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

Лабораторная работа №1

по дисциплине «Уравнения математической физики»

РЕШЕНИЕ ЭЛЛИПТИЧЕСКИХ КРАЕВЫХ ЗАДАЧ МЕТОДОМ КОНЕЧНЫХ РАЗНОСТЕЙ

Группа ПМ-92 АРТЮХОВ РОМАН

Вариант 7

Преподаватели ЗАДОРОЖНЫЙ А. Г.

ПАТРУШЕВ И. И.

Новосибирск, 2022

Цель работы:

Разработать программу решения эллиптической краевой задачи методом конечных разностей. Протестировать программу и численно оценить порядок аппроксимации.

Задача (вариант 7):

Область имеет Г-образную форму. Предусмотреть учет первых и вторых краевых условий.

Уравнение: $-div\left(\lambda gradu\right)+\gamma u=f$

Краевые условия:

$$u\big|_{S_1}=u_g$$

$$\lambda\frac{\partial u}{\partial n}\bigg|_{S_2}=\theta$$

Анализ:

Производные первого порядка, аппроксимированные следующими конечными разностями первого порядка:

$$\nabla_h^+ u_i=\frac{u_{i+1}-u_i}{h_i},->правая\ \ разность$$

$$\nabla_h^- u_i=\frac{u_i-u_{i-1}}{h_{i-1}},->левая\ \ разность$$

$$\overline{\nabla}_h u_i=\frac{u_{i+1}-u_{i-1}}{h_i+h_{i-1}},->двусторонняя\ \ разность$$

Пусть область Ω двумерная и определена прямоугольная сетка Ω_h как совокупность точек.

Тогда для двумерного оператора Лапласа:

$$Vu=\frac{\partial^2 x}{\partial x^2}+\frac{\partial^2 x}{\partial y^2}$$

Дискретный аналог на неравномерной прямоугольной сетке может быть определен пятиточечным разностным выражением:

$$V_h u_{i,j}=\frac{2u_{i-1,j}}{h_{i-1}^x\left(h_i^x+h_{i-1}^x\right)}+\frac{2u_{i,j-1}}{h_{j-1}^y\left(h_j^y+h_{j-1}^y\right)}+\frac{2u_{i+1,j}}{h_i^x\left(h_i^x+h_{i-1}^x\right)}+\frac{2u_{i,j+1}}{h_j^y\left(h_j^y+h_{j-1}^y\right)}-\left(\frac{2}{h_{i-1}^x h_i^x}+\frac{2}{h_{j-1}^y h_j^y}\right)u_{i,j}$$

Подставив в уравнение, получим:

$$\frac{-2\lambda u_{i-1,j}}{h_{i-1}^x\left(h_i^x+h_{i-1}^x\right)}+\frac{-2\lambda u_{i,j-1}}{h_{j-1}^y\left(h_j^y+h_{j-1}^y\right)}+\frac{-2\lambda u_{i+1,j}}{h_i^x\left(h_i^x+h_{i-1}^x\right)}+\frac{-2\lambda u_{i,j+1}}{h_j^y\left(h_j^y+h_{j-1}^y\right)}-\left(\frac{-2\lambda}{h_{i-1}^x h_i^x}+\frac{-2\lambda}{h_{j-1}^y h_j^y}\right)u_{i,j}=f_{i,j}$$

Учет краевых условий:

Для узлов, расположенных на границе **S_1**, на которых заданы краевые условия первого рода, соответствующие разностные уравнения заменяются соотношениями, точно передающими краевые условия, т.е. диагональные элементы матрицы, соответствующие

этим узлам, заменяются на **1**, а соответствующий элемент вектора правой части заменяется на значение $\mathbf{u_g}$ функции в этом узле.

Если расчетная область представляет собой прямоугольник со сторонами, параллельными координатным осям, то направление нормали к границе $\mathbf{S_2}$, на которых заданы краевые условия второго рода, совпадает с одной из координатных линий, и тогда методы аппроксимации производной по нормали $\frac{\partial u}{\partial n}$ (которая в этом случае будет равна либо $\pm\frac{\partial u}{dx}$, либо $\pm\frac{\partial u}{dy}$) сводятся к одномерным.

Helper.cs

```
namespace FDM;
public struct Matrix /// Структура матрицы
{
    public int N;
    public int maxIter;
    public double EPS;
    public int shift1, dshift, shift2;
    public double[] di, du1, du2, dl1, dl2, pr, x, absolut_x;
}

public static class Helper
{
    //* Вычисление нормы вектора
    public static double Norm(double[] array) {
        double norm = 0;
        for (int i = 0; i < array.Count(); i++)
            norm += array[i] * array[i];
        return Sqrt(norm);
    }
}
```

Data.cs

```
namespace FDM;
public class Data
{
    //* Данные задачи
    public int         CountX { get; set; }   /// Количество точек на оси X
    public int         CountY { get; set; }   /// Количество точек на оси Y
    public double[]    X      { get; set; }   /// Значения X-ов
    public double[]    Y      { get; set; }   /// Значения Y-ов
    public int         GX     { get; set; }   /// К-во точек на нижней границе области "Г"
    public int         GY     { get; set; }   /// К-во точек на правой границе области "Г"

    public void Deconstruct(out int      countX,
                            out int      countY,
                            out double[] x,
                            out double[] y,
                            out int      gx,
                            out int      gy)
    {
        countX = CountX;
        countY = CountY;
        x = X;
        y = Y;
        gx = GX;
        gy = GY;
    }
}
```

