Finite Volume Design

Shuang Hu

2022 年 7 月 19 日

1 问题描述

设计四阶精度的有限体积算法求解**对流扩散方程**的初边值问题和**不可压 Navier-Stokes 方程**的周期边界问题。方程的表示形式具体如下：

对流扩散方程：

$$\begin{cases} \dfrac{\partial \phi}{\partial t} = -\nabla \cdot (\mathbf{u}\phi) + \nu \Delta \phi + f, \\ \phi(\mathbf{x}, 0) = g_1(\mathbf{x}), \mathbf{x} \in \Omega, \\ \phi(\mathbf{x}, t) = g_2(\mathbf{x}, t), \mathbf{x} \in \partial\Omega. \end{cases} \tag{1}$$

如果是 Neumann 边界条件，第三个表达式则改为

$$\frac{\partial \phi}{\partial \mathbf{n}} = g_2(\mathbf{x}, t). \tag{2}$$

周期边界的 INSE, 区域 $\Omega := [0, 1]^2$:

$$\begin{cases} \dfrac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \mathbf{g} - \nabla p + \nu \Delta \mathbf{u}, \\ \nabla \cdot \mathbf{u} = 0, \mathbf{x} \in \Omega, \\ \mathbf{u}(x, y, t) = \mathbf{u}(x + 1, y, t). \\ \mathbf{u}(x, y, 0) = \mathbf{u}_0(x, y) \end{cases} \tag{3}$$

在本次作业中，需要实现对流扩散方程在 **Dirichlet** 和 **Neumann** 两种边界条件下的求解，其中时间积分方法利用 **ERK-ESDIRK IMEX Runge-Kutta 格式**，并且在近似 **Leray-Helmholtz 投影算子**时，需要采用多重网格算法。

2 底层程序

底层的数据结构和数值算法沿用先前组里求解 Navier-Stokes 方程的软件包，在本程序中需要用到的是以下内容：

- `class Vec`: 用来表示空间中的点。
- `class Tensor`: 用来存储体平均值，表示系数矩阵等。

- `class RowSparse`: 用于存储稀疏矩阵。
- `class Box`: 用于网格离散。
- `class RectDomain`: 用于表示问题区域（矩形）。
- `numlib.h`: 一些常用的数值算法，这里会多次用到数值积分程序段。
- 约定一些符号表示：
  - `template<int Dim>` 表示问题区域的维数。
  - `using rVec=Vec<Real,Dim>`

3 class VectorFunction

- 函数 $\mathbb{R}^{\mathrm{Dim}_1} \to \mathbb{R}^{\mathrm{Dim}_2}$ 的基类，用于表示方程的初值/边值信息，或者是右端项。注意这里的 `Dim2` 不等于 1，`Dim2=1` 时我们用 `ScalarFunction` 存储。
- **模板:** `template<int Dim1,int Dim2>`:

  `Dim1` 和 `Dim2` 分别表示定义域和值域所在的空间维数。
- **成员函数:**

  1. `virtual const Vec<Dim2> operator()(const Vec<Dim1>& pt) const = 0;`

     public 成员函数

     输入: `pt` 表示 `Dim1` 维空间中的一个点。

     输出: 该函数在 `pt` 点处的取值。

     作用: 计算函数在一个离散点上的值。纯虚函数，需要在继承类中具体实现。

4 class ScalarFunction

- 函数 $\mathbb{R}^{\mathrm{Dim}} \to \mathbb{R}$ 的基类。
- **模板:**`template<int Dim>`: `Dim` 表示定义域的维数。
- **成员函数:**

  1. `virtual const Real operator()(const Vec<Dim>& pt) const = 0;`

     public 成员函数

     输入: `pt` 表示 `Dim` 维空间中的一个点。

     输出: 该函数在 `pt` 点处的取值。

     作用: 计算函数在一个离散点上的值。纯虚函数，需要在继承类中具体实现。

5 class FuncFiller

- 将所给函数的积分平均值填充到离散网格中。
- **模板:**`template<int Dim>`:

  `Dim` 表示问题空间的维数。在本次作业中，该模板参数取 2，下同。
- **成员变量:**

  1. `RectDomain<Dim> domain`: **private 成员变量**，表示需要填充的均匀网格。
- **成员函数:**

  1. `FuncFiller(const RectDomain<Dim>& adomain);`

     public 成员函数

     输入: 需要进行函数值填充的问题区域，要求为矩形区域。

     作用: 构造函数，记录问题区域的信息。
  2. `template<int Dim2>`

     `Vec<Real,Dim2> Quadrature(const rVec& lo,const rVec& hi,`
     `const VectorFunction<Dim,Dim2>* func) const;`

