摘要

本次编程作业设计了一个二维 Poission 方程（$\Delta u = f$）求解器，用来求解定义在正方形区域 Ω 和不规则区域 $\Omega \setminus \mathbb{D}$ 的 Poission 的边值问题，实现了对 Dirichlet 边界条件、Neumann 边界条件和混合边界条件三种边界条件对应的 Poission 方程求解。

1 程序实现

1.1 类和成员

1. **EquationSolver 类**首先我定义了 EquationSolver 类，用来实现求解 Poission 方程，私有成员如下

```
        class EquationSolver{
        private:
            vector<vector<double>> grids;  // denote the 2 dimension grids
            vector<double> values; //求解的数值解将储存在values中
            int N=0;  //the number of grids in 1 dimension
            double h=0.0;
            Circle *c=nullptr;
            vector<bool> incircle;
            vector<double> laplacian;
            vector<vector<double>> A; 离散拉普拉斯算子得到的矩阵
            void precondition(); //处理内部的点
            void coeffMatrix(const vector<int> & mixed); //处理规则边界
            void Diri(const Function &g); //处理不规则边界的Dirichlet边值
            void Neum(const Function &g, const vector<int> &mixed);
            //处理不规则边界的Neumann边值
            void coeffMatrix(const Function &g, const vector<int> &mixed); //处理不规则边界
            void getcolumn(const Function &g, const vector<int> &mixed);
            //处理右端项
            void solve(const Function &g, const vector<int> &mixed);//lapacke求解
            vector<double> convert(const Function &g, const vector<int> &mixed);
            void adjust(const double &initial); //对不规则区域去掉圆内的格点，对Neumann边界条件加一个常数
            vector<double> errors(const Function &f) //计算误差
            vector<double> realValues(const Function &f) //计算真实值

```

Listing 1: 运算符实现

公有成员如下，其中调用 `void solveEquation(const Function &g, const double &initial=0.0, const vector<int> &mixed=vector<int>0,0,0,0,0)` 函数将求解方程，这里的 `const Function &g` 是边值条件，`const double &initial=0.0` 是点 $(\frac{1}{h}, \frac{1}{h})$ 的值，由于 Neumann 边值问题的求解会相差一个常数，为了消除这个常数的影响，由用户自行指定希望得到的函数在 $(\frac{1}{h}, \frac{1}{h})$ 的值。`const vector<int> &mixed=vector<int>0,0,0,0,0` 给出混合边值条件，具体地：分别给直线 $y = 0$, $x = 0$, $x = 1$, $y = 1$, 编号为 0, 1, 2, 3, , 给 $\partial\mathbb{D}$ 编号为 4, 0 代表 Dirichlet 边值条件，1 代表 Neumann 边值条件。

```
        public:
            EquationSolver(const int &_N, const Function &f) //对于规则区域求解，需要输入格点数目和方程右端函数
            EquationSolver(const int &_N, const Function &f, Circle *_c):N(_N), c(_c) //对于不规则区域求解，需要输入格点数目和方程右端函数以及圆的参数
            void norm_error(const Function &f,const string &filename)//计算$l_1$ $l_2$ $l_\infty$范数下的误差，将误差输出到屏幕和文件
            void solveEquation(const Function &g, const double &initial=0.0, const vector<int> &mixed=vector<int>{0,0,0,0,0}) //求解方程
            void print(const string &filename, const Function &f) //将格点 格点上的真实值 格点上的数值解输出到文件。

```

Listing 2: 运算符实现

2. **Circle 类：**定义了 Circle 类，记录圆的圆心半径，实现了和一些简单的计算，如点到圆的距离，x 轴方向点到圆的有向距离，y 轴方向点到圆的有向距离。具体如下：

```
class Circle{
    private:
    double x0=0.0;
    double y0=0.0;
    double radius=0.0;
    public:
    Circle();
    Circle(double x, double y, double r);
    bool inCircle(double x, double y) const;

    bool onCircle(double x, double y) const;

    double get_radius() const;

    double getX() const;

    double getY()const;

    double distance(double x, double y) const;

    double x_distance_to_circle(double x, double y) const;  //有向距离

    double y_distance_to_circle(double x, double y) const; //有向距离

    double angle_x_direction(double x) const;

    double angle_y_direction(double y) const;
};


```

Listing 3: 运算符实现

1.2 测试函数

测试函数全部定义在 TestFunction.cpp 中。一共定义了 3 个测试函数。

1.3 输入和输出

由于测试用例很多，采用 jsoncpp 控制输入和输出。

- **输入：**输入文件放在 input 文件夹下，输入文件格式如下：

```
{
  "boundary_condition": "Mixed",
  "domain": "irregular",
  "grid_number": 32,
  "functionType": 3,
  "circle": [0.6,0.53,0.17],
  "mixed": [1,0,0,1,1]
}

```

这里"circle" 的参数分别是 x 坐标，y 坐标，半径。"mixed" 的参数只能是 0, 1，代表边界上的 Dirichlet 条件或者 Neumann 边界条件。

- **输出:** 最终程序运行的具体结果写在 output 文件夹下的 output.json 文件中，记录了格点、格点上数值解、格点上的函数真实值。error.json 格式如下：

```
{
  "boundary_condition": 0,
  "domain": 0,
  "l1_norm": 0.0018029833955581775,
  "l2_norm": 0.0008034791522725579,
  "linfinity_norm": 0.0005369806601569493
}
    }

```