Function.cs

```
namespace FDM;
public static class Function
{
    public static uint   NumberFunc;     /// Номер задачи
    public static double lambda;         /// Лямбда
    public static double gamma;          /// Гамма

    //* Инициализации лямбды и гаммы
    public static void Init() {
        switch(NumberFunc)
        {
            case 1: /// easy
            lambda = 2; gamma = 3;
            break;

            case 2: /// sec_kraev
            lambda = 2; gamma = 3;
            break;

            case 3: /// polynom_2
            lambda = 2; gamma = 3;
            break;

            case 4: /// polynom_3
            lambda = 2; gamma = 3;
            break;

            case 5: /// polynom_4
            lambda = 2; gamma = 3;
            break;

            case 6: /// not_polynom
            lambda = 1; gamma = 1;
            break;
        }
    }

    //* Функция u(x,y)
    public static double Absolut(double x, double y, uint side = 0)
    {
        switch(NumberFunc)
        {
            case 1: /// easy
            return 2*x + 4*y;

            case 2: /// sec_kraev
            return side switch
            {
                3 => 4,
                6 => 8,
                _ => 2*x + 4*y
            };

            case 3: /// polynom_2
            return 2*Pow(x, 2) + 4*Pow(y, 2);

            case 4: /// polynom_3
            return 2*Pow(x, 3) + 4*Pow(y, 3);

            case 5: /// polynom_4
            return 2*Pow(x, 4) + 4*Pow(y, 4);

            case 6: /// not_polynom
            return Sin(x + y);
```

```
    }
    return 0;
}

//* Функция f(x,y)
public static double Func(double x, double y)
{
    switch(NumberFunc)
    {
        case 1: /// easy
        return 6*x + 12*y;

        case 2: /// sec_kraev
        return 6*x + 12*y;

        case 3: /// polynom_2
        return 6*Pow(x, 2) + 12*Pow(y, 2) - 24;

        case 4: /// polynom_3
        return -24*x - 48*y + 6*Pow(x, 3) + 12*Pow(y, 3);

        case 5: /// polynom_4
        return -48*Pow(x, 2) - 96*Pow(y, 2) + 6*Pow(x, 4) + 12*Pow(y, 4);

        case 6: /// not_polynom
        return 3*Sin(x + y);
    }
    return 0;
}

//* Функция проверки имеется ли на стороне области второе краевое
public static bool IsSecondKraev(uint side)
{
    switch(NumberFunc)
    {
        case 1: /// easy
        return side switch
        {
            _ => false,
        };

        case 2: /// sec_kraev
        return side switch
        {
            3 => true,
            6 => true,
            _ => false
        };

        case 3: /// polynom_2
        return side switch
        {
            _ => false,
        };

        case 4: /// polynom_3
        return side switch
        {
            _ => false,
        };

        case 5: /// polynom_4
        return side switch
        {
            _ => false,
        };

        case 6: /// not_polynom
```

```
            return side switch
            {
                _ => false,
            };
        }
        return false;
    }
}
```

Seidel.cs

```
namespace FDM;
public class Seidel
{
    private Matrix matrix; /// Матрица
    private double omega;  /// Параметр релаксации

    public Seidel(Matrix matrix, int iter, double eps, double omega = 1) {
        this.matrix         = matrix;
        this.omega          = omega;
        this.matrix.maxIter = iter;
        this.matrix.EPS     = eps;
    }

    //* Решение СЛАУ
    public void solve(bool flag = false) {
        double sum, Nev = 0, norm_f;
        int Iter = 0;
        norm_f = Norm(matrix.pr);

        do {
            Nev = 0;
            for (int i = 0; i < matrix.N; i++) {
                sum = matrix.di[i] * matrix.x[i];

                if (i < matrix.N - 1)
                    sum += matrix.du1[i] * matrix.x[i + 1];
                if (i >= 2)
                    sum += matrix.dl1[i - 1] * matrix.x[i - 1];

                if (i < matrix.dshift)
                    sum += matrix.du2[i] * matrix.x[matrix.shift1 + i];
                else if (i < matrix.N - matrix.shift2)
                    sum += matrix.du2[i] * matrix.x[matrix.shift2 + i];

                if (i >= matrix.shift1 + matrix.dshift)
                    sum += matrix.dl2[i - matrix.shift1] * matrix.x[i - matrix.shift2];
                else if (i >= matrix.shift1)
                    sum += matrix.dl2[i - matrix.shift1] * matrix.x[i - matrix.shift1];

                Nev += (matrix.pr[i] - sum) * (matrix.pr[i] - sum);
                matrix.x[i] += omega / matrix.di[i] * (matrix.pr[i] - sum);
            }

            Nev = Sqrt(Nev) / norm_f; /// Относительная невязка
            Iter++;
            if (flag)
                WriteLine($"Iter: {Iter, -10} Nev: {Nev.ToString("E3")}");
        } while (Nev > matrix.EPS &&
                    Iter <= matrix.maxIter);
    }
}
```

