

# Curve Factory 程序设计文档

## 1 Curve(已有)

- 表示由分片多项式组成的曲线.
- **模板:** `template<int Dim, int Order>`
- 模板中 `Dim` 表示维数, `Order` 表示多项式阶数.

## 2 OrientedJordanCurve

- 表示由分片多项式组成的有向若当曲线, 继承自 `Curve<Dim, Order>`
- **模板:** `template<int Dim, int Order>`
- **Public 成员函数:**

(1) `OrientedJordanCurve() = default;`

默认构造函数

(2) `virtual void define(const std::string& parameters);`

**输入:** 一行字符串用于存储相关参数.

**前置条件:** `parameters` 需能依次读入以下信息.

- `unsigned int nKnots`: 结点数, 读入一个 `unsigned int` 得到.
- `std::vector<Vec<Real, Dim>> points`: 结点坐标, 读入 `nKnots*Dim` 个 `Real` 得到.
- `SimplicialComplex kinks`: 尖点信息, 读入剩于 `unsigned int` 得到.

**作用:** 使用 `sstream` 中相关操作将 `parameters` 转化成输入流, 依顺序读入参数, 得到 `points` 和 `kinks`. 再通过 `protected` 成员函数 `define` 构造曲线.

- **Protected 成员函数:**

(1) `void define(const std::vector<Vec<Real, Dim>>& points, const SimplicialComplex& kinks)`

**输入:**

- `points` 是一列点的空间坐标, 作为样条的结点.
- `kinks` 存储了曲线中不光滑结点编号.

**前置条件:**

- `points` 中点存储顺序需与曲线绕向相同.

- `points` 长度至少为四, 且首尾点需相同.
- `kinks` 存储的是 `knots` 中对应结点编号.

**作用:**

根据 `kinks` 中尖点信息将 `points` 中点分段. 如果没有尖点, 则使用周期性边界条件进行曲线拟合, 否则先使用自然边界条件对每片曲线进行拟合再拼接成所要的曲线.

### 3 Circle

- 表示由分片多项式拟合的有向圆形曲线. 继承自 `OrientedJordanCurve<2, Order>`.
- **模板:** `template <int Order>`
- **Public 成员函数:**

(1) `Circle() = default;`

默认构造函数.

(2) `void define(const std::string& parameters);`

**输入:** 一行字符串用于存储相关参数.

**前置条件:** `parameters` 需能依次读入下列参数:

- `Vec<Real, 2> center`: 圆心. 读入两个 `Real` 确定.
- `Real radius`: 半径. 读入一个 `Real` 确定. 需满足 `radius > 0`.
- `bool orientation`: 定向. 读入一个 `bool` 确定, 正定向存为 `true`.
- `Real hL`: 相邻结点间最大距离. 读入一个 `Real` 确定. 需满足 `hL > 0`.

**作用:** 使用 `sstream` 中相关操作将 `parameters` 转化成输入流, 依次读入参数. 将圆均匀分割并确定结点, 相邻结点距离不超过 `hL`. 所有结点均设为光滑结点, 调用 `OrientedJordanCurve` 中的 `protected` 成员函数 `define` 构造出圆.

### 4 Rectangle

- 表示由分片多项式拟合的有向方形曲线. 继承自 `OrientedJordanCurve<2, Order>`.
- **模板:** `template<int Order>`
- **Public 成员函数:**

(1) `Rectangle() = default;`

默认构造函数.

(2) `void define(const std::string& parameters);`

**输入:** 一行字符串用于存储相关参数.

**前置条件:** `parameters` 需能依次读入下列参数:

- `Vec<Real, 2> smallEnd`: 方形在旋转之前左下顶点坐标. 读入两个 `Real` 确定.

- `Vec<Real, 2> bigEnd`: 方形在旋转之前右上顶点坐标. 读入两个 `Real` 确定. 需满足 `smallEnd < bigEnd`.
- `Real theta`: 方形绕原点逆时针旋转过的弧度. 读入一个 `Real` 确定.
- `bool orientation`: 定向. 读入一个 `bool`, 正定向存为 `true`.
- `Real hL`: 相邻结点间最大距离. 读入一个 `Real` 确定. 需满足 `hL >= 0`.

**作用:** 使用 `sstream` 中相关操作将 `parameters` 转化成输入流, 依次读入参数. 如果 `hL = 0`, 则只取四个顶点作为结点, 否则将每条边均匀分割确定结点, 相邻结点距离不超过 `hL`. 将四个顶点设为尖点, 调用 `OrientedJordanCurve` 中的 `protected` 成员函数 `define` 构造出方形.

## 5 CurveFactory

- 用于根据用户输入的字符串生成 `OrientedJordanCurve<Dim, Order>` 或其子类的对象.

- **模板:** `template <int Dim, int Order>`

- **Public 成员函数:**

(1) `CurveFactory() = default`

默认构造函数.

(2) `std::unique_ptr<OrientedJordanCurve<Dim, Order>> createCurve(const std::string& parameters);`

**输入:** 一行字符串含有曲线的类型及构造所需参数.

**前置条件:** 使用 `sstream` 中相关操作将 `parameters` 转化成输入流, 需首先读入一个 `string` 表示对应曲线类型 `type`. 然后将输入流重新转化为 `string`, 需满足对应曲线类型下 `define` 函数的前置条件.

**输出:** 一个由智能指针存储的 `OrientedJordanCurve<Dim, Order>` 的指针, 指向由用户输入字符串对应的对象.

**作用:** 对第一个读入的字符串 `type` 使用 `switch` 生成对应的对象, 然后调用对象中的 `public` 成员函数 `define` 由剩余输入字符串构造出曲线.

## 6 CurveFactory<2,Order>

- 二维情形下特例化, 可以根据用户输入的一系列字符串生成殷集.

- **模板:** `template <int Order>`

- **Public 成员函数:**

(1) `YinSet<2, Order> createYinSet(const std::vector<std::string>& parameters);`

**输入:** 一系列字符串, 每行字符串含有曲线的类型及构造所需参数.

**前置条件:** 每行字符串需满足 `createCurve` 的前置条件.

**输出:** 由所输入参数生成一系列曲线组合成的殷集.

**作用:** 对每行字符串, 调用 `createCurve` 生成一个 `OrientedJordanCurve<2, Order>` 或其子类的对象. 将所有曲线组合生成殷集.