Содержание

Введение

После появления в 1982 году первого сканирующего туннельного микроскопа (СТМ) Биннига и Рорера туннельная микроскопия получила бурное развитие.

Интерес к СТМ объясняется, в первую очередь, его уникальным пространственным разрешением: до нескольких сотых ангстрема по нормали к поверхности исследуемого образца и единицы ангстрем – вдоль неё. При этом в принципе для работы микроскопа вовсе не требуется высокий вакуум, как для электронных микроскопов других типов: он может работать на воздухе и даже в жидкой среде.

СТМ позволяет получать весьма богатую информацию: сведения о микрорельефе поверхности, локальной работе выхода, спектре электронных состояний с атомным пространственным разрешением, составе поверхностного слоя, распределении поверхностного слоя, распределении потенциалов при протекании тока через образец. СТМ может работать в широком температурном диапазоне: от гелиевых температур до сотен градусов по Цельсию.

1. Исходное задание

Выполнить исследование работы отрицательной обратной связи сканирующего туннельного микроскопа и построить СТМ-профилограммы для следующей поверхности (рисунок 1) при различных параметрах туннельного промежутка и острия иглы (туннельный зазор $Z0 = 5Å, 7Å, 10Å, 15Å$; туннельное напряжение $U_T = 0.01$ В, 0.1В;локальная работа электронов $\varphi_0 = 4.5$ эВ; уровень Ферми $E_f = 5.71$ эВ).

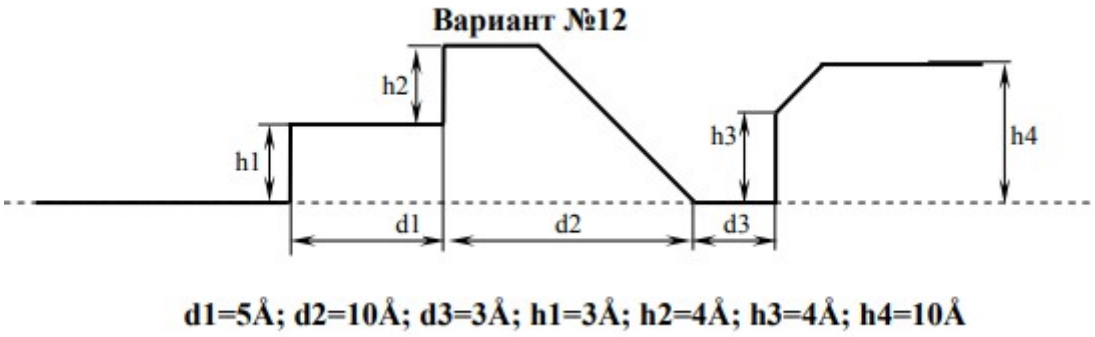

Рисунок 1 - Поверхность

2. Краткие теоретические сведения

2.1 Принцип работы СТМ

В основе работы микроскопа лежит явление квантово механического туннелирования электрона сквозь барьер, высота которого превышает энергию электрона. Если приблизить два проводника (электрода), разделённые диэлектриком (вакуум, газ или жидкость) на достаточно малое расстояние ($\sim 4 - 20$Å) и создать между ними смещающее напряжение U_T, то в цепи потечёт туннельный ток I_T, пропорциональный приложенному напряжению и экспоненциально зависящий от расстояния между электродами.

Для реализации туннельного микроскопа проводящий образец используется в качестве основного электрода по отношению к металлическому острию, служащему сканирующим электродом. Сканирующий электрод установлен над основным на расстоянии нескольких ангстрем, представляющим собой туннельный барьер. Металлическое остриё служит в качестве источника электронов или, так называемого, холодного катода. Туннельный эффект применяется только для высвобождения электронов из металла контакта в диэлектрик. Под влиянием сильного электрического поля электроны высвобождаются из эмиттирующего острия и ускоряются для формирования изображения. Каждый из электродов может двигаться по трём осям X, Y, Zс помощью трёх пьезопреобразователей (ПП), обеспечивающих грубое перемещение образца и точное перемещение острия. Электронный блок управления подсоединен к электродам (т.е. к образцу и к острию) и к ПП, обеспечивающим малые перемещения электродов, которые также должны быть определёнными и воспроизводимыми. Точное положение острия в трёх координатах становится известным на основании значений напряжений, обеспечивающих соответствующий шаг ПП.

