MARS2D & 几何流程序设计文档

下文中 Vector 表示 std::vector，Point 表示 Vec<Real, Dim>。根据测试结果，我们的两种容器备选方案是 Vector 和 IMV，所以不再需要 Container 模板，统一用 Vector 来存储点列。为了实现两种不同速度场的行为，在 TimeIntegrator 类中添加了一个模板参数 template <int> class VectorRHS 来作为速度场的 policy 类，并通过特例化来实现不同的行为。

1 class VectorFunction

- 函数 $\mathbb{R}^{\mathtt{Dim}} \times \mathbb{R} \to \mathbb{R}^{\mathtt{Dim}}$ 的基类，可以作为速度场的基类使用。
- **模板：** template<int Dim>：
  Dim 表示空间维数。
- **成员函数：**

  (1) virtual const Point operator()(const Point &pt, Real t) const=0
  Private 成员函数
  输入： pt 表示 Dim 维空间中的一个点，t 表示当前时间。
  输出： pt 点处对应的速度场。
  作用： 作为一个 helper function 使用。当速度场是给定的时，每个点对应的速度场函数都是相同的，所以没有必要去重载点列版本的速度场，只需要重载一个单点版本就可以满足。

  (2) virtual const Vector<Point> operator()(const Vector<Point> &pts, Real t=0) const;
  输入： pts 表示一列 Point，t 为当前时间。
  输出： 点列中每个点对应的速度场。
  作用： 提供一个默认的调用单点 operator()() 实现的版本，便于给定速度场的使用。

2 class VectorOnHypersurface

- 描述嵌入在 Dim 维空间中余维数是 1 的超曲面上的离散点列处的速度场，且在每个离散点处的速度场是依赖于其他离散点的。

- **模板：** template<int Dim>:
  Dim 表示空间维数。

- 继承:~~class DiscreteVecCoDimOne: public VectorFunction<Dim>~~。

- **成员函数：**

  (1) virtual const Vector<Point> operator()(const Vector<Point> &pts, Real t=0) const=0;
  输入： pts 表示一列 Point，t 为当前时间。
  输出： 点列中每个点对应的速度场。
  作用： 求点列对应的速度场，需要在其继承类中实现。

  (2) virtual const Tensor<Real, 2> getJacobi(const Vector<Point> &pts, Real t=0) const;
  输入： pts 表示一列 Point，t 为当前时间。
  输出： 用 Tensor<Real, 2> 表示的雅克比矩阵。
  作用： 提供一个什么都不做的默认版本，因为可能有些情况下是不需要雅克比矩阵的。在其继承类中，若需要使用隐式方法进行界面追踪问题的求解，则必须重载这一函数。暂时先调用 LAPACKE 来进行隐式方法的求解。先得到一个正确的结果之后再寻找高效的求解分块循环多对角线性方程组的算法。

3 class VectorForCurvatureFlow

- 曲率流的速度场类。

- **模板：** template<int Dim, int Order>:
  Dim 表示空间维数，Order 表示空间离散的阶数。

- **继承：** class VectorForCurvatureFlow: public VectorOnHypersurface<Dim>。

- **成员函数：**

(1) `const Vector<Point> operator()(const Vector<Point> &pts, Real t=0) const;`
输入输出： 同基类一致。
作用： 根据曲率流对应阶数的空间离散方法，计算出示踪点列速度场的近似值。

(2) `const Tensor<Real, 2> getJacobi(const Vector<Point> &pts, Real t=0) const;`
输入输出： 同基类一致。
作用： 根据曲率流对应阶数的空间离散方法，计算出示踪点列速度场对应的雅克比矩阵。

4 class VectorForSurfaceDiffusionFlow

- 表面扩散流的速度场类。
- **模板：** `template<int Dim, int Order>`:
  `Dim` 表示空间维数，`Order` 表示空间离散的阶数。
- **继承：**`class VectorForSurfaceDiffusionFlow: public VectorOnHypersurface<Dim>`。
- **成员函数：**

(1) `const Vector<Point> operator()(const Vector<Point> &pts, Real t=0) const;`
输入输出： 同基类一致。
作用： 根据表面扩散流对应阶数的空间离散方法，计算出示踪点列速度场的近似值。

(2) `const Tensor<Real, 2> getJacobi(const Vector<Point> &pts, Real t=0) const;`
输入输出： 同基类一致。
作用： 根据表面扩散流对应阶数的空间离散方法，计算出示踪点列速度场对应的雅克比矩阵。

5 class TimeIntegrator

- 时间积分方法的基类。
- **模板：** `template<int Dim, template <int> class VectorRHS>`:
  `Dim` 表示空间维数 , `VectorRHS<Dim>` 表示选择的速度场类型。
- **成员函数：**