Solve.cs

```
namespace FDM;
public class Solve
{
    public int         CountX { get; set; }   /// Количество точек на оси X
    public int         CountY { get; set; }   /// Количество точек на оси Y
    public double[]    X      { get; set; }   /// Значения X-ов
    public double[]    Y      { get; set; }   /// Значения Y-ов
    public int         GX     { get; set; }   /// К-во точек на нижней границе области "Г"
    public int         GY     { get; set; }   /// К-во точек на правой границе области "Г"
    private Matrix matrix;   /// 5-диагональная матрица

    public Solve(Data data, uint Num) {
        (CountX, CountY, X, Y, GX, GY) = data;
        Function.NumberFunc = Num;
        Function.Init();
    }

    //* Решение задачи
    public void solve() {
        memory();                                          //? Выделение памяти
        completion();                                      //? Заполнение матрицы
        var task = new Seidel(matrix, 10000, 1e-14);       //? Создание метода Гаусса-Зейделя
        task.solve(true);                                  //? Решение СЛАУ
        writeTable();                                      //? Записб таблички с решением
    }

    //* Заполнение матрицы (область "Г") снизу->вверх
    private void completion() {
        int id = 0;                //: индекс узла
        double hx1, hx2, hy1, hy2; //: h-ки приращения аргументов

        // Нижняя линия области "Г"
        matrix.pr[0] = Absolut(X[0], Y[0]);       // Левый нижний угол
        hy1 = Abs(Y[0] - Y[1]);
        for (int i = 1; i < GX - 1; i++) {
            matrix.pr[i] = Absolut(X[i], Y[0], 1);
            if (IsSecondKraev(1)) {
                matrix.di[i]  = -lambda / hy1;
                matrix.du2[i] =  lambda / hy1;
            }
        }
        id = GX - 1;
        matrix.pr[id] = Absolut(X[GX - 1], Y[0]);  // Правый нижний угол
        id++;

        // До линии между шапкой и ножкой
        for (int i = 1; i < GY - 1; i++, id++) {
            hy1 = Abs(Y[i] - Y[i - 1]);
            hy2 = Abs(Y[i + 1] - Y[i]);
            matrix.pr[id] = Absolut(X[0], Y[i], 2);
            if (IsSecondKraev(2)) {
                hx1 = Abs(X[0] - X[1]);
                matrix.di [id] = -lambda / hx1;
                matrix.du1[id] =  lambda / hx1;
            }
            id++;

            for (int j = 1; j < GX - 1; j++, id++) {
                hx1 = Abs(X[j] - X[j - 1]);
                hx2 = Abs(X[j + 1] - X[j]);
                matrix.pr [id]                 = Func(X[j], Y[i]);
                matrix.dl1[id - 1]             = -2*lambda / (hx1 * (hx2 + hx1));
                matrix.dl2[id - matrix.shift1] = -2*lambda / (hy1 * (hy2 + hy1));
                matrix.du1[id]                 = -2*lambda / (hx2 * (hx2 + hx1));
```

```
            matrix.du2[id]                 = -2*lambda / (hy2 * (hy2 + hy1));
            matrix.di [id]                 = lambda * (2/(hx1*hx2) + 2/(hy1*hy2)) + gamma;
        }

        matrix.pr[id] = Absolut(X[GX - 1], Y[i], 3);
        if (IsSecondKraev(3)) {
            hx1 = Abs(X[GX - 1] - X[GX - 2]);
            matrix.di[id] = lambda / hx1;
            matrix.dl1[id - 1] = -lambda / hx1;
        }
    }

    matrix.dshift = id;

    // Между шляпкой и ножкой
    hy1 = Abs(Y[GY - 1] - Y[GY - 2]);
    hy2 = Abs(Y[GY] - Y[GY - 1]);
    matrix.pr[id] = Absolut(X[0], Y[GY - 1], 2);
    if (IsSecondKraev(2)) {
        hx1 = Abs(X[0] - X[1]);
        matrix.di[id] = -lambda / hx1;
        matrix.du1[id] = lambda / hx1;
    }
    id++;

    for (int i = 1; i < GX; i++, id++) {
        hx1 = Abs(X[i] - X[i - 1]);
        hx2 = Abs(X[i + 1] - X[i]);
        matrix.pr[id]                  = Func(X[i], Y[GY - 1]);
        matrix.dl1[id - 1]             = -2*lambda / (hx1 * (hx2 + hx1));
        matrix.dl2[id - matrix.shift1] = -2*lambda / (hy1 * (hy2 + hy1));
        matrix.du1[id]                 = -2*lambda / (hx2 * (hx2 + hx1));
        matrix.du2[id]                 = -2*lambda / (hy2 * (hy2 + hy1));
        matrix.di [id]                 = lambda * (2/(hx1*hx2) + 2/(hy1*hy2)) + gamma;
    }

    for (int i = GX; i < CountX - 1; i++, id++) {
        matrix.pr[id] = Absolut(X[i], Y[GY - 1], 4);
        if (IsSecondKraev(4)) {
            matrix.di[id] = -lambda / hy2;
            matrix.du2[id] = lambda / hy2;
        }
    }
    matrix.pr[id] = Absolut(X[CountX - 1], Y[GY - 1]);
    id++;

    // Шляпка
    for (int i = GY; i < CountY - 1; i++) {
        hy1 = Abs(Y[i] - Y[i - 1]);
        hy2 = Abs(Y[i + 1] - Y[i]);
        matrix.pr[id] = Absolut(X[0], Y[i], 2);
        if (IsSecondKraev(2)) {
            hx1 = Abs(X[0] - X[1]);
            matrix.di[id] = -lambda / hx1;
            matrix.du1[id] = lambda / hx1;
        }
        id++;

        for (int j = 1; j < CountX - 1; j++, id++) {
            hx1 = Abs(X[j] - X[j - 1]);
            hx2 = Abs(X[j + 1] - X[j]);
            matrix.pr[id]                  = Func(X[j], Y[i]);
            matrix.dl1[id - 1]             = -2*lambda / (hx1 * (hx2 + hx1));
            matrix.dl2[id - matrix.shift1] = -2*lambda / (hy1 * (hy2 + hy1));
            matrix.du1[id]                 = -2*lambda / (hx2 * (hx2 + hx1));
            matrix.du2[id]                 = -2*lambda / (hy2 * (hy2 + hy1));
            matrix.di [id]                 = lambda * (2/(hx1*hx2) + 2/(hy1*hy2)) + gamma;
        }
```

```
        matrix.pr[id] = Absolut(X[CountX - 1], Y[i], 5);
        if (IsSecondKraev(5)) {
            hx1 = Abs(X[CountX - 1] - X[CountX - 2]);
            matrix.di[id] = lambda / hx1;
            matrix.dl1[id - 1] = -lambda / hx1;
        }
        id++;
    }

    // Верхушка шляпки
    matrix.pr[id] = Absolut(X[0], Y[CountY - 1]);
    id++;
    hy1 = Abs(Y[CountY - 1] - Y[CountY - 2]);
    for (int i = 1; i < CountX - 1; i++, id++) {
        matrix.pr[id] = Absolut(X[i], Y[CountY - 1], 6);
        if (IsSecondKraev(6)) {
            matrix.di[id] = lambda / hy1;
            matrix.dl2[id - matrix.shift1] = -lambda / hy1;
        }
    }
    matrix.pr[id] = Absolut(X[CountX - 1], Y[CountY - 1]);
}

//* Выделяем память под матрицу
private void memory() {
    matrix.N = GX * (CountY - GY) + CountX * GY;  // Размерность матрицы
    matrix.shift1    = GX;
    matrix.shift2    = CountX;
    matrix.di        = new double[matrix.N];
    matrix.du1       = new double[matrix.N];
    matrix.du2       = new double[matrix.N];
    matrix.dl1       = new double[matrix.N];
    matrix.dl2       = new double[matrix.N];
    matrix.pr        = new double[matrix.N];
    matrix.x         = new double[matrix.N];
    matrix.absolut_x = new double[matrix.N];
    Array.Fill(matrix.di, 1); // Заполнение диагонали единицами
}

//* Заполнение и запись таблички с решением
private void writeTable() {
    StringBuilder table = new StringBuilder();
    string margin = String.Join("", Enumerable.Repeat("-", 16));

    table.Append(String.Join("", Enumerable.Repeat("-", 86)) + "\n");
    table.Append($"|X{" ",-14} | Y{" ",-13} | U{" ",-12}  | U`{" ",-12} | |U`- U| {" ",-7}|\n");
    table.Append($"|" + margin + "|" + margin + "|" + margin + "|" + margin + "|" + margin + "|\n");

    int id      = 0;
    for (int i = 0; i < CountY - GY; i++) {
        for (int j = 0; j < GX; j++, id++) {
            double absolut = Absolut(X[j], Y[i]);
            table.Append($"|{String.Format("{0,16}", X[j])}" +
                         $"|{String.Format("{0,16}", Y[i])}" +
                         $"|{String.Format("{0,16}", matrix.x[id].ToString("E6"))}" +
                         $"|{String.Format("{0,16}", absolut.ToString("E6"))}" +
                         $"|{String.Format("{0,16}", Abs(absolut - matrix.x[id]).ToString("E6"))}|\n");
        }
    }

    for (int i = GY - 1; i < CountY; i++) {
        for (int j = 0; j < CountX; j++, id++) {
            double absolut = Absolut(X[j], Y[i]);
            table.Append($"|{String.Format("{0,16}", X[j])}" +
                         $"|{String.Format("{0,16}", Y[i])}" +
                         $"|{String.Format("{0,16}", matrix.x[id].ToString("E6"))}" +
                         $"|{String.Format("{0,16}", absolut.ToString("E6"))}" +
```