Различные конструкции ПП позволяют перемещать остриё зонда на расстояния от долей ангстрема до нескольких десятков микрометров.

Структурная схема СТМ представлена на рисунке 2, где 1 – образец; 2 – примесные атомы; И – игла; $\text{ПП}_{X,Y,Z}$ – пьезопреобразователь; СОС – система обратной связи; ЦАП и АЦП – цифро-аналоговые и аналого-цифровые преобразователи; $\text{ВУ}_{X,Y,Z}$ – высоковольтные усилители.

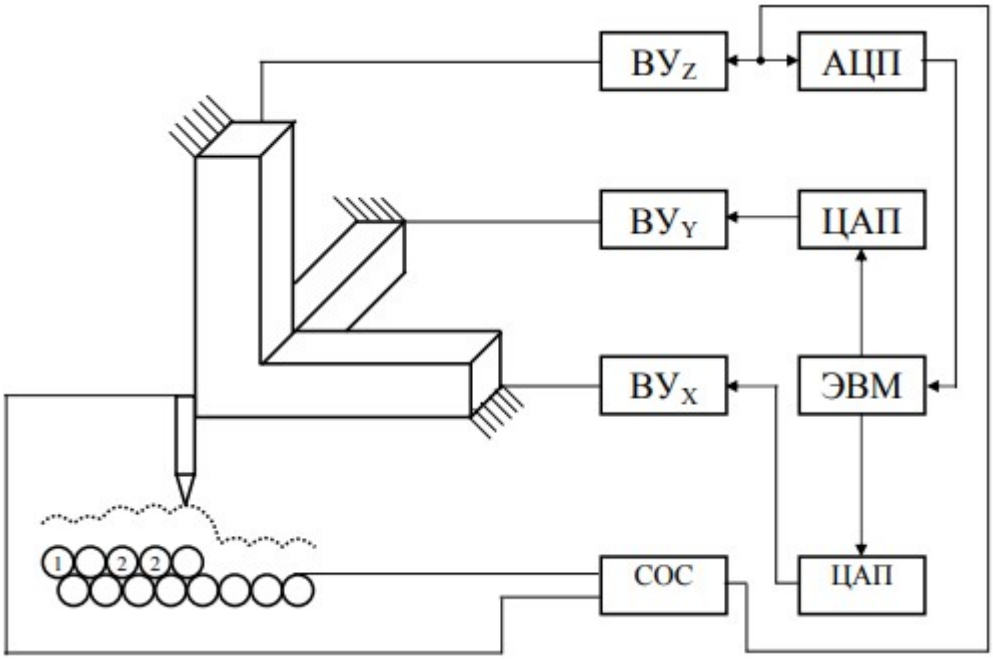

Рисунок 2 – Структурная схема СТМ

Туннельный ток измеряется и поддерживается постоянным с помощью точной настройки ПП по оси Z посредством системы отрицательной обратной связи. Строчная развёртка осуществляется пьезоприводом по оси X, кадровая – по оси Y. Поскольку положение пьезопривода пропорционально пьезонапряжению, напряжения трёх пьезоэлектрических двигателей определяют положение острия на каждой оси. В целом координаты являются декартовыми с тремя ортогональными осями. Анализ информации осуществляется в виде анализа трёхмерного изображения поверхности образца.

Система обратной связи функционирует таким образом, чтобы при постоянном туннельном напряжении поддерживать действительный

(измеренный) туннельный ток равным заданному. Исследованиями установлено, что при поддерживании системой СОС заданного значения туннельного тока с точностью не хуже 2% величина туннельного зазора остаётся также постоянной с ошибкой, не превышающей $0{,}1Å$. Если производить сканирование остриём поверхности образца, подавая на пьезоприводыX, Y развёртывающие напряжения подобно телевизионной развёртке, то система СОС будет стремиться поддерживать заданным туннельный ток, а значит и постоянный зазор игла-остриё, изменяя напряжение на -пьезоприводе в соответствии с рельефом образца.