(1) `virtual const Point timeStep(const VectorRHS<Dim> &v, const Point &pt, Real tn, Real k)=0`:
输入： `v` 为速度场，`pt` 表示当前点，`tn` 为当前时间，`k` 为时间步长。
输出： `pt` 在速度场 `v` 作用 `k` 时间后得到的新点。
作用： 函数使得 `pt` 在速度场 `v` 的作用下运动 `k` 时间。需要在继承类中进行实现。这里保留单点版本的接口是因为在 MARS2D 中，splitLongEdges 函数在速度场为 VectorFunction 的情形下的实现会用到时间积分方法的单点版本，把要加入的点先流映射再插入，可以进一步降低时间复杂度。

(2) `virtual void timeStep(const VectorRHS<Dim> &v, Vector<Point> &pts, Real tn, Real k)=0`:
输入： `v` 为速度场，`pts` 表示一列 `Point`，`tn` 为当前时间，`k` 为时间步长。
输出： `void`，原址更改 `pts`。
作用： 函数使得 `pts` 中的一列点在速度场 `v` 的作用下运动 `k` 时间。需要在继承类中进行实现。

6 ButcherTableau

6.1 namespace RK

定义名字空间，不然 `enum` 中的 `ERK` 等会和 `class ERK` 等冲突。

6.1.1 enum Type_Major

- `enum Type_Major{ERK=0, DIRK, ARK, nRK_Major}`。
- **作用：** 表示 RK 方法的一般类型。

6.1.2 enum Type_Minor

- `enum Type_Minor{ForwardEuler=0, ClassicRK4, SDIRK2, ESDIRK4, nRK_Minor}`。
- **作用：** 表示 RK 方法的细分类型。

6.2 struct ButcherTableau

- **模板：** `template<RK::Type_Major Type1, RK::Type_Minor Type2>`:
`Type1` 表示 RK 的一般类型，如 ERK, DIRK, ARK 等；`Type2` 表示 RK 的细分类型，

如 Type1 是 ERK 时，Type2 可以是 ForwardEuler，ClassicRK4 等；Type2 是 DIRK 时，Type2 可以是 SDIRK2，ESDIRK4 等。对于给定的 RK::Type_Major 中的一般类型，其数据结构是固定的。

- **特例化：**

(1)
```
template <>
struct ButcherTableau<ERK, ForwardEuler>
{
  static constexpr int order = 1;
  static constexpr int nStages = 1;
  static constexpr Real a[nStages][nStages] = ...;
  static constexpr Real b[nStages] = ...;
  static constexpr Real c[nStages] = ...;
};
```

(2)
```
template <>
struct ButcherTableau<ERK, ClassicRK4>
{...};
```

(3)
```
template <>
struct ButcherTableau<DIRK, SDIRK2>
{...};
```

(4)
```
template <>
struct ButcherTableau<DIRK, ESDIRK4>
{...};
```

7 class ERK

- 继承自 TimeIntegrator<Dim, VectorRHS>，用于实现所有 ERK 方法。
- **模板：**template<int Dim, template <int> class VectorRHS, RK::Type_Minor Type>：Dim 表示空间维数，VectorRHS<Dim> 表示选择的速度场类型，Type 是 ERK 方法的某个子方法。这里可以这么做是因为所有 ERK 方法的数据结构都是一致的，所以在 class ERK 中，只需要知道所用的是哪种子方法就可以建立对应的 Butcher 表，进而实现对应的 ERK 方法。

- `using ButcherTab = ButcherTableau<RK::ERK, Type>;`
  `const int order = ButcherTab::order;`

- **成员函数：**

  (1) `const Point timeStep(const VectorRHS<Dim> &v, const Point &pt, Real tn, Real k)`：
  输入输出： 同基类一致。
  作用： 调用 ButcherTab 中的成员常量以及 v 的单点版本速度场实现对应的 ERK 方法，进一步实现基类中的纯虚函数。只会在 VectorRHS 为 VectorFunction 时使用，若在 VectorRHS 为 VectorOnHypersurface 时使用则会抛出异常。

  (2) `void timeStep(const VectorRHS<Dim> &v, Vector<Point> &pts, Real tn, Real k)`：
  输入输出： 同基类一致。
  作用： 调用 ButcherTab 中的成员常量以及 v 的点列版本速度场实现对应的 ERK 方法，进一步实现基类中的纯虚函数。

8 class DIRK

- 继承自 `TimeIntegrator<Dim, VectorRHS>`，用于实现所有 DIRK 方法。

- **模板：**`template<int Dim, template <int> class VectorRHS, RK::Type_Minor Type>`：Dim 表示空间维数，VectorRHS<Dim> 表示选择的速度场类型，Type 是 DIRK 方法的某个子方法。这里可以这么做是因为所有 DIRK 方法的数据结构都是一致的，系数矩阵 a 都是下三角的，所以在 class DIRK 中，只需要知道所用的是哪种子方法就可以建立对应的 Butcher 表，进而实现对应的 DIRK 方法。

- `using ButcherTab = ButcherTableau<RK::DIRK, Type>;`
  `const int order = ButcherTab::order;`

- **成员函数：**

  (1) `const Point timeStep(const VectorRHS<Dim> &v, const Point &pt, Real tn, Real k)`：
  输入输出： 同基类一致。
  作用： 调用 ButcherTab 中的成员常量以及 v 的单点版本速度场实现对应的 DIRK