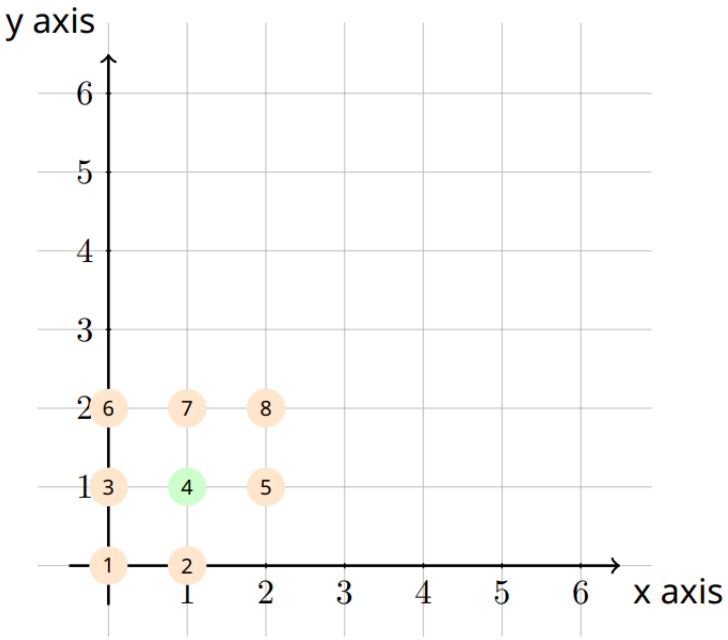
```
                                $"|{String.Format("{0,16}", Abs(absolut - matrix.x[id]).ToString("E6"))}|\n");
            }
        }
        table.Append(String.Join("", Enumerable.Repeat("-", 86)) + "\n");
        File.WriteAllText("test/tables/table.txt", table.ToString());
    }

}
```

Program.cs

```
try
{
    string json = File.ReadAllText(@"test\easy.json");          //: Простейший тест (1)
    //string json = File.ReadAllText(@"test\sec_kraev.json");     //: Тест на вторые краевые (2)
    //string json = File.ReadAllText(@"test\polynom_2.json");     //: Тест на полином второй степени (3)
    //string json = File.ReadAllText(@"test\polynom_3.json");     //: Тест на полином третьей степени (4)
    //string json = File.ReadAllText(@"test\polynom_4.json");     //: Тест на полином четвертой степени (5)
    //string json = File.ReadAllText(@"test\not_polynomh1.json"); //: Тест на не полиномиальной функции (h1 - шаг) (6)
    //string json = File.ReadAllText(@"test\not_polynomh2.json"); //: Тест на не полиномиальной функции (h1/2 - шаг) (6)
    //string json = File.ReadAllText(@"test\not_polynomh3.json"); //: Тест на не полиномиальной функции (h1/4 - шаг) (6)
    //string json = File.ReadAllText(@"test\uneven.json");        //: Тест на неравномерной сетке (1)

    Data data = JsonConvert.DeserializeObject<Data>(json)!;

    if (data is null) throw new FileNotFoundException("File uncorrected!");

    Solve task = new Solve(data, 1);
    task.solve();
}
catch (FileNotFoundException ex)
{
    WriteLine(ex.Message);
}
```

Данные задачи (Простой тест):

$$\begin{cases} u(x,y) = 2x + 4y \\ f(x,y) = 6x + 12y \\ \lambda = 2 \\ \gamma = 3 \end{cases}$$



Краевые условия:

Первого рода на всех ребрах

Содержимое json-файла:

```
{
    "CountX": 3,
    "CountY": 3,
    "X"     : [0, 1, 2],
    "Y"     : [0, 1, 2],
    "GX"    : 2,
    "GY"    : 2
}
```

CountX -> Количество точек на оси X

CountY -> Количество точек на оси Y

X -> Значения X-ов

Y -> Значения Y-ов

GX -> К-во точек на нижней границе области «Г»

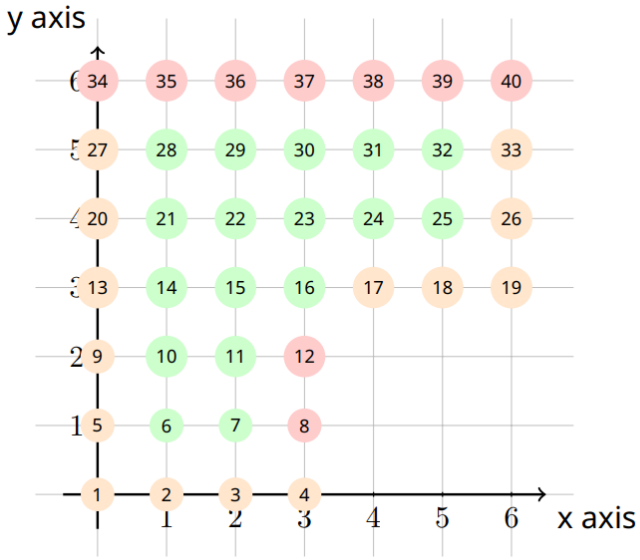
GY -> К-во точек на правой границе области «Г»

Табличка с решением:

X	Y	U	U`	|U` - U|
0	0	0,000000E+000	0,000000E+000	0,000000E+000
1	0	2,000000E+000	2,000000E+000	0,000000E+000
0	1	4,000000E+000	4,000000E+000	0,000000E+000
1	1	6,000000E+000	6,000000E+000	0,000000E+000
2	1	8,000000E+000	8,000000E+000	0,000000E+000
0	2	8,000000E+000	8,000000E+000	0,000000E+000
1	2	1,000000E+001	1,000000E+001	0,000000E+000
2	2	1,200000E+001	1,200000E+001	0,000000E+000

Данные задачи (Тест на вторые краевые):

$$\begin{cases} u(x,y) = 2x + 4y \\ f(x,y) = 6x + 12y \\ \lambda = 2 \\ \gamma = 3 \end{cases}$$



Краевые условия:

Первого рода – оранжевый цвет

Второго рода – красный цвет

Содержимое json-файла:

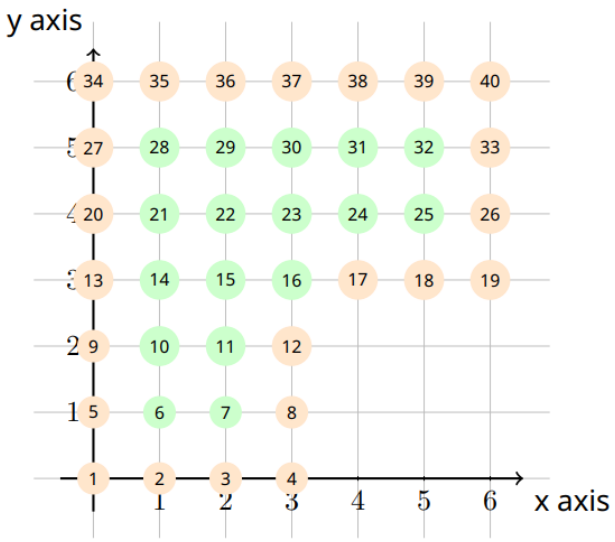
```
{
    "CountX": 7,
    "CountY": 7,
    "X"     : [0, 1, 2, 3, 4, 5, 6],
    "Y"     : [0, 1, 2, 3, 4, 5, 6],
    "GX"    : 4,
    "GY"    : 4
}
```