Таким образом, напряжение на -пьезоприводе острия будет линейно отражать текущую высоту рельефа поверхности образца. Для управления процессом сканирования, задания необходимых туннельных тока и напряжения, запоминания и хранения измерительной информации, а также её преобразования, обработки и отображения в требуемом виде в составе СТМ необходима ЭВМ с цифро-аналоговыми и аналого-цифровыми преобразователями и устройствами отображения.

Первичное СТМ-изображение обычно формируется в виде карты распределения измеряемого параметра (высоты рельефа, работы выхода электрона и т.д.) по плоскости сканирования. Строго говоря, регистрируется не рельеф, а электронное состояние поверхности, но в большинстве случаев на однородных по составу образцах эти зависимости подобны. Для корректного использования таких изображений требуется тщательная калибровка пьезоприводов на объектах с известной топографией.

2.2 Метод статистических испытаний (Монте-Карло) для вычисления интегралов

В СТМ реализуется косвенный метод измерений. В результате косвенных измерений измеряются значения физических величин, функционально связанных с другими, являющимися конечной целью измерения. Для численных исследований параметров СТМ вместо физических экспериментальных данных можно воспользоваться экспериментальными данными, полученными в опытах над моделью объекта, т.е. с помощью вычислительного эксперимента.

Задачи, связанные с вычислением многократных интегралов для исследования параметров СТМ, могут быть эффективно решены методом статистических испытаний, так как в этом случае к точности результатов расчёта не предъявляются очень жёсткие требования. Рассмотрим простейший пример для случая одного измерения.

Пусть ξ – непрерывная случайная величина, принимающая свои значения x_i в некоторой области Ω на оси OX. Закон распределения задан плотностьювероятностей $f(x)$ в Ω. Рассмотрим задачу об определении вероятности попадания случайной величины ξ в интервал ω с фиксированными границами a и b, содержащийся в Ω. Если обозначить искомую вероятность $P(a \leq \xi < b) = p$, то она выражается в виде интеграла:

$$p = \int_a^b f(x)dx. \qquad (1)$$

Этот интеграл можно вычислить по методу статистических испытаний (эксперимент с получением случайных значений случайной величины ξ). Если появившееся при данном испытании значение x_i находится внутри интервала ω, данное испытание будем считать удачным. После проведения N испытаний подсчитаем число m удачных испытаний и вычислим частоту p попадания случайной величины ξ в интервал ω:

$$\bar{p} = \frac{m}{N}. \quad (2)$$

Располагая частотой$\bar{p}$, мы можем приближённо оценить искомую вероятность p на основании закона больших чисел.

Для этого воспользуемся теоремой Бернулли: если событие A имеет вероятность p и если m – число наступления событий A при N независимых испытаниях, то каково бы ни было постоянное $\varepsilon > 0$

$$\lim_{N\to\infty} P\left(\left|\frac{m}{N} - P\right| < \varepsilon\right) = 1. \quad (3)$$

При достаточно большом числе испытаний в качестве оценки для интеграла можно взять частоту $\frac{m}{N}$,т.е.

$$p \approx \bar{p}. \quad (4)$$

Моделирование эксперимента включает:

— из совокупности случайных чисел с законом распределения $f(x)$ извлекается число x_i;

— случайное число x_i сравнивается с границами a и b интервала ω. Результаты сравнения отмечаются специальным признаком β, равным единице, если выполнено неравенство

$$a \le x_i < b; \quad (5)$$

и равным нулю в противном случае;

— полученная величина β прибавляется к содержимому «счётчику числа удачных испытаний»;

— к содержимому «счётчика количества испытаний» прибавляется единица.

После проведения N испытаний определяется приближённое значение искомой вероятности

$$p \approx \frac{m}{N}. \quad (6)$$

Описанная процедура не требует запоминания всех случайных чисел, извлекаемых в процессе счёта. По ходу вычисления запоминаются только число испытаний N и число удачных испытаний m.

