МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Новосибирский государственный технический университет»

НГТУ
НЭТИ | Факультет прикладной
математики и информатики

Кафедра прикладной математики
Практическая работа №2
по дисциплине «Цифровые модели и оценивание параметров»

НЕЛИНЕЙНЫЕ ОБРАТНЫЕ ЗАДАЧИ

Группа ПМ-92 АРТЮХОВ РОМАН

Вариант 3 ВАСЬКИН ЛЕОНИД

Преподаватели ВАГИН Д.В.

Новосибирск, 2022

Цель работы

Изучить материал и научиться находить решение в нелинейных обратных задачах.

Задание (Вариант 3)

Положение приемников:

M1(200,0,0), N1(300,0,0);
M2(500,0,0), N2(600,0,0);
M3(1000,0,0), N3(1100,0,0).

Положение источников:

A1(0,-500,0), B1(100,-500,0);
A2(0,0,0), B2(100,0,0);
A3(0,500,0), B3(100,500,0).

Однородное полупространство. Приёмники 1–3. Источник 2. Определить значение σ полупространства. Добавить шум, равный 10% от значения измерения.

Теория

Формула связи электрического тока в источнике и напряжения в приёмнике:

$$V_{AB}^{MN} = \frac{I}{2\pi\sigma}\left(\left(\frac{1}{r_B^M} - \frac{1}{r_A^M}\right) - \left(\frac{1}{r_B^N} - \frac{1}{r_A^N}\right)\right)$$

Производная потенциала по силе тока:

$$\frac{\partial V}{\partial I} = -\frac{1}{2\pi\sigma^2}\left(\left(\frac{1}{r_B^M} - \frac{1}{r_A^M}\right) - \left(\frac{1}{r_B^N} - \frac{1}{r_A^N}\right)\right)$$

Выходить из итерационного процесса будем по малости функционала:

$$\Phi(\sigma) = \sum_{i=1}^{3} \left(\omega_i \cdot \left(V_i(\sigma) - \overline{V}_i\right)\right)^2$$

Тесты

Тест №1 (Без шума)

$$\begin{cases} I = 1 \\ \text{Шум } = 0\% \\ \text{Начальная sigma } = 0,01 \\ \text{Истинная sigma } = 0,1 \\ \text{Точность } = 1e-7 \end{cases}$$

Iteration-Solution		
Iter	Sigma	V
1	1,900000E-002	5,452355E+001
2	3,439000E-002	1,091935E+001
3	5,695328E-002	1,713815E+000
4	8,146980E-002	1,551988E-001
5	9,656632E-002	3,793067E-003
6	9,988210E-002	4,180105E-006
7	9,999986E-002	5,797021E-012

Тест №2 (С шумом +10%)

$$\begin{cases} I = 1 \\ \text{Шум } = +10\% \\ \text{Начальная sigma } = 0,01 \\ \text{Истинная sigma } = 0,1 \\ \text{Точность } = 1e-7 \end{cases}$$

Iteration-Solution		
Iter	Sigma	V
1	1,890000E-002	4,354841E+001
2	3,387069E-002	8,507615E+000
3	5,512192E-002	1,264525E+000
4	7,682115E-002	1,008915E-001
5	8,872592E-002	1,816332E-003
6	9,085666E-002	9,989487E-007
7	9,090906E-002	3,318664E-013

Тест №3 (С шумом -10%)

$$\begin{cases} I = 1 \\ \text{Шум } = -10\% \\ \text{Начальная sigma } = 0,01 \\ \text{Истинная sigma } = 0,1 \\ \text{Точность } = 1e-7 \end{cases}$$

Iteration-Solution		
Iter	Sigma	V
1	1,910000E-002	6,962017E+001
2	3,491671E-002	1,428567E+001
3	5,886083E-002	2,363991E+000
4	8,654028E-002	2,418378E-001
5	1,056776E-001	7,930848E-003
6	1,108454E-001	1,723848E-005
7	1,111105E-001	9,811212E-011

Листинг

Program.cs

```
try
{
    // Путь к задаче
    string path = @"files/task.json";

    // Десериализация данных
    string json = File.ReadAllText(path);
    Data data = JsonConvert.DeserializeObject<Data>(json)!;
    if (data is null) throw new FileNotFoundException("File uncorrected!");

    // Решение нелинейной задачи
    Solve task = new Solve(data, Path.GetDirectoryName(path));
    task.solve(data.Eps);
}
catch (FileNotFoundException ex) {
    WriteLine(ex.Message);
}
```

Solve.cs

```
namespace DigitalModelsNotLinear;

// % ***** Class NotLinearSolution ***** % //
public class Solve
{
    //: Поля и свойства
    private Receiver[] Receivers;   //@ Положение приемников
    private Source[]   Sources;     //@ Положение источника

    private double I            { get; set; } //@ Сила тока
    private double Noise        { get; set; } //@ Шум
    private double SigmaInit    { get; set; } //@ Начальная сигма
    private double SigmaAbsolut { get; set; } //@ Точная сигма

    private Table table;

    public string Path { get; set; } //@ Путь к задаче


    //: Конструктор
    public Solve(Data _data, string _path)  {

        // Данные
        (Receivers, Sources, I, Noise, SigmaInit, SigmaAbsolut) = _data;