Табличка с решением:

X	Y	U	U`	|U` - U|
0	0	0,000000E+000	0,000000E+000	0,000000E+000
1	0	2,000000E+000	2,000000E+000	0,000000E+000
2	0	4,000000E+000	4,000000E+000	0,000000E+000
3	0	6,000000E+000	6,000000E+000	0,000000E+000
0	1	4,000000E+000	4,000000E+000	0,000000E+000
1	1	6,000000E+000	6,000000E+000	3,375078E-014
2	1	8,000000E+000	8,000000E+000	4,529710E-014
3	1	1,000000E+001	1,000000E+001	4,618528E-014
0	2	8,000000E+000	8,000000E+000	0,000000E+000
1	2	1,000000E+001	1,000000E+001	4,440892E-014
2	2	1,200000E+001	1,200000E+001	5,684342E-014
3	2	1,400000E+001	1,400000E+001	5,684342E-014
0	3	1,200000E+001	1,200000E+001	0,000000E+000
1	3	1,400000E+001	1,400000E+001	4,263256E-014
2	3	1,600000E+001	1,600000E+001	4,796163E-014
3	3	1,800000E+001	1,800000E+001	3,197442E-014
4	3	2,000000E+001	2,000000E+001	0,000000E+000
5	3	2,200000E+001	2,200000E+001	0,000000E+000
6	3	2,400000E+001	2,400000E+001	0,000000E+000
0	4	1,600000E+001	1,600000E+001	0,000000E+000
1	4	1,800000E+001	1,800000E+001	4,263256E-014
2	4	2,000000E+001	2,000000E+001	4,973799E-014
3	4	2,200000E+001	2,200000E+001	3,907985E-014
4	4	2,400000E+001	2,400000E+001	1,776357E-014
5	4	2,600000E+001	2,600000E+001	7,105427E-015
6	4	2,800000E+001	2,800000E+001	0,000000E+000
0	5	2,000000E+001	2,000000E+001	0,000000E+000
1	5	2,200000E+001	2,200000E+001	4,618528E-014
2	5	2,400000E+001	2,400000E+001	4,973799E-014
3	5	2,600000E+001	2,600000E+001	3,907985E-014
4	5	2,800000E+001	2,800000E+001	2,131628E-014
5	5	3,000000E+001	3,000000E+001	1,065814E-014
6	5	3,200000E+001	3,200000E+001	0,000000E+000
0	6	2,400000E+001	2,400000E+001	0,000000E+000
1	6	2,600000E+001	2,600000E+001	4,618528E-014
2	6	2,800000E+001	2,800000E+001	4,973799E-014
3	6	3,000000E+001	3,000000E+001	3,907985E-014
4	6	3,200000E+001	3,200000E+001	2,131628E-014
5	6	3,400000E+001	3,400000E+001	1,421085E-014
6	6	3,600000E+001	3,600000E+001	0,000000E+000

Данные задачи (Полином второй степени):

$$\begin{cases} u(x,y) = 2x^2 + 4y^2 \\ f(x,y) = 6x^2 + 12y^2 - 24 \\ \lambda = 2 \\ \gamma = 3 \end{cases}$$



Краевые условия:

Первого рода на всех ребрах

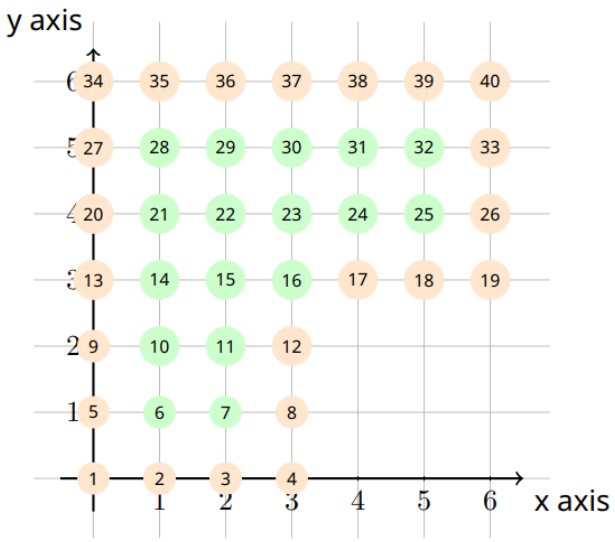
Содержимое json-файла: как в прошлом тесте

Табличка с решением:

X	Y	U	U`	|U`- U|
0	0	0,000000E+000	0,000000E+000	0,000000E+000
1	0	2,000000E+000	2,000000E+000	0,000000E+000
2	0	8,000000E+000	8,000000E+000	0,000000E+000
3	0	1,800000E+001	1,800000E+001	0,000000E+000
0	1	4,000000E+000	4,000000E+000	0,000000E+000
1	1	6,000000E+000	6,000000E+000	1,803002E-013
2	1	1,200000E+001	1,200000E+001	1,083578E-013
3	1	2,200000E+001	2,200000E+001	0,000000E+000
0	2	1,600000E+001	1,600000E+001	0,000000E+000
1	2	1,800000E+001	1,800000E+001	2,060574E-013
2	2	2,400000E+001	2,400000E+001	1,278977E-013
3	2	3,400000E+001	3,400000E+001	0,000000E+000
0	3	3,600000E+001	3,600000E+001	0,000000E+000
1	3	3,800000E+001	3,800000E+001	1,705303E-013
2	3	4,400000E+001	4,400000E+001	1,278977E-013
3	3	5,400000E+001	5,400000E+001	3,552714E-014
4	3	6,800000E+001	6,800000E+001	0,000000E+000
5	3	8,600000E+001	8,600000E+001	0,000000E+000
6	3	1,080000E+002	1,080000E+002	0,000000E+000
0	4	6,400000E+001	6,400000E+001	0,000000E+000
1	4	6,600000E+001	6,600000E+001	9,947598E-014
2	4	7,200000E+001	7,200000E+001	8,526513E-014
3	4	8,200000E+001	8,200000E+001	4,263256E-014
4	4	9,600000E+001	9,600000E+001	1,421085E-014
5	4	1,140000E+002	1,140000E+002	0,000000E+000
6	4	1,360000E+002	1,360000E+002	0,000000E+000
0	5	1,000000E+002	1,000000E+002	0,000000E+000
1	5	1,020000E+002	1,020000E+002	4,263256E-014
2	5	1,080000E+002	1,080000E+002	2,842171E-014
3	5	1,180000E+002	1,180000E+002	0,000000E+000
4	5	1,320000E+002	1,320000E+002	0,000000E+000
5	5	1,500000E+002	1,500000E+002	0,000000E+000
6	5	1,720000E+002	1,720000E+002	0,000000E+000
0	6	1,440000E+002	1,440000E+002	0,000000E+000
1	6	1,460000E+002	1,460000E+002	0,000000E+000
2	6	1,520000E+002	1,520000E+002	0,000000E+000
3	6	1,620000E+002	1,620000E+002	0,000000E+000
4	6	1,760000E+002	1,760000E+002	0,000000E+000
5	6	1,940000E+002	1,940000E+002	0,000000E+000
6	6	2,160000E+002	2,160000E+002	0,000000E+000