Для того, чтобы метод статистических испытаний можно было считать практически приемлемым, необходимо оценить точность равенства (4) и на этом основании определить число испытаний N для вычисления интеграла с достаточной точностью. Представление о точности можно получить, рассматривая $\bar{p}$, как случайную величину. Она имеет математическое ожидание

$$M(\bar{p}) = p. \tag{7}$$

и дисперсию

$$D(\bar{p}) = \frac{p(1-p)}{N}. \tag{8}$$

Поэтому средняя квадратичная ошибка σp равенства (21) будет равна

$$\sigma_{\bar{p}} = \sqrt{\frac{p(1-p)}{N}}. \tag{9}$$

Легко видеть, что максимум $\sigma_{\bar{p}}$ достигается при $p = 0.5$.

2.3 Исследование системы обратной связи СТМ и построение профилограмм

Иглу нужно поддерживать на такой высоте, чтобы туннельный ток оставался постоянным. Для этого нужно находить такую новую высоту, чтобы выполнялось равенство:

$$f(z) = I(z) - I_{\text{эт}} = 0. \quad (10)$$

Это уравнение проще всего решить методом секущих или модифицированным методом Ньютона последовательными итерациями по формуле:

$$z_{i+1} = z_i - f(z_i) \cdot \frac{z_i - z_{i-1}}{f(z_i) - f(z_{i-1})}.$$

Как видно, что перед тем, как приступить к последовательным итерациям, нам нужно знать два предыдущих приближения решения уравнения.

Пусть,$f(z_1) = I_1 - I_{\text{эт}}$и$f(z_2) = I_2 - I_{\text{эт}}$– первое и второе приближение соответственно. Принимая во внимание итерационную формулу метода секущих, можно записать:

$$z_{i+1} = z_i - \frac{(I_i - I_{\text{эт}}) \cdot (z_i - z_{i-1})}{I_i - I_{i-1}},$$

где $i = 2,3,4, \dots$.

Для метода секущих требуется знать два приближения z_1 и z_2. При этом очень важно, чтобы эти числа были близко расположены к решению уравнения (10), в противном случае алгоритм будет работать неустойчиво, возможно, бесконечное зацикливание, или будет найдена совсем другая высота z, которая тоже является решением уравнения (10). Чтобы избежать этих неприятностей, нужно выбирать малый шаг движения иглы в направлении оси OX, чтобы z_1близко находилось к решению уравнения (10). z_2следует выбирать, как

$$z_2 = z_1 + \delta, \text{если } I_1 > I_{\text{эт}}$$

$$z_2 = z_1 - \delta, \text{если } I_1 < I_{\text{эт}}$$

где δ достаточно мало, чтобы не отойти далеко от решения интеграла.

У этого метода имеются следующие недостатки:

— малый шаг вдоль оси OX означает большое количество вычислений, а следовательно проигрыш в скорости вычислений;

— требование малого шага – является необходимым, но недостаточным условием сходимости метода секущих т.к. по мере продвижения иглы вдоль оси OX все равно возможны большие скачки значений туннельного тока из-за неравномерности поверхности. Поэтому, z_1 будет отклоняться от решения интеграла на различные величины из широкого диапазона. Это может привести к затягиванию итерационного процесса, что неизбежно влечет потерю производительности.

3. Ход работы

Для выполнения лабораторной работы используются следующие формулы:

— Формулы для вычисления$j\big(Z(x,y)\big)$, далее $j(Z)$:

$$j(Z) = 1620 \cdot U \cdot E_f \cdot \exp\left(-1.025 \cdot Z(x,y) \cdot \sqrt{\bar{\varphi}(Z)}\right), \tag{11}$$

где $U = 0.01$ В, 0.1 В, уровень Ферми $E_f = 5.71$.

$$\bar{\varphi}(Z) = \varphi - \frac{U \cdot (S_1 + S_2)}{2 \cdot Z} - \frac{2.86}{k \cdot (S_2 - S_1)} \cdot \ln\left(\frac{S_2 \cdot (Z - S_1)}{S_1 \cdot (Z - S_2)}\right), \tag{12}$$