     private 成员函数

     输入: `lo` 表示正方体区域左下角，`hi` 表示区域右上角，`func` 表示需要求积分的函数。

     输出: 函数 `func` 在网格 `[lo,hi]` 上积分平均的近似值。

     作用: 计算向量值函数在一个网格上积分平均的近似值，即体平均值。
  3. `Real Quadrature(const rVec& lo,const rVec& hi,`
     `const ScalarFunction<Dim>* func) const;`

     private 成员函数

     输入: `lo` 表示正方体区域左下角，`hi` 表示区域右上角，`func` 表示需要求积分的函数。

     输出: 函数 `func` 在网格 `[lo,hi]` 上积分平均的近似值。

     作用: 计算标量函数在一个网格上积分平均的近似值，即体平均值。
  4. `template<int Dim2>`

     `void fill(Tensor<Vec<Real,Dim2>,Dim>& target, const VectorFunction<Dim,Dim2>*`
     `func) const;`

     public 成员函数

     输入: `target` 为待填充的 `Dim` 维 `Tensor`; `func` 指向一个向量值函数。

     作用: 将函数 `*func` 在每个离散网格上的积分平均值填入 `target`。
  5. `void fill(Tensor<Real,Dim>& target, const ScalarFunction<Dim>* func) const;`

     public 成员函数

     输入: `target` 为待填充的 `Dim` 维 `Tensor`; `func` 指向一个标量函数。

     作用: 将函数 `*func` 在每个离散网格上的积分平均值填入 `target`。

6 class GhostFiller

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

- 利用已知的边界条件填充 Ghost Cell。
- **模板:** template<int Dim,BdryType BCType>:

  Dim 表示空间维数。BCType 表示边界条件种类。

  BCType 表示边值条件的种类是 Dirichlet 或 Neumann。
- **成员变量:**

  1. RectDomain<Dim> domain: **private 成员变量**，表示需要填充 Ghost Cell 的区域。
  2. Tensor<Real,Dim> bdryCond: **private 成员变量**，表示边界处的面平均信息。
  3. int nGhost: **private 成员变量**，表示需要填充的 Ghost Cell 层数。
- **成员函数:**

  1. Real FaceAverage(const rVec& lo,const rVec& hi,
     const ScalarFunction<Dim>* func) const;

     private 成员函数

     输入: lo 表示控制面的左下角，hi 表示控制面的右上角，func 指向需要求积分的函数。

     输出: 函数 func 在控制面 [lo,hi] 上的积分平均值。

     作用: 计算函数在控制面上的积分平均。
  2. GhostFiller(const RectDomain<Dim>& aDomain, int numGhost);

     public 成员函数

     输入: aDomain 表示需要进行 Ghost Cell 填充的区域，atype 表示边界条件的种类，
     numGhost 表示需要填写 Ghost Cell 的层数。

     作用: 构造函数，记录区域信息，边界条件信息，并初始化 bdryCond。
  3. void fillBdry(const ScalarFunction<Dim>* func);

     public 成员函数

     输入: func 表示边值条件。

     作用: 利用边值条件计算区域边界控制面的平均值，并填充至 bdryCond。
  4. void fillGhost(Tensor<Real,Dim>& res) const;

     输入: res 表示需要填充 Ghost Cell 张量。

     作用: 利用 res 中原有的体平均值和计算出的面平均值，向外填充 Ghost Cell，更新 res 的值。

7 class SpacialOp

- 虚基类，用于表示离散的空间算子。
- **模板:**`template<int Dim,class T1,class T2>`:

  `Dim` 表示空间维数，`T1` 表示算子作用前的数据类型，`T2` 表示算子作用后的数据类型。
- **成员变量:**

  1. `RectDomain<Dim> domain`: **protected 成员变量**，表示需要填充 `Ghost Cell` 的区域。
  2. `int nGhost`: **protected 成员变量**，表示需要填充 `Ghost Cell` 的层数。
- **成员函数:**

  1. `SpacialOp(const RectDomain<int>& aDomain,int numGhost);`

     public 成员函数

     输入: `aDomain` 表示问题区域，`numGhost` 表示需要填充 `Ghost Cell` 的层数。

     作用: 构造函数，记录空间算子所需的信息。
  2. `void getFaceAvg(const Tensor<T1,Dim>& origin, Tensor<T1,Dim>& res, int dim);`

     public 成员函数

     输入:`origin` 表示**填写过 Ghost Cell 的**控制体平均值张量，`res` 表示控制面平均张量的近似值，`dim` 表示在第 `dim` 维度上求解面平均。

