Chapter 1

EmbeddedGrid 生成流程

1.1 c++ 类设计

1.1.1 Curve

数据成员

1. `vector<Polynomial> polys;`
   分段拟合的多项式样条.

2. `vector<Real> knots;`
   拟合样条的 knots 顺序集合.

1.1.2 OrientedJordanCurve

继承自 Curve, 拥有同样的数据结构, 额外满足首尾相连.

函数成员

1. `void define(vector<Vec>& points, vector<size_t>& kinks);`
   根据输入的点 points 和尖点 kinks 集合, 拟合初始化带尖点的 OrientedJordanCurve.

2. `void split(const vector<Real> &brks,`
   `vector<Curve> &out, Real tol) const;`
   输入断点 brks, 得到 OrientedJordanCurve 从断点剪开的 Curve 集合 out.

1.1.3 YinSet

数据成员

1. `vector<OrientedJordanCurve> orientedJordanCurves`
   表示 YinSet 边界的 JordanCurve 集合.

2. `SimplicialComplex kinks`
   YinSet 边界上的尖点.

函数成员

1. 构造函数通过输入流, 逐个构造 OrientedJordanCurve 并记录尖点 kinks.

2. `void splitWithKinks(vector<Curve> &out) const;`
   输出沿所有尖点剪开 orientedJordanCurves 得到的 Curve 集合.

1.1.4 BoundaryCondition

1. `static const string value[3] = {"Periodic", "Neumann", "Drichlet"};`
   支持的各种边界条件.

2. `size_t type;`
   当前类表示的边界条件.

1.1.5 EmbeddedGrid

内部类 LabelBit

1. bdry = 0x01 标记边界 cell/face.

2. inter = 0x02 标记内部 cell/face.

3. disable = 0x04.

4. master = 0x08

5. slave = 0x10

6. b_master = 0x20

7. b_slave = 0x40

8. rbdry = 0x80 标记规则边界 cell/face.

9. irrbdry = 0x100 标记不规则边界 cell/face.

10. Periodic = 0x200 标记 Periodic 边界条件的边界 cell/face.

11. Neumann = 0x400 标记 Neumann 边界条件的边界 cell/face.

12. Drichlet = 0x800 标记 Drichlet 边界条件的边界 cell/face.

数据成员

1. `public: RectDomain<Dim>`
   覆盖计算区域的网格

2. `Tensor<vector<Crv>,Dim> bdryRepo[2*Dim + 1];`
   存储每个 cell 内的边界,0-2*Dim-1 对应 face 上的边界最后一项是 cell 内部的边界.

3. `vector<iVec> difficultFaces[Dim], difficultCells, difficultBdries;`

4. `Real fullVol, tinyFrac, tol;`
   fullVol 是单个 cell 的体积, tinyFrac 表示不被合并的最小体积, tol 是计算容忍度.

5. `Tensor<int,Dim> cellLabel;`
   记录每个 cell 的类型 bdry/inter, 如果是边界 cell 继续区分规则/不规则边界和各种边界条件.

6. `Tensor<LinkedListNode,Dim> cellMergeInfo;`

7. `Tensor<Real,Dim> cellVol;`
   记录每个 cell 在计算区域内部的体积.

8. `Tensor<Crv,Dim> cellBdry;`
   记录每个 cell 中的计算区域的边界 (只能有一个连通区域, 否则报错).

9. `Tensor<int,Dim> faceLabel[Dim];`
   记录每个 face 的类型, 区分规则边界和不规则边界和各种边界条件.

10. `Tensor<LinkedListNode,Dim> faceMergeInfo[Dim];`

11. `Tensor<Real,Dim> faceVol[Dim];`
    记录每个 face 在计算区域内的体积.

12. `Tensor<Crv,Dim> faceBdry[Dim];`
    记录每个 face 上的边界.

13. `Tensor<Real,Dim> irFaceVol;`
    每个 cell 中不规则边界的体积.

14. `Tensor<Crv,Dim> irFaceBdry;`
    每个 cell 中的不规则边界.

15. `Tensor<LinkedListNode, Dim> irMergeInfo;`

16. `vector<iVec> bdryCellIdx;`
    包含边界的 cell 集合.

17. `vector<iVec> bdryFaceIdx[Dim];`
    包含边界的 face 集合.

18. `map<iVec, int, iVCMP> bdryCellIdxInv;`
    bdryCellIdx 的逆运算.

19. `Tensor<int, Dim> nodeLabel;`
    记录网格中在计算区域内部的网格点.

函数成员

1. `EmbeddedGrid(const RectDomain<Dim> &aDomain, const YinSet<Dim, Order> &y, const vector<BoundaryCondition> &bc, Real aTinyFrac, Real aTol)`
   初始化函数
   - aDomain 是覆盖计算区域的网格.
   - y 是计算区域 YinSet.
   - bc 是 YinSet 边界上的边界条件.(满足与 y.split 输出的 Curve 集合一一对应.)
   - aTinyFrac 是不需要合并的 cell 的体积下限.
   - aTol 是计算容忍度.

1.2 EmbeddedGrid 初始化过程

1.2.1 构造 OrientedJordanCurve

调用 define 输入拟合的 `points` 和尖点 `kinks` 进行初始化.

1.2.2 构造 YinSet

输入已经初始化的 OrientedJordanCurve 集合和尖点的下标集合构造函数初始化 `YinSet y`.

1.2.3 构造 BoundaryCondition

根据 YinSet::split 输出的 Curve 集合上的边界条件, 构造 `vector<BoundaryCondition> bc`.

1.2.4 构造 EmbeddedGrid

`YinSet y` 生成恰当的计算网格 `aDomain`, 设定 `aTinyFrac, aTol`. 调用构造函数.