где $k = 1$ – диэлектрическая постоянная, $\varphi = 4.5$ – локальный выход электронов.

$$S_1 = \frac{3}{k \cdot \varphi}, \tag{13}$$

$$S_2 = Z \cdot \left(1 - \frac{23}{3 \cdot \varphi \cdot k \cdot Z + 10 - 2 \cdot U \cdot k * Z}\right) + S_1 \tag{14}$$

$$Z(x,y) = \sqrt{x^2 + y^2 + z_0^2} \tag{15}$$

Функция (15)служит для определения высоты иглы и используется в интегрировании в качестве аргумента функции (11).

— Основной интеграл для вычисления тока:

$$I = \int_{-Yn}^{Yn} \int_{-Xn}^{Xn} J(Z)dxdy, \tag{16}$$

В особых случаях, когда необходимо определить высоту иглы y угловой поверхности, происходит перестановка плоскостей и формула (15) приобретает следующий вид:

$$Z(x,y) = \sqrt{(d_0 - x_{\text{тек}})^2 + {z_{\text{тек}}}^2 + y^2} \tag{17}$$

где d_0 – диапазон интегрирования по $x, x_{\text{тек}}$ – текущее значение иглы по $x, z_{\text{тек}}$ – текущее значение иглы по z.

В этом случае формула (17) для вычисления тока приобретает вид:

$$I = \int_{-Yn}^{Yn} \int_{-Xn}^{d_0 - x_{\text{тек}}} J(Z)dxdy, \tag{18}$$

Программа написана на ЯВУ C# в MicrosoftVisualStudio 2019 и использованием WinForms и представлена в Приложении.

4. Результаты моделирования

При $U = 0.01$ В получены следующие профилограммы СТМ, представленные на рисунке 3.