     作用: 利用体平均来近似计算面平均。
  3. `virtual void apply(const Tensor<T1,Dim>& origin, Tensor<T2,Dim>& res) = 0;`

     public 成员函数

     输入:`origin` 表示控制体平均张量, `res` 表示在 `origin` 上作用离散算子之后得到的控制体平均值张量。

     作用: 用于描述空间离散算子的作用结果。

8 class Laplacian

- 用于表示拉普拉斯算子的四阶有限体积离散。
- **模板:** `template<int Dim>`
- **继承:** `class Laplacian:public SpacialOp<Dim,Real,Real>`
- **成员函数:**

  1. `void apply(const Tensor<Real,Dim>& origin, Tensor<Real,Dim>& res);`

     public 成员函数

输入:origin 表示控制体上的体平均张量，res 表示作用拉普拉斯算子后控制体上的体平均张量。

作用: 对拉普拉斯算子进行离散化。

9 class Divergent

- 用于表示散度算子的四阶有限体积离散
- **模板:** template<int Dim>
- using rVec=Vec<Real,Dim>;
- **继承:** class Divergent:public SpacialOp<Dim,rVec,Real>
- **成员函数:**

1. void apply(const Tensor<rVec,Dim>& origin, Tensor<Real,Dim>& res);

   public 成员函数

   输入:origin 表示一个向量场的体平均值，res 表示该向量场上作用离散散度算子后的体平均值。

   作用: 对散度算子进行离散化。

10 class Gradient

- 用于表示梯度算子的四阶有限体积离散
- **模板:**template<int Dim>
- using rVec=Vec<Real,Dim>
- **继承:**class Gradient:public SpacialOp<Dim,Real,rVec>
- **成员函数:**

1. void apply(const Tensor<Real,Dim>& origin, Tensor<rVec,Dim>& res);

   public 成员函数

   输入:origin 表示一个标量场的体平均值，res 表示在该标量场上作用离散梯度算子后得到的体平均值。

   作用: 对梯度算子进行离散化。

11 class InnerProduct

- 用于求解内积算子面平均值的离散化。
- **模板:** `template<int Dim>`
- **继承:** `class InnerProduct:public SpacialOp<Dim,Real,Real>`
- **成员变量:**
  1. `int dim`: **private 成员变量,** 用于表示当前我们所关心的控制面。
  2. `Tensor<Real,Dim> phi`: **private 成员变量,** 表示内积表达式中的其中一个函数。
- **成员函数:**
  1. `InnerProduct(const RectDomain<Dim>& aDomain, int numGhost,const Tensor<Real,Dim>& aphi, int adim);`
     public 成员函数
     输入:`aDomain,numGhost` 用于初始化基类，`adim,aphi` 用于初始化本派生类。
     作用: 构造函数。
  2. `void apply(const Tensor<Real,Dim>& origin, Tensor<Real,Dim>& res);`
     public 成员函数
     输入:`origin` 表示一个标量场在 `dim` 维上的**面平均值**，`res` 表示在该控制面上 `origin` 和 `phi` 作离散内积后的面平均值。

12 class Advection

- 用于实现对流项的离散化。
- **模板:**`template<int Dim>`
- **继承:**`class Advection:public SpacialOp<Dim,Real,Real>`
- TBD

13 class Projection

- TBD

14 class RungeKutta

- 进行一步 Runge-Kutta 迭代，求解初值问题

$$\begin{cases} \dfrac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = f(\mathbf{x}, t), \\ \mathbf{x}(t_0) = \mathbf{x}_0. \end{cases} \tag{4}$$

- **模板:** `template<int Dim,class DType>`: DType 表示初值问题变量的数据类型。
- **成员变量:**
  1. `Tensor<Real,Dim> butlar`: **private 成员变量**, 记录 Runge-Kutta 法对应的 Butlar 表。
  2. `RectDomain<Dim> domain`: **private 成员变量**, 记录区域信息。
- **成员函数:**
  1. `virtual void getRHS(const Tensor<DType,Dim>& input, Tensor<DType,Dim>& rhs, Real t) = 0;`

     public 成员函数

     输入:`input` 表示目前的体平均张量值，`rhs` 表示在 t 时刻的右端项。

     作用: 用于描述 $f(\mathbf{x},t)$，纯虚函数，依赖于派生类的具体实现，与算子离散方式有关。
  2. `void apply(Tensor<DType,Dim>& res, const Tensor<DType,Dim>& input, Real t0, Real k);`

     public 成员函数

     输入: `input` 表示初始值 $\mathbf{x}(t_0)$，`rhs` 表示近似值 $\mathbf{x}(t_0+k)$, `t0` 表示初始时刻, `k` 表示时间间隔。

     作用: 在已知 $\mathbf{x}(t_0)=\mathbf{x}_0$ 的情况下，利用 `Runge-Kutta` 算法求解 $\mathbf{x}(t_0+k)$。

15 class Advection$_D$iffusion