```

```
26          // Табличка
27          table = new Table("Iteration-Solution");
28          table.AddColumn(
29              ("Iter", 5),
30              ("Sigma", 16),
31              ("V", 16)
32          );
33
34          // Путь
35          this.Path = _path;
36      }
37
38      //: Основной метод решения
39      public void solve(double EPS) {
40
41          // Подсчет потенциалов от sigmaAbsolut вместе с шумом
42          var V_abs = new Vector<double>(3);
43          for (int i = 0; i < V_abs.Length; i++)
44              V_abs[i] = Potential(Sources[0], Sources[1], Receivers[2 * i], Receivers[2 * i
    ↪  + 1], I, SigmaAbsolut);
45
46          // Добавление шума
47          if (Noise != 0) {
48              double znak = Noise / Abs(Noise);
49              V_abs = V_abs + znak * V_abs * Abs(Noise);
50          }
51
52
53          // Подсчет весов
54          Vector<double> W = 1 / V_abs;
55
56          // Подготовка к итерационому процессу
57          double F, a, b;
58          double SigmaIter = SigmaInit;
59          uint Iter = 0;
60
61          // Подсчет потенциала
62          var V = new Vector<double>(3);
63          for (int i = 0; i < V.Length; i++)
64              V[i] = Potential(Sources[0], Sources[1], Receivers[2 * i], Receivers[2 * i +
    ↪  1], I, SigmaIter);
65
66          // Подсчет производной потенциалов
67          var V_diff = new Vector<double>(3);
68          for (int i = 0; i < V_diff.Length; i++)
69              V_diff[i] = DiffPotential(Sources[0], Sources[1], Receivers[2 * i], Receivers[2
    ↪  * i + 1], I, SigmaIter);
70
71          // Итерационный процесс
72          do
```

```
        {
            // Обнуление компонент
            a = 0;
            b = 0;
            F = 0;

            for (int i = 0; i < 3; i++)
            {
                a += Pow((W[i] * V_diff[i]), 2);
                b += -Pow(W[i], 2) * V_diff[i] * (V[i] - V_abs[i]);
            }

            // Подсчет новой сигмы
            SigmaIter += b / a;

            // Подсчет потенциала
            for (int i = 0; i < V.Length; i++)
                V[i] = Potential(Sources[0], Sources[1], Receivers[2 * i], Receivers[2 * i
↪  + 1], I, SigmaIter);

            // Подсчет производной потенциалов
            for (int i = 0; i < V_diff.Length; i++)
                V_diff[i] = DiffPotential(Sources[0], Sources[1], Receivers[2 * i],
↪  Receivers[2 * i + 1], I, SigmaIter);

            // Подксчет компоненты для выхода
            for (int i = 0; i < 3; i++)
                F += Pow((W[i] * (V[i] - V_abs[i])), 2);

            // Добавление записи в табличку
            table.AddRow((++Iter).ToString(), SigmaIter.ToString("E6"), F.ToString("E6"));

        } while (F > EPS);

        table.WriteToFile(Path + "/solution.txt");
    }
}
```

Helper.cs

```
namespace DigitalModelsNotLinear.other;

// % ***** Стркутура приемника ***** % //
public struct Receiver
{
    //: Поля и свойства
    public double X { get; set; }  //@ Координата X
```

```
    public double Y { get; set; }  //@ Координата Y
    public double Z { get; set; }  //@ Координата Z

    //: Конструкторы
    public Receiver(double _x, double _y, double _z) {
        this.X = _x; this.Y = _y; this.Z = _z;
    }

    public Receiver(double[] point) {
        this.X = point[0]; this.Y = point[1]; this.Z = point[2];
    }

    //: Деконструктор
    public void Deconstruct(out double x,
                            out double y,
                            out double z)
    {
        x = this.X;
        y = this.Y;
        z = this.Z;
    }

    public override string ToString() => $"{X,20} {Y,24} {Z,26}";
}

// % ***** Структура источника ***** % //
public struct Source
{
    //: Поля и свойства
    public double X { get; set; }  //@ Координата X
    public double Y { get; set; }  //@ Координата Y
    public double Z { get; set; }  //@ Координата Z

    //: Конструкторы
    public Source(double _x, double _y, double _z) {
        this.X = _x; this.Y = _y; this.Z = _z;
    }

    public Source(double[] point) {
        this.X = point[0]; this.Y = point[1]; this.Z = point[2];
    }

    //: Деконструктор
    public void Deconstruct(out double x,
                            out double y,
                            out double z)
    {
        x = this.X;
        y = this.Y;
        z = this.Z;
```

```
58      }
59
60      public override string ToString() => $"{X,20} {Y,24} {Z,26}";
61  }
62
63  public static class Helper
64  {
65      // * Расстояние от точки измерения до электрода
66      public static double Interval(Receiver R, Source S) {
67          return Sqrt(Pow(S.X - R.X, 2) + Pow(S.Y - R.Y, 2) + Pow(S.Z - R.Z, 2));
68      }
69
70      // * Расчет потенциала
71      public static double Potential(Source A, Source B, Receiver M, Receiver N, double I,
    ↪  double sigma) {
72          double diff = (1 / Interval(M, B) - 1 / Interval(M, A)) - (1 / Interval(N, B) - 1 /
    ↪  Interval(N, A));
73          return I / (2.0 * PI * sigma) * diff;
74      }
75
76      // * Расчет diff потенциала
77      public static double DiffPotential(Source A, Source B, Receiver M, Receiver N, double
    ↪  I, double sigma) {
78          double diff = (1 / Interval(M, B) - 1 / Interval(M, A)) - (1 / Interval(N, B) - 1 /
    ↪  Interval(N, A));
79          return -1 / (2.0 * PI * sigma * sigma) * diff;
80      }
81  }
```