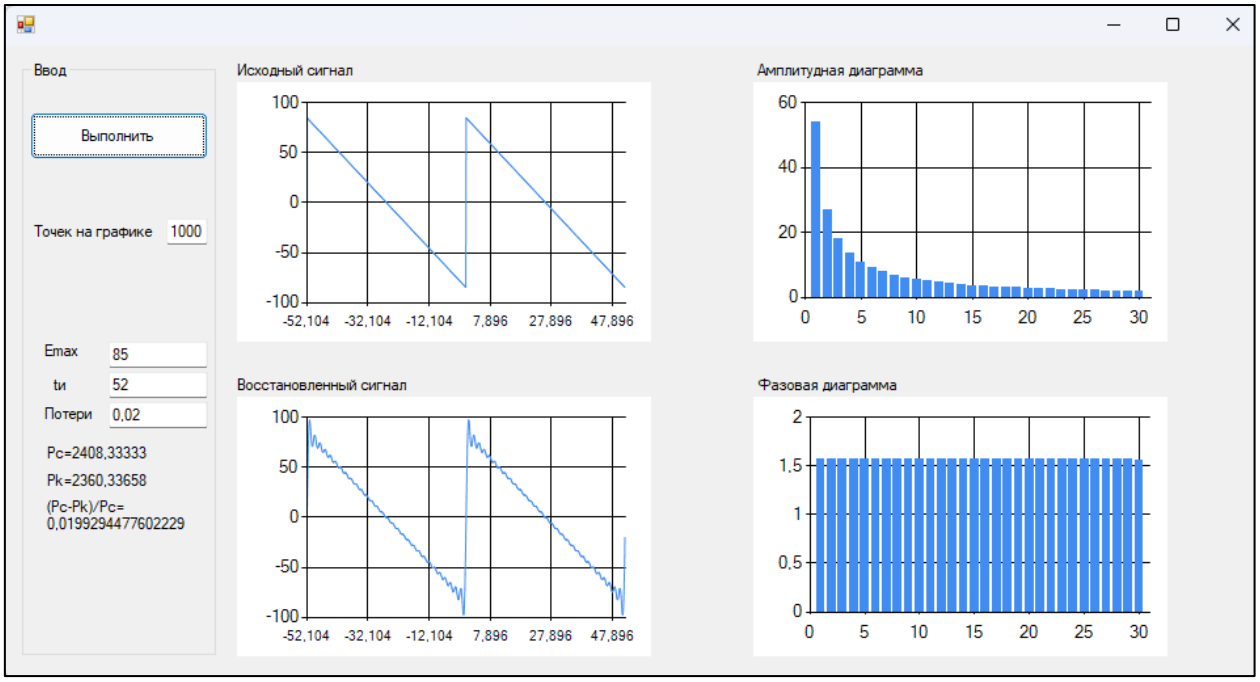


Рисунок – статистика и графики сигналов при потерях 2%

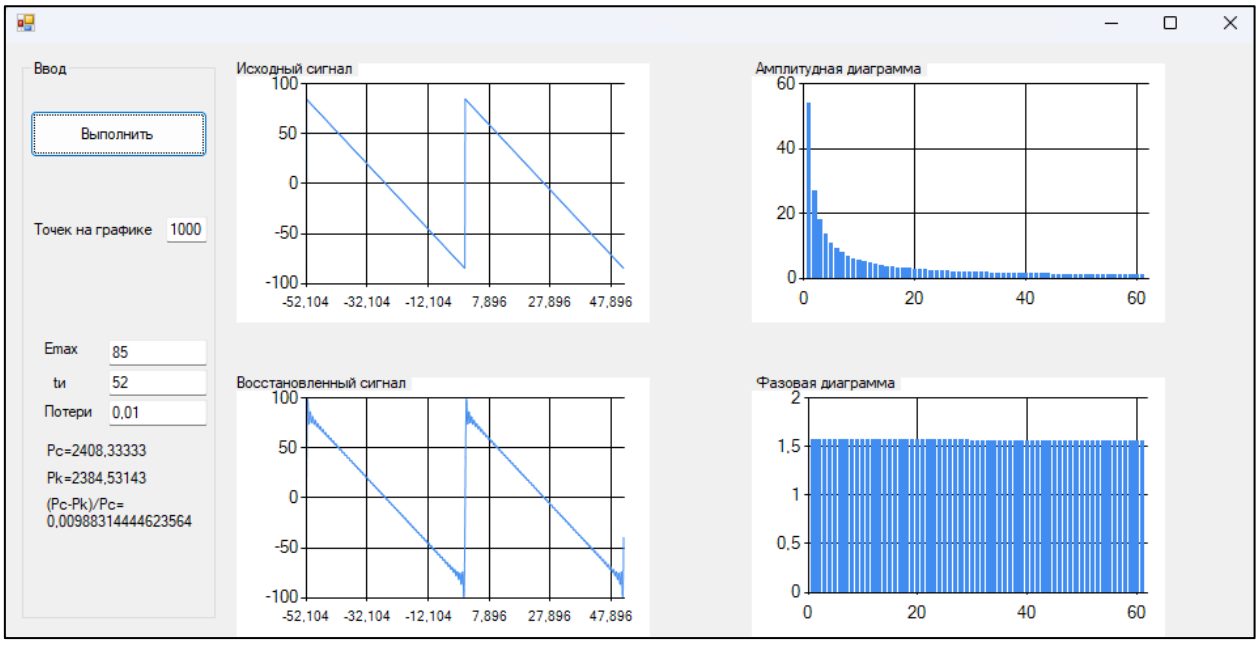


Рисунок – статистика и графики сигналов при потерях 1%

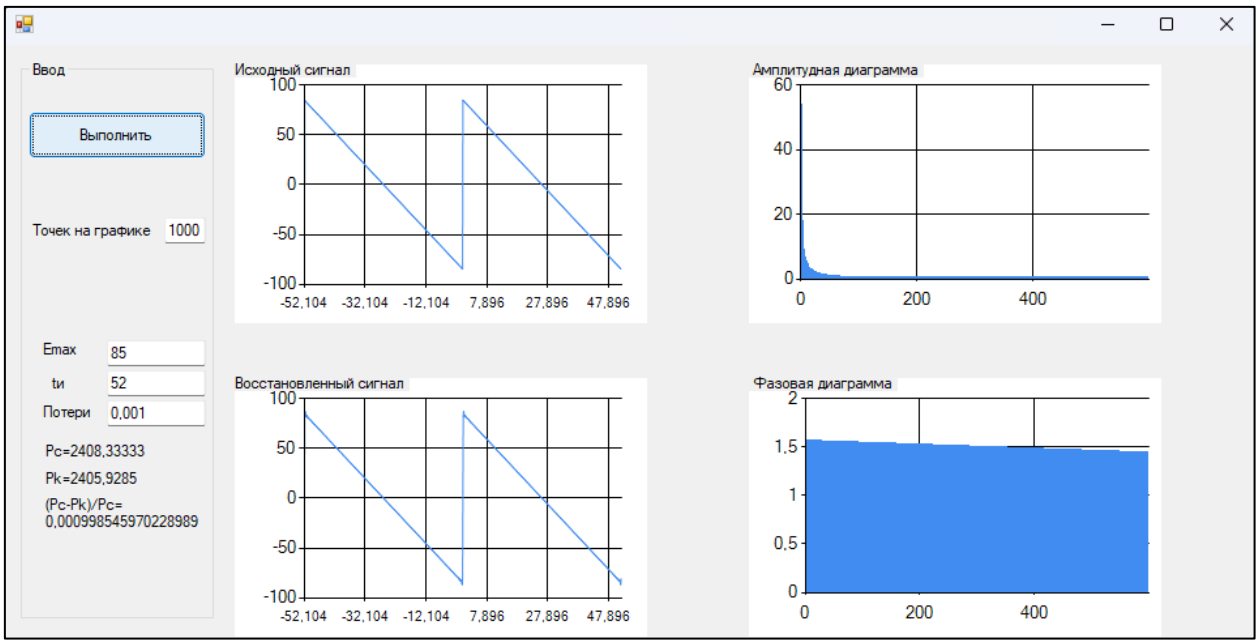


Рисунок – статистика и графики сигналов при потерях 0.1%

ВЫВОД

В ходе работы разработано программное обеспечение, осуществляющее спектральный анализ периодической последовательности импульсов. Рассчитаны аналитические выражения для коэффициентов разложения. Найдена амплитуда и фаза гармоник. Построены амплитудные и фазовые спектральные диаграммы. Проведен цикл вычислительных экспериментов, в котором определено количество гармоник исходя из потери относительной мощности сигнала (10 %, 5 %, 2 %, 1 %, 0,1 %). Графически изображен исходный и аппроксимированный периодические сигналы для различного количества гармоник при разложении в спектр.