|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
| Практическое задание № 2 | | |
| по дисциплине «Цифровые модели и оценивание параметров» | | |
| **Нелинейные обратные задачи** | | |
|  | | |
|  |  |  |
| Группа ПМ-05 |  |
| Вариант 1 |  |
|  | болдырев сергей |
|  | грушев андрей |
|  | пучков дмитрий |
| Преподаватели | Вагин Денис Владимирович |
|  |  |
| Новосибирск,2023 | | |

1. **Задание**

Положение приёмников: 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  полупространства. Добавить шум, равный 10 % от значения измерения.

1. **Решение**

Формула связи электрического тока в источнике(AB) и напряжения в приёмнике(MN):

Для решения нашей задачи необходимо собрать СЛАУ следующего вида:

В качестве **ε** мы рассматриваем сумму напряжений в каждом из приёмников от всех источников (в нашем случае от одного источника), – вектор токов.

Производные будут посчитаны аналитически. Формула:

Продолжение итерационного процесса даст истинное значение искомой силы тока.

1. **Результаты работы программы**

Точность решения: 1e-10.

Максимум итераций: 1000.

Тест 1 – Без зашумления

|  |  |
| --- | --- |
| **Итерация** |  |
| 1 | 1,900000E-002 |
| 2 | 3,439000E-002 |
| 3 | 5,695328E-002 |
| 4 | 8,146980E-002 |
| 5 | 9,656632E-002 |
| 6 | 9,988210E-002 |
| 7 | 9,999986E-002 |
| 8 | 1,000000E-001 |

Тест 2 - Зашумление по всем 3 приемникам на плюс 10 процентов.

|  |  |
| --- | --- |
| **Итерация** |  |
| 1 | 1,890000E-002 |
| 2 | 3,387069E-002 |
| 3 | 5,512192E-002 |
| 4 | 7,682115E-002 |
| 5 | 8,872592E-002 |
| 6 | 9,085666E-002 |
| 7 | 9,090906E-002 |
| 8 | 9,090909E-002 |

Тест 3 - Зашумление по всем 3 приемникам на минус 10 процентов.

|  |  |
| --- | --- |
| **Итерация** |  |
| 1 | 1,910000E-002 |
| 2 | 3,491671E-002 |
| 3 | 5,886083E-002 |
| 4 | 8,654028E-002 |
| 5 | 1,056776E-001 |
| 6 | 1,108454E-001 |
| 7 | 1,111105E-001 |
| 8 | 1,111111E-001 |

Тест 4

|  |  |
| --- | --- |
| **Итерация** |  |
| 1 | 1,899671E-002 |
| 2 | 3,437280E-002 |
| 3 | 5,689185E-002 |
| 4 | 8,131041E-002 |
| 5 | 9,628953E-002 |
| 6 | 9,955735E-002 |
| 7 | 9,967202E-002 |
| 8 | 9,967215E-002 |
| 9 | 9,967215E-002 |
| … | … |
| 999 | 9,967215E-002 |
| 1000 | 9,967215E-002 |

1. **Вывод**

При одинаковом по силе и знаку зашумлении на всех приёмниках полученный результат отличается от истинного обратно пропорционально зашумлению. То есть при зашумлении на плюс 10 процентов, то есть умножении разности потенциалов на 1,1, полученное значение удельной теплоёмкости будет равно истинному поделённому на 1,1.

При разном по силе и/или знаку зашумлении хотя бы на двух приёмниках наш алгоритм не выходит по невязке, а выходит только по количеству итераций. Однако он продолжает давать близкие к истинным значениям результаты и при этом довольно быстро начинает стагнировать. Хорошим решением было бы реализовать выход из-за стагнации.

1. **Код программы**

#define\_USE\_MATH\_DEFINES

#include<iostream>

#include<cmath>

#include<vector>

usingnamespace std;

/\*

\* Структура точки

\*/

struct point

{

double x, y, z;

point(doublex, doubley, doublez)

{

this->x = x;

this->y = y;

this->z = z;

}

point()

{

this->x = 0.0;

this->y = 0.0;

this->z = 0.0;

}

};

/\*

\* Структура электрода

\*/

struct electrodes

{

point A;

point B;

electrodes(pointA, pointB)

{

this->A =A;

this->B =B;

}

};

/\*

\* Структура линии

\*/

struct line

{

point M;

point N;

line(pointM, pointN)