Данные задачи (Полином третьей степени):

$$\begin{cases} u(x,y) = 2x^3 + 4y^3 \\ f(x,y) = -24x - 48y + 6x^3 + 12y^3 \\ \lambda = 2 \\ \gamma = 3 \end{cases}$$



Краевые условия:

Первого рода на всех ребрах

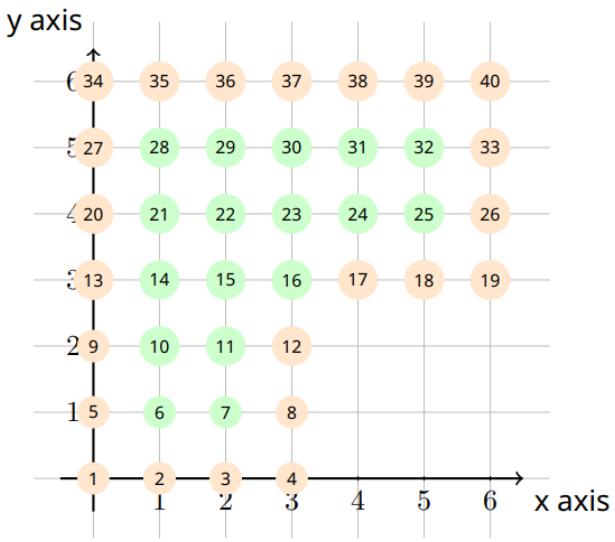
Содержимое json-файла: как в прошлом тесте

Табличка с решением:

X	Y	U	U`	|U` - U|
0	0	0,000000E+000	0,000000E+000	0,000000E+000
1	0	2,000000E+000	2,000000E+000	0,000000E+000
2	0	1,600000E+001	1,600000E+001	0,000000E+000
3	0	5,400000E+001	5,400000E+001	0,000000E+000
0	1	4,000000E+000	4,000000E+000	0,000000E+000
1	1	6,000000E+000	6,000000E+000	2,859935E-013
2	1	2,000000E+001	2,000000E+001	1,740830E-013
3	1	5,800000E+001	5,800000E+001	0,000000E+000
0	2	3,200000E+001	3,200000E+001	0,000000E+000
1	2	3,400000E+001	3,400000E+001	3,268497E-013
2	2	4,800000E+001	4,800000E+001	2,131628E-013
3	2	8,600000E+001	8,600000E+001	0,000000E+000
0	3	1,080000E+002	1,080000E+002	0,000000E+000
1	3	1,100000E+002	1,100000E+002	2,842171E-013
2	3	1,240000E+002	1,240000E+002	2,131628E-013
3	3	1,620000E+002	1,620000E+002	8,526513E-014
4	3	2,360000E+002	2,360000E+002	0,000000E+000
5	3	3,580000E+002	3,580000E+002	0,000000E+000
6	3	5,400000E+002	5,400000E+002	0,000000E+000
0	4	2,560000E+002	2,560000E+002	0,000000E+000
1	4	2,580000E+002	2,580000E+002	1,705303E-013
2	4	2,720000E+002	2,720000E+002	1,705303E-013
3	4	3,100000E+002	3,100000E+002	5,684342E-014
4	4	3,840000E+002	3,840000E+002	0,000000E+000
5	4	5,060000E+002	5,060000E+002	0,000000E+000
6	4	6,880000E+002	6,880000E+002	0,000000E+000
0	5	5,000000E+002	5,000000E+002	0,000000E+000
1	5	5,020000E+002	5,020000E+002	5,684342E-014
2	5	5,160000E+002	5,160000E+002	0,000000E+000
3	5	5,540000E+002	5,540000E+002	0,000000E+000
4	5	6,280000E+002	6,280000E+002	0,000000E+000
5	5	7,500000E+002	7,500000E+002	0,000000E+000
6	5	9,320000E+002	9,320000E+002	0,000000E+000
0	6	8,640000E+002	8,640000E+002	0,000000E+000
1	6	8,660000E+002	8,660000E+002	0,000000E+000
2	6	8,800000E+002	8,800000E+002	0,000000E+000
3	6	9,180000E+002	9,180000E+002	0,000000E+000
4	6	9,920000E+002	9,920000E+002	0,000000E+000
5	6	1,114000E+003	1,114000E+003	0,000000E+000
6	6	1,296000E+003	1,296000E+003	0,000000E+000

Данные задачи (Полином четвертой степени):

$$\begin{cases} u(x,y) = 2x^4 + 4y^4 \\ f(x,y) = -48x^2 - 96y^2 + 6x^4 + 12y^4 \\ \lambda = 2 \\ \gamma = 3 \end{cases}$$



Краевые условия:

Первого рода на всех ребрах

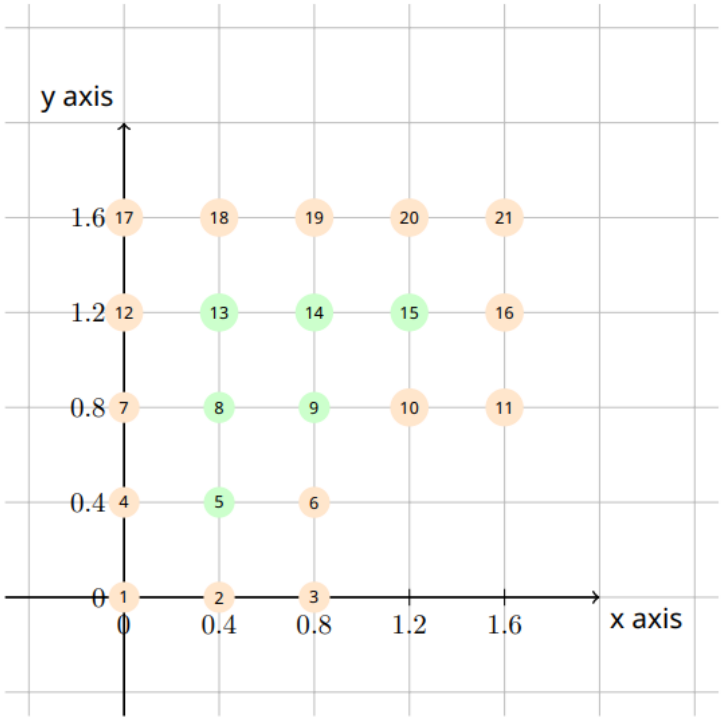
Содержимое json-файла: как в прошлом тесте

Табличка с решением:

