1D Linear Finite Element Method Design

Shuang Hu

2022 年 9 月 24 日

1 问题描述

设计并实现一维线性有限元方法，求解下面的边值问题:

$$\begin{cases} -(k(x)u')' + q(x)u = f(x), \\ u(0) = \alpha, u(1) = \beta. \end{cases} \tag{1}$$

其中 $k(x) \in C^1$, $k(x) \geq k_0 > 0$, $q(x) \geq 0$。

边界条件可以修改为下面的 Neumann 条件

$$-k(0)u'(0) = \gamma_1, k(1)u'(1) = \gamma_2. \tag{2}$$

或者 Robin 条件

$$-k(0)u'(0) = -\beta_1 u(0) + \gamma_1, k(1)u'(1) = -\beta_2 u(1) + \gamma_2. \tag{3}$$

2 class RealFunc

- 表示标量函数 $f : \mathbb{R} \to \mathbb{R}$。
- **成员函数:**

1. `virtual const double operator(double x) const = 0;`

   public 成员函数

   输入: 自变量 $x \in \mathbb{R}$。

   输出: 函数值 $f(x) \in \mathbb{R}$。

   作用: 存放方程中需要存放的具体函数，在该问题中为 $k(x), q(x), f(x)$。具体函数表达式由派生类实现。

3 class MeshGrid

- 表示有限元离散网格。
- **成员变量:**

1. `bool uniform`:**private 成员变量**，表示当前离散网格是否为规则网格。
2. `int size`:**private 成员变量**，表示当前离散小区间的数量。
3. `double LeftSide`:**private 成员变量**，表示区间左端点。
4. `double RightSide`:**private 成员变量**，表示区间右端点。
5. `std::vector<double> meshgrid`:**private 成员变量**，表示网格节点 $\{x_i\}$。

- **成员函数:**

1. `MeshGrid() = default;`

   默认构造函数。
2. `MeshGrid(int _size, double Left, double Right, bool isuniform=true, const std::vector<double>& meshgrid={});`

   构造函数

   输入:`MeshGrid` 需要的所有参数。

   作用: 用具体参数初始化离散网格对象，包括规则网格情形和不规则网格情形。

4 class PiecewiseLinear

- 表示分段线性函数。
- **成员变量:**

1. `std::vector<double> nodes`:**private 成员变量**，表示分段线性函数的每个分段点。
2. `std::vector<double> value`: **private 成员变量**，表示分段函数每个分段点上的取值。

- **成员函数:**

1. `PiecewiseLinear(const std::vector<double>& _nodes, const std::vector<double>& _value);`

   构造函数
2. `PiecewiseLinear operator*(double r);`

   public 成员函数

   输入: 数乘系数 r。

   输出: 数乘后得到的分段线性函数 rf。

   作用: 实现实数和分段线性函数的数乘。
3. `friend PiecewiseLinear operator+(const PiecewiseLinear& p1, const PiecewiseLinear& p2);`

   友元函数

   输入: 分段线性函数 p_1, p_2。

   输出: 两个分段线性函数的和 $p_1 + p_2$。

   作用: 实现分段线性函数的加法。

4. `double operator()(double x):`

   public 成员函数

   输入: 自变量取值 x。

   输出: 该自变量代入分段线性函数 p 求得的函数值 $p(x)$。

   作用: 用于刻画分段线性函数的解析式。

5 class FuncBasis

- 表示分段线性函数基底。
- **成员变量:**
  1. `std::vector<double> nodes`:**private 成员变量,** 表示插值基函数节点。
  2. `std::vector<PiecewisePolynomial>`:**private 成员变量,** 存储区间上所有的 hat-function。
- **成员函数:**
  1. `FuncBasis() = default;`

     默认构造函数
  2. `FuncBasis(const std::vector<double>& _nodes;)`

     构造函数

     输入: 构造 B-样条插值基底的所有插值节点。

     作用: 利用给定的节点构造一维 B-样条插值基函数。

6 class BVP

```
enum BdryType{Dirichlet = 0, Neumann = 1, Robin = 2};
```

- 存储边值问题的相关信息，并求解之。
- **模板:**`template<BdryType BCType>`:`BCType` 表示边界条件类型。
- `friend class MeshGrid;`
- `friend class FuncBasis;`
- **成员变量:**
  1. `RealFunc* k,q,f`:**private 成员变量**，表示当前 BVP 的函数参量。
  2. `double LeftBC`: **private 成员变量**，表示当前 BVP 的左侧边值。
  3. `double RightBC`: **private 成员变量**，表示当前 BVP 的右侧边值。
  4. `MeshGrid mesh`:**private 成员变量**，表示当前离散网格。
  5. `FuncBasis basis`:**private 成员变量**，表示当前离散网格下的插值基函数。

- **成员函数:**

  1. `BVP(std::string jsonfile);`

     构造函数

     输入: 包含方程所有信息的 json 文件

     作用: 记录方程的所有参数。

  2. `template<class Func>`

     `double Numerical_Integral(double Left, double Right, Func f) const;`

     private 成员函数

     输入: 积分的左右端点，被积函数 f。

     输出: 积分的近似值。

     作用: 近似计算 $\int_{Left}^{Right} f(x)\mathrm{d}x$。

  3. `Eigen::VectorXd overload() const;`

     private 成员函数

     输出: 负载向量。

     作用: 近似计算负载向量。

  4. `void Equip(Eigen::MatrixXd& mat, int pos) const;`

     private 成员函数

     输入: 刚度矩阵 mat, 需要装配矩阵块的位置 pos。

     作用: 对刚度矩阵进行计算与装配。

  5. `PiecewisePolynomial solve() const;`

     public 成员函数

     输出: 求解得到的分段线性函数。

     作用: 对离散后的有限元问题进行数值求解。