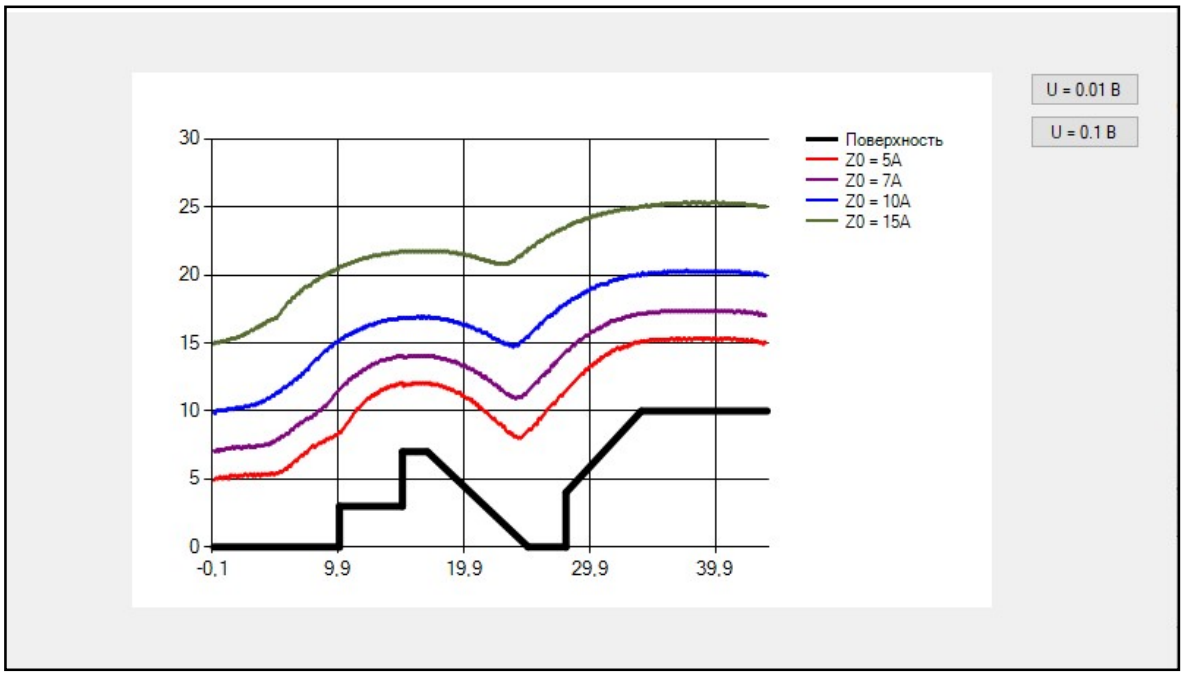

Рисунок 3

При $U = 0.1$ В получены следующие профилограммы СТМ, представленные на рисунке 4.

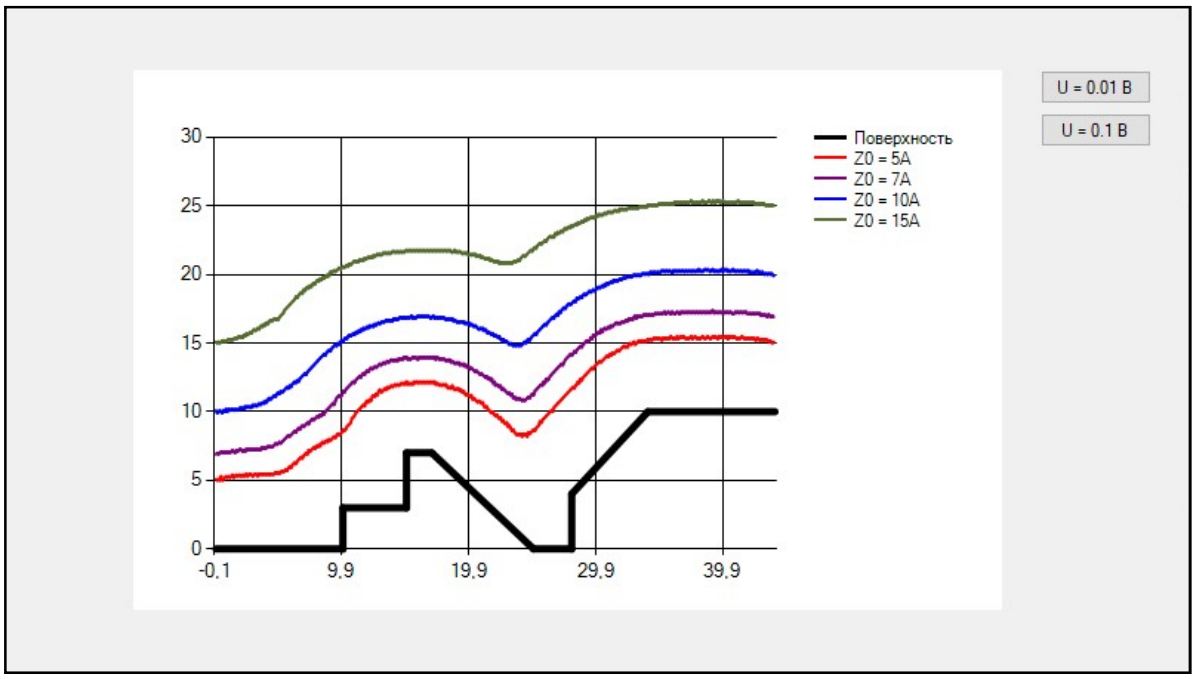

Рисунок 4

Заключение

В ходе выполнения лабораторной работы была достигнута задача моделирования работы иглы ИИС Сканирующий туннельный микроскоп с помощью вычислительного метода Монте-Карло, а также продемонстрированы результаты работы.

Список используемой литературы

1. Искусство программирования: Дональд Кнут. Том 2, М.: "Вильямс", 2007г.