X	Y	U	U`	|U` - U|
0	0	0,000000E+000	0,000000E+000	0,000000E+000
1	0	2,000000E+000	2,000000E+000	0,000000E+000
2	0	3,200000E+001	3,200000E+001	0,000000E+000
3	0	1,620000E+002	1,620000E+002	0,000000E+000
0	1	4,000000E+000	4,000000E+000	0,000000E+000
1	1	9,699224E+000	6,000000E+000	3,699224E+000
2	1	3,972019E+001	3,600000E+001	3,720188E+000
3	1	1,660000E+002	1,660000E+002	0,000000E+000
0	2	6,400000E+001	6,400000E+001	0,000000E+000
1	2	7,062554E+001	6,600000E+001	4,625542E+000
2	2	1,007618E+002	9,600000E+001	4,761812E+000
3	2	2,260000E+002	2,260000E+002	0,000000E+000
0	3	3,240000E+002	3,240000E+002	0,000000E+000
1	3	3,309794E+002	3,260000E+002	4,979443E+000
2	3	3,618442E+002	3,560000E+002	5,844237E+000
3	3	4,903070E+002	4,860000E+002	4,306995E+000
4	3	8,360000E+002	8,360000E+002	0,000000E+000
5	3	1,574000E+003	1,574000E+003	0,000000E+000
6	3	2,916000E+003	2,916000E+003	0,000000E+000
0	4	1,024000E+003	1,024000E+003	0,000000E+000
1	4	1,030917E+003	1,026000E+003	4,917159E+000
2	4	1,062095E+003	1,056000E+003	6,095053E+000
3	4	1,191844E+003	1,186000E+003	5,844237E+000
4	4	1,540762E+003	1,536000E+003	4,761812E+000
5	4	2,277720E+003	2,274000E+003	3,720188E+000
6	4	3,616000E+003	3,616000E+003	0,000000E+000
0	5	2,500000E+003	2,500000E+003	0,000000E+000
1	5	2,505970E+003	2,502000E+003	3,969876E+000
2	5	2,536917E+003	2,532000E+003	4,917159E+000
3	5	2,666979E+003	2,662000E+003	4,979443E+000
4	5	3,016626E+003	3,012000E+003	4,625542E+000
5	5	3,753699E+003	3,750000E+003	3,699224E+000
6	5	5,092000E+003	5,092000E+003	0,000000E+000
0	6	5,184000E+003	5,184000E+003	0,000000E+000
1	6	5,186000E+003	5,186000E+003	0,000000E+000
2	6	5,216000E+003	5,216000E+003	0,000000E+000
3	6	5,346000E+003	5,346000E+003	0,000000E+000
4	6	5,696000E+003	5,696000E+003	0,000000E+000
5	6	6,434000E+003	6,434000E+003	0,000000E+000
6	6	7,776000E+003	7,776000E+003	0,000000E+000

Данные задачи (Не полиноминальная функция):

$$\begin{cases} u(x,y) = \sin(x+y) \\ f(x,y) = 3\sin(x+y) \\ \lambda = 1 \\ \gamma = 1 \end{cases}$$



Краевые условия:

Первого рода на всех ребрах

Содержимое json-файлов

h:

```
{
    "CountX": 5,
    "CountY": 5,
    "X"     : [0, 0.4, 0.8, 1.2, 1.6],
    "Y"     : [0, 0.4, 0.8, 1.2, 1.6],
    "GX"    : 3,
    "GY"    : 3
}
```

h/2:

```
{
    "CountX": 9,
    "CountY": 9,
    "X"     : [0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6],
    "Y"     : [0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6],
    "GX"    : 5,
    "GY"    : 5
}
```

h/4:

```
{
    "CountX": 17,
    "CountY": 17,
    "X"     : [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6],
    "Y"     : [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6],
    "GX"    : 9,
    "GY"    : 9
}
```

X	Y	U`	U(h)	U(h/2)	U(h/4)	|U` - U(h)|	|U` - U(h/2)|	|U` - U(h/4)|
0	0	0,000000E+000	0,000000E+000	0,000000E+000	0,000000E+000	0,000000E+000	0,000000E+000	0,000000E+000
0,4	0	3,894183E-001	3,894183E-001	3,894183E-001	3,894183E-001	0,000000E+000	0,000000E+000	0,000000E+000
0,8	0	7,173561E-001	7,173561E-001	7,173561E-001	7,173561E-001	0,000000E+000	0,000000E+000	0,000000E+000
0	0,4	3,894183E-001	3,894183E-001	3,894183E-001	3,894183E-001	0,000000E+000	0,000000E+000	0,000000E+000
0,4	0,4	7,173561E-001	7,186332E-001	7,176716E-001	7,174342E-001	1,277102E-003	3,155432E-004	7,814708E-005
0,8	0,4	9,320391E-001	9,320391E-001	9,320391E-001	9,320391E-001	0,000000E+000	0,000000E+000	0,000000E+000
0	0,8	7,173561E-001	7,173561E-001	7,173561E-001	7,173561E-001	0,000000E+000	0,000000E+000	0,000000E+000
0,4	0,8	9,320391E-001	9,343074E-001	9,325508E-001	9,321622E-001	2,268302E-003	5,117270E-004	1,230924E-004
0,8	0,8	9,995736E-001	1,001675E+000	9,998866E-001	9,996215E-001	2,101741E-003	3,129873E-004	4,790169E-005
1,2	0,8	9,092974E-001	9,092974E-001	9,092974E-001	9,092974E-001	0,000000E+000	0,000000E+000	0,000000E+000
1,6	0,8	6,754632E-001	6,754632E-001	6,754632E-001	6,754632E-001	0,000000E+000	0,000000E+000	0,000000E+000
0	1,2	9,320391E-001	9,320391E-001	9,320391E-001	9,320391E-001	0,000000E+000	0,000000E+000	0,000000E+000
0,4	1,2	9,995736E-001	1,001675E+000	1,000079E+000	9,996976E-001	2,101741E-003	5,056700E-004	1,239681E-004
0,8	1,2	9,092974E-001	9,115302E-001	9,098002E-001	9,094183E-001	2,232774E-003	5,027895E-004	1,208528E-004
1,2	1,2	6,754632E-001	6,766890E-001	6,757650E-001	6,755378E-001	1,225823E-003	3,017819E-004	7,463351E-005
1,6	1,2	3,349882E-001	3,349882E-001	3,349882E-001	3,349882E-001	0,000000E+000	0,000000E+000	0,000000E+000
0	1,6	9,995736E-001	9,995736E-001	9,995736E-001	9,995736E-001	0,000000E+000	0,000000E+000	0,000000E+000
0,4	1,6	9,092974E-001	9,092974E-001	9,092974E-001	9,092974E-001	0,000000E+000	0,000000E+000	0,000000E+000
0,8	1,6	6,754632E-001	6,754632E-001	6,754632E-001	6,754632E-001	0,000000E+000	0,000000E+000	0,000000E+000
1,2	1,6	3,349882E-001	3,349882E-001	3,349882E-001	3,349882E-001	0,000000E+000	0,000000E+000	0,000000E+000
1,6	1,6	-5,837414E-002	-5,837414E-002	-5,837414E-002	-5,837414E-002	0,000000E+000	0,000000E+000	0,000000E+000