方法，进一步实现基类中的纯虚函数。只会在 VectorRHS 为 VectorFunction 时使用，若在 VectorRHS 为 VectorOnHypersurface 时使用则会抛出异常。

(2) `void timeStep(const VectorRHS<Dim> &v, Vector<Point> &pts, Real tn, Real k)`:
输入输出： 同基类一致。
作用： 调用 ButcherTab 中的成员常量以及 v 的点列版本速度场和 getJaocbi 函数来实现对应的 DIRK 方法，进一步实现基类中的纯虚函数。过程中需要使用牛顿迭代求解非线性方程组。

9 class MARS

- 殷集界面追踪方法的基类。
- **模板：** `template<int Dim, int Order, template<int> class VelocityField>`:
其中 Dim 表示维数，Order 表示殷集边界表示的阶数，VelocityField<Dim> 表示选择的速度场类型。
- **成员变量：**

(1) `TimeIntegrator<Dim, VelocityField<Dim> > *TI`: **Protected 成员变量**，时间积分方法基类指针。

- **成员函数：**

(1) `MARS(TimeIntegrator<Dim, VelocityField<Dim> > *_TI):TI(_TI){}`:
构造函数。

(2) `virtual void timeStep(const VelocityField<Dim> &v, YinSet<Dim, Order> &ys, Real tn, Real k) = 0`:
输入： v 为速度场，ys 为殷集，tn 和 k 的定义同时间积分方法中一致。
输出： void，原址更改 ys。
作用： 纯虚函数，作为二维和三维 MARS 方法 timeStep 的公共接口，在继承类中进行实现。函数将 tn 时刻的殷集映射到 k 时间后的殷集并赋值给 ys。

(3) `void trackInterface(const VelocityField<Dim> &v, YinSet<Dim,Order> &ys, Real startTime, Real k, Real endTime)`:
输入： v 为速度场，ys 为殷集，startTime 和 endTime 表示 MARS 方法作用的起止时间，k 为时间步长。

输出： void，原址更改 ys。
作用： 在速度场 v 的作用下，将 startTime 时刻的殷集 ys 通过时间步长为 k 的 MARS 方法映射到 endTime 时刻的殷集并赋值给 ys。调用 timeStep 在此基类中进行实现。

10 class MARS2D

- 二维殷集的界面追踪方法。
- **模板：** template<int Order, template <int> class VelocityField>:
  其中 Order 表示二维殷集边界所用样条曲线的阶数，VelocityField<2> 表示选择的速度场类型。
- **继承：** class MARS2D: public MARS<2, Order, VelocityField>。
  using Base = MARS<2, Order, VelocityField>。
- **成员变量：**

  (1) Interval<1> chdLenRange：殷集边界上相邻节点间弦长取值范围。

- **成员函数：**

  (1) MARS2D(TimeIntegrator<2, VelocityField<2> > *_TI, Real hL, Real rtiny=0.1): Base(_TI){...}:
  构造函数，使 chdLenRange 为 [rtiny*hL,hL]，rtiny 默认值为 0.1。

  (2) void discreteFlowMap(const VelocityField<2> &v, Vector<Point> &pts, Real tn, Real k):
  Private 成员函数
  输入： v 为速度场，pts 为示踪点列，tn 和 k 的定义同时间积分方法中的一致。
  输出： void，原址更改 pts。
  作用： 函数调用 Base::TI->timeStep 将 pts 映射到 k 时间后的示踪点列。

  (3) Vector<unsigned int> splitLongEdges(const VelocityField<2> &v, Vector<Point> &pts, const Curve<2, Order> &crvtn, Real tn, Real k):
  Private 成员函数
  输入： v 为速度场，pts 为下一时刻的示踪点列，crvtn 为当前时刻对应的样条曲线，tn 和 k 的定义同时间积分方法中的一致。
  输出： pts 中新加入点的序号。

作用： 函数首先从 crvtn 中提取出当前时刻对应的示踪点列 oldpts，之后在 pts 中寻找相邻点间距离过长的弦，并在 crvtn 中对应的曲线段上加点得到加点后 oldpts，利用 Base::TI->timeStep 将新加入的点映射到下一时刻得到新的 pts。

(4) `Vector<unsigned int> removeSmallEdges(Vector<Point> &pts)`：

Private 成员函数

输入： pts 为下一时刻的示踪点列。

输出： 删除的节点在原 pts 中的序号。

作用： 函数将 pts 中相邻点间弦长过小的点在原址进行删除，满足不删除首尾两点、不连续删点的条件。

(5) `void timeStep(const VelocityField<2> &v, YinSet<2,Order> &ys, Real tn, Real k)`：

输入： 同基类一致。

输出： 同基类一致。

作用： 实现基类中的纯虚函数，函数依次调用 discreteFlowMap, splitLongEdges, removeSmallEdges 和 fitCurve 实现二维 MARS 方法中的一个时间步，即预处理、流映射、后处理的复合，将当前时刻的殷集映射到 k 时间后的殷集并赋值给 ys。