{

this->M =M;

this->N =N;

}

};

/\*

\* Расстояние между точками

\*/

double PointsDistance(pointa, pointb)

{

return sqrt(pow(b.x - a.x, 2) +

pow(b.y - a.y, 2) +

pow(b.z - a.z, 2));

}

/\*

\* Вспомогательная функция вычисления значения в больших скобках формулы разности потенциалов

\*/

double brackets(pointA, pointB, pointM, pointN)

{

return ((1 / PointsDistance(B, M) - 1 / PointsDistance(A, M))

-

(1 / PointsDistance(B, N) - 1 / PointsDistance(A, N)));

}

/\*

\* Разность потенциалов

\*/

double PD(electrodesElectrodes, lineLine, doubleamperage, doublesigma)

{

double k = amperage / (2 \* M\_PI \* sigma);

return k \* brackets(Electrodes.A, Electrodes.B, Line.M, Line.N);

}

/\*

\* Производная разности потенциалов по сигма

\*/

double PDDerivativeBySigma(electrodesElectrodes, lineLine, doubleamperage, doublesigma)

{

double k = (-1) \* amperage / (2 \* M\_PI \* sigma \* sigma);

return k \* brackets(Electrodes.A, Electrodes.B, Line.M, Line.N);

}

/\*

\* Производная разности потенциалов по силе тока

\*/

double PDDerivativeByAmperage(electrodesElectrodes, lineLine, doublesigma)

{

double k = 1 / (2 \* M\_PI \* sigma);

return k \* brackets(Electrodes.A, Electrodes.B, Line.M, Line.N);

}

int main()

{

cout << scientific;

unsignedshort nLines = 3; // Кол-во линий в задаче

unsignedshortnParams = 1; // Кол-во неизвестных параметров задачи

unsignedint i, j, k; // Итераторы

#pragmaregion Точки по условию

electrodes Electrodes = electrodes(point(0, 0, 0), point(100, 0, 0));

vector<line> Lines;

Lines.push\_back(line(point(200, 0, 0), point(300, 0, 0)));

Lines.push\_back(line(point(500, 0, 0), point(600, 0, 0)));

Lines.push\_back(line(point(1000, 0, 0), point(1100, 0, 0)));

#pragmaendregion

#pragmaregion Истинные значения

double properAmperage = 5; // Сила тока (I)

doubleproperSigma = 2; // Удельная электрическая проводимость

#pragmaendregion

#pragmaregion Практические данные

vector<double> pracV;

for (i = 0; i < nLines; i++)

pracV.push\_back(PD(Electrodes, Lines[i], properAmperage, properSigma));

#pragmaendregion

#pragmaregion Весовые коэффициенты

vector<double> w;

for (i = 0; i < nLines; i++)

w.push\_back(1 / pracV[i]);

#pragmaendregion

double amperage = 5;

double sigma = properSigma;

double delta = 0.0;

double eps = 1e-7;

int iters = 0;

double functional = 0.0;

for (i = 0; i < nLines; i++)

functional += pow(w[i]\*(PD(Electrodes, Lines[i], amperage, sigma) - pracV[i]), 2);

vector<vector<double>> A;

vector<double> b;

A.resize(nParams);

b.resize(nParams);

for (auto& vec : A)

vec.resize(nParams);

cout << iters <<"\t"<< amperage <<"\t"<< functional << endl;

do

{

functional = 0.0;

for (auto& vec : A)

for (auto& x : vec)

x = 0.0;

for (auto& x : b)

x = 0.0;

for (i = 0; i < nParams; i++)

for (j = 0; j < nParams; j++)

for (k = 0; k < nLines; k++)

A[i][j] += pow(w[k] \* PDDerivativeByAmperage(Electrodes, Lines[k], sigma), 2);

for (i = 0; i < nParams; i++)

for (k = 0; k < nLines; k++)

b[i] -= w[k] \* w[k] \*

PDDerivativeByAmperage(Electrodes, Lines[k], sigma) \*

(PD(Electrodes, Lines[k], amperage, sigma) - pracV[k]);

delta = b[0] / A[0][0];

amperage += delta;

for (i = 0; i < nLines; i++)

functional += pow(w[i] \* (PD(Electrodes, Lines[i], amperage, sigma) - pracV[i]), 2);

iters++;

cout << iters <<"\t"<< amperage <<"\t"<< functional << endl;

} while (delta > eps);

return 0;

}