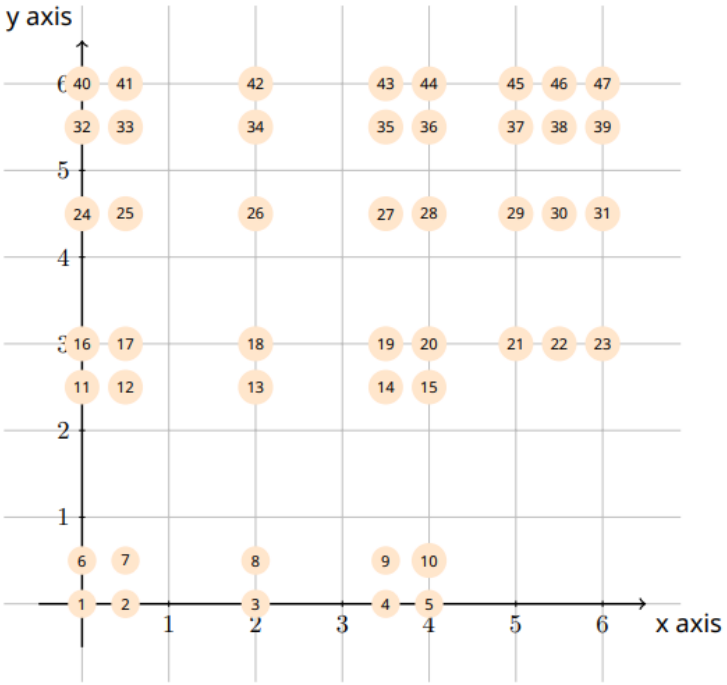
$$k = \log_2 \frac{\left\| u^* - u_h \right\|}{\left\| u^* - u_{h/2} \right\|} \approx 2{,}1916$$

$$k = \log_2 \frac{\left\| u^* - u_{h/2} \right\|}{\left\| u^* - u_{h/4} \right\|} \approx 2{,}0817$$

Данные задачи (Полином третьей степени):

$$\begin{cases} u(x,y) = 2x + 4y \\ f(x,y) = 6x + 12y \\ \lambda = 2 \\ \gamma = 3 \end{cases}$$

Краевые условия:
Первого рода на всех ребрах



Содержимое json-файла:

```
{
    "CountX": 8,
    "CountY": 7,
    "X"     : [0, 0.5, 2, 3.5, 4, 5, 5.5, 6],
    "Y"     : [0, 0.5, 2.5, 3, 4.5, 5.5, 6],
    "GX"    : 5,
    "GY"    : 4
}
```

Табличка с решением:

X	Y	U	U`	|U` - U|
0	0	0,000000E+000	0,000000E+000	0,000000E+000
0,5	0	1,000000E+000	1,000000E+000	0,000000E+000
2	0	4,000000E+000	4,000000E+000	0,000000E+000
3,5	0	7,000000E+000	7,000000E+000	0,000000E+000
4	0	8,000000E+000	8,000000E+000	0,000000E+000
0	0,5	2,000000E+000	2,000000E+000	0,000000E+000
0,5	0,5	3,000000E+000	3,000000E+000	7,105427E-015
2	0,5	6,000000E+000	6,000000E+000	1,243450E-014
3,5	0,5	9,000000E+000	9,000000E+000	3,552714E-015
4	0,5	1,000000E+001	1,000000E+001	0,000000E+000
0	2,5	1,000000E+001	1,000000E+001	0,000000E+000
0,5	2,5	1,100000E+001	1,100000E+001	2,131628E-014
2	2,5	1,400000E+001	1,400000E+001	3,552714E-014
3,5	2,5	1,700000E+001	1,700000E+001	1,776357E-014
4	2,5	1,800000E+001	1,800000E+001	0,000000E+000
0	3	1,200000E+001	1,200000E+001	0,000000E+000
0,5	3	1,300000E+001	1,300000E+001	1,598721E-014
2	3	1,600000E+001	1,600000E+001	2,664535E-014
3,5	3	1,900000E+001	1,900000E+001	2,131628E-014
4	3	2,000000E+001	2,000000E+001	7,105427E-015
5	3	2,200000E+001	2,200000E+001	0,000000E+000
5,5	3	2,300000E+001	2,300000E+001	0,000000E+000
6	3	2,400000E+001	2,400000E+001	0,000000E+000
0	4,5	1,800000E+001	1,800000E+001	0,000000E+000
0,5	4,5	1,900000E+001	1,900000E+001	7,105427E-015
2	4,5	2,200000E+001	2,200000E+001	1,421085E-014
3,5	4,5	2,500000E+001	2,500000E+001	2,131628E-014
4	4,5	2,600000E+001	2,600000E+001	1,421085E-014
5	4,5	2,800000E+001	2,800000E+001	7,105427E-015
5,5	4,5	2,900000E+001	2,900000E+001	7,105427E-015

6	4,5	3,000000E+001	3,000000E+001	0,000000E+000
0	5,5	2,200000E+001	2,200000E+001	0,000000E+000
0,5	5,5	2,300000E+001	2,300000E+001	3,552714E-015
2	5,5	2,600000E+001	2,600000E+001	3,552714E-015
3,5	5,5	2,900000E+001	2,900000E+001	1,065814E-014
4	5,5	3,000000E+001	3,000000E+001	1,065814E-014
5	5,5	3,200000E+001	3,200000E+001	7,105427E-015
5,5	5,5	3,300000E+001	3,300000E+001	0,000000E+000
6	5,5	3,400000E+001	3,400000E+001	0,000000E+000
0	6	2,400000E+001	2,400000E+001	0,000000E+000
0,5	6	2,500000E+001	2,500000E+001	0,000000E+000
2	6	2,800000E+001	2,800000E+001	0,000000E+000
3,5	6	3,100000E+001	3,100000E+001	0,000000E+000
4	6	3,200000E+001	3,200000E+001	0,000000E+000
5	6	3,400000E+001	3,400000E+001	0,000000E+000
5,5	6	3,500000E+001	3,500000E+001	0,000000E+000
6	6	3,600000E+001	3,600000E+001	0,000000E+000

Вывод:

В результате всех исследований можем сказать, что:

- Метод конечных разностей отлично справляется с уравнениями с полиномиальным решением до 3 степени включительно. На полиномах более высокого уровня решение несколько хуже.
- Порядок аппроксимации соответствует теоретическому, на равномерных сетках с краевыми условиями первого рода МКР имеет второй порядок аппроксимации.
- На неравномерных сетках может привести к понижению порядка.