2. Конспект лекций по моделированию и ИИС, Е.Ю. Шелковников;

3. Многопоточность. Введение в многопоточность. – URL: https://metanit.com/sharp/tutorial/11.1.php;

4. Многопоточность в элементах управления Windows Forms– URL: https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/controls/multithreading-in-windows-forms-controls?view=netframeworkdesktop-4.8;

Приложение А

Файл Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Security.Cryptography;

namespace STM
{
    public partial class Form1 : Form
    {
        public static int[] place = new int[22] { 0, 0, 10, 0, 10, 3, 15, 3, 15, 7, 17, 7, 25, 0, 28, 0,
28, 4, 34, 10, 44, 10 };

        RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider();

        public static readonly int N = 1000;

        public static byte[] rngBytes = new byte[2];

        public static double u = 0.1, ef = 5.71, fi = 4.5, k = 1;

        public static double[] z0 = new double[4] { 5, 7, 10, 15 };

        public double I_et = 0;

        public double delta = 0.02, EPS = 0.001;

        public int line_count;

        public double rangeY_integral = 10, rangeX_integral = 10;

        public Form1()
        {
            InitializeComponent();
        }
        public struct Line
        {
            public double dx, dy, x0, y0, x1, y1, Length, nX, nY;
        }
        public Line[] Line_ = new Line[11];
        public void SetLine(double x0, double y0, double x1, double y1, int counter)
        {
            Line_[counter].dx = x1 - x0;
            Line_[counter].dy = y1 - y0;
            Line_[counter].x0 = x0;
            Line_[counter].x1 = x1;
            Line_[counter].y0 = y0;
            Line_[counter].Length = Math.Sqrt(Line_[counter].dx * Line_[counter].dx + Line_[counter].dy *
Line_[counter].dy);
            Line_[counter].dx /= Line_[counter].Length;
            Line_[counter].dy /= Line_[counter].Length;
            Line_[counter].nX = -Line_[counter].dy;
            Line_[counter].nY = Line_[counter].dx;
        }
        private void chart1_Click(object sender, EventArgs e)
        public double J(double x, double y, double z) {
            double zxy, s1, s2, fi_s;
            zxy = Math.Sqrt(x * x + y * y + z * z);
            s1 = 3 / (k * fi);
            s2 = zxy * (1 - 23 / (3 * fi * k * zxy + 10 - 2 * u * k * zxy)) + s1;
            fi_s = fi - (u * (s1 + s2) / (2 * zxy))
                - (2.86 / (k * (s2 - s1)))
                * Math.Log(s2 * (zxy - s2) / (s1 * (zxy - s2)));
            return 1620 * u * ef * Math.Exp(-1.025 * zxy * Math.Sqrt(fi_s));
```

```
        }

        public double Integrals(double xi, double yi) {
            double sum = 0;
            double z, x1, x2;
            for (int i = 0; i < line_count; i++)
            {
                double d = (xi - Line_[i].x0) * Line_[i].nX + (yi - Line_[i].y0) * Line_[i].nY;
                if (d > 0)
                {
                    z = ((xi - d * Line_[i].nX) - Line_[i].x0) * Line_[i].dx + ((yi - d * Line_[i].nY) -
Line_[i].y0) * Line_[i].dy;
                    if (z + rangeX_integral > Line_[i].Length) x1 = Line_[i].Length;
                    else x1 = z + rangeX_integral;
                    x2 = (z - rangeX_integral > 0) ? z - rangeX_integral : 0;
                    x1 -= z;
                    x2 -= z;
                    if (x2 < x1) sum += GetIntegral(d, x2, x1);
                    //if(line_count > 1)Console.WriteLine($"Ток {i} -й линии: {GetIntegral(d, x2, x1)}");
                }

            }
            return sum;
        }
        public double GetIntegral(double z, double x_min, double x_max) {
            double sum = 0, x, y, rng_coef;
            int errorCounter = 0;
            for (int i = 0; i < N; i++) {
                rng.GetBytes(rngBytes);
                rng_coef = rngBytes[0];
                rng_coef /= 255;
                x = x_min + (x_max - x_min) * rng_coef;
                rng_coef = rngBytes[1];
                rng_coef /= 255;
                y = rangeY_integral * rng_coef;
                if (x < x_min || x > x_max) errorCounter++;
                else
                {
                    sum += J(x, y, z);
                }
            }
            return sum* (x_max - x_min) * rangeY_integral / N;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            chart1.Series[1].Points.Clear();
            chart1.Series[2].Points.Clear();
            chart1.Series[3].Points.Clear();
            chart1.Series[4].Points.Clear();
            button1.Enabled = false;
            u = 0.01;
            double b = 44, h = 0.1;
            double delt = 0.1;
            this.chart1.Invoke((MethodInvoker)delegate
            {
                Profilogram1(b, z0[0], Integrals(0, z0[0]), delt, h, 1);
            });
        }
        private void button2_Click(object sender, EventArgs e)
        {
            chart1.Series[1].Points.Clear();
            chart1.Series[2].Points.Clear();
            chart1.Series[3].Points.Clear();
            chart1.Series[4].Points.Clear();
            button2.Enabled = false;
            u = 0.1;
            double b = 44, h = 0.1;
            double delt = 0.1;
            this.chart1.Invoke((MethodInvoker)delegate
            {
                Profilogram1(b, z0[0], Integrals(0, z0[0]), delt, h, 1);
            });
        }
        public async void Profilogram1(double b, double zh, double I_et, double delt, double h, int index)
{
```

```
            double Ik, zh2 = zh, Ik2;
            byte sign;
            button1.Enabled = false;
            button2.Enabled = false;
            for (double i = 0; i < b; i += h)
            {
                Ik = Integrals(i, zh2);
                if (Ik > I_et)
                {
                    zh2 = zh + delt;
                    sign = 0;
                }
                else
                {
                    zh2 = zh - delt;
                    sign = 1;
                }
                Ik2 = Integrals(i, zh2);
                if (Ik2 > I_et && sign == 0)
                {
                    while(Integrals(i, zh2)> I_et) zh2 = zh2 + delt;
                    //zh2 = zh2 + (Ik2 - I_et) * (zh2 - zh) / (Ik2 - Ik);
                    //Ik2 = Integrals(i, zh2);
                }
                else if (Ik2 < I_et && sign == 1)
                {
                    while (Integrals(i, zh2) < I_et) zh2 = zh2 - delt;
                    //zh2 = zh2 + (Ik2 - I_et) * (zh2 - zh) / (Ik2 - Ik);
                    //Ik2 = Integrals(i, zh2);
                }
                chart1.Series[index].Points.AddXY(i, zh2);
                zh = zh2;
                await Task.Delay(1);
            }
            if (index < 4)
            {
                this.chart1.Invoke((MethodInvoker)delegate
                {
                    Profilogram1(b, z0[index], Integrals(0, z0[index]), delt, h, index + 1);
                });
            }
            if (index == 3)
            {
                button1.Enabled = true;
                button2.Enabled = true;
            }
        }
        public void Form1_Load(object sender, EventArgs e)
        {
            this.chart1.Series[0].BorderWidth = 5;
            this.chart1.ChartAreas[0].AxisY.Maximum = 30;
            this.chart1.Series[0].Color = Color.Black;
            chart1.Series[1].BorderWidth = 2;
            chart1.Series[1].Color = Color.Red;
            chart1.Series[2].BorderWidth = 2;
            chart1.Series[2].Color = Color.Purple;
            chart1.Series[3].BorderWidth = 2;
            chart1.Series[3].Color = Color.Blue;
            chart1.Series[4].BorderWidth = 2;
            chart1.Series[4].Color = Color.DarkOliveGreen;
            chart1.Series[0].Name = "Поверхность";
            chart1.Series[1].Name = "Z0 = 5A";
            chart1.Series[2].Name = "Z0 = 7A";
            chart1.Series[3].Name = "Z0 = 10A";
            chart1.Series[4].Name = "Z0 = 15A";
            for (int i = 0; i < 11; i++)
            {
                this.chart1.Series[0].Points.AddXY(place[i * 2], place[i * 2 + 1]);
                if (i > 0) SetLine(place[(i - 1) * 2], place[(i - 1) * 2 + 1], place[i * 2], place[i * 2 +
1], i - 1);
                line_count++;
            }
        }
    }
}
```