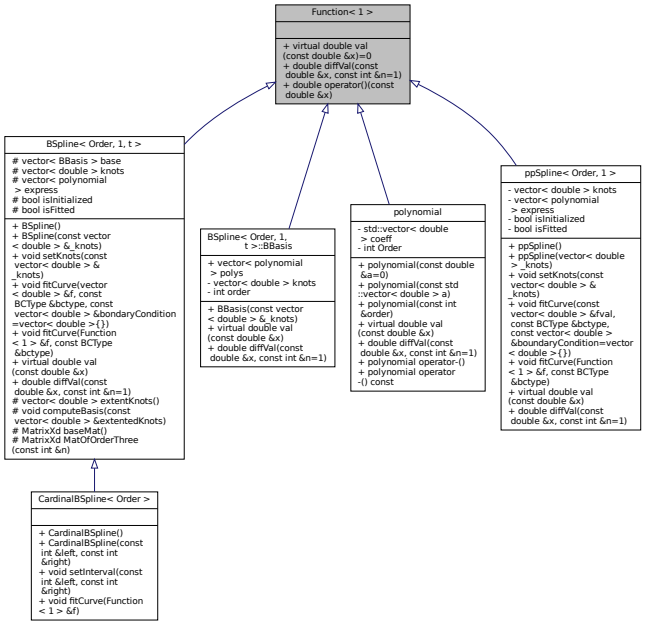
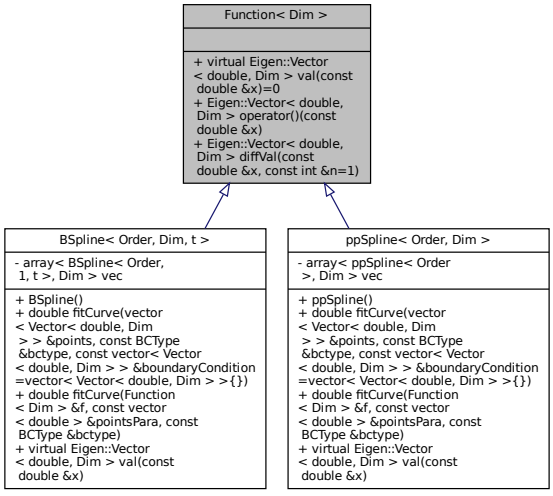
设计文档

陈震翔
3210103924 信息与计算科学

（来不及了，这里就全用中文了，不过后面的文档是用 Doxygen 生成的所以还是英文的，因为容易产生歧义，代码含义中的函数都用接口表述）

1 UML 类图总览





2 设计思路

2.1 $class\ Function < Dim >$

首先因为算法需要定义所需要拟合的函数的接口，并且多项式以及样条都是函数，因此首先设计了一个支持定义 $f : \mathbb{R} \to \mathbb{R}^n$ 的函数抽象模板类 ($Function < Dim >$) 为用户提供定义所需函数的接口，以及作为算法中需要承接函数的接口，并提供了一个接口 ($diffval(x)$) 来计算函数的（高阶）导数，因为 1 情况从一个高维空间中的点会退化成一个实数 (代码中表现为 $Vector < double, Dim >$ 应该退化为 $double$)，对一维函数类进行特化 ($Function < 1 >$)。

2.2 $class\ polynomial$

项目要求实现样条拟合，而样条本质是分段多项式函数，所以通过继承一维函数类设计了多项式类 ($polynomial : public\ Function < 1 >$)，并将运算符 $(+, -, *, /)$ 重载以实现支持多项式之间的运算。

2.3 $class\ spline$

任意维分段样条和 B 样条都是通过先特化实现一维的样条，再利用一维的样条在高维的样条类里去对分量作拟合实现，有一点不同的是高维拟合接口会有一个 double 类型返回值作为 $cumulative\ chordal\ lengths$ 的右端点。

同时对函数的拟合实际上仍是对（高维）散点进行拟合，所以对函数拟合的实现都是通过调用对（高维）散点拟合的接口实现。

这里就只介绍一维的实现思路。

2.3.1 分段多项式样条

参照讲义上的算法先计算得到各个插值节点的二阶导数，再通过二阶导数计算各段多项式的系数存入样条类中。

2.3.2 B 样条

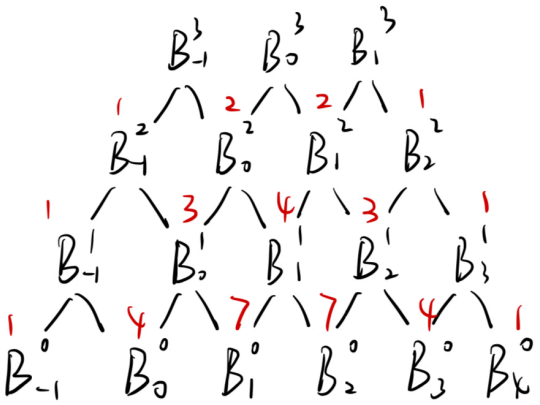
B 样条实现的一大困难是如何去计算它的基函数，计算得到基函数后，利用插值条件（以及边界条件）计算基函数的系数即可。

2.3.2.1 额外型值点的选取 因为 B 样条构成了一个 $n + N - 1$ 维的线性空间（n 是阶数，N 是型值点个数），所以左右各需要 n 个型值点来构建基函数，但是因为并不涉及这些点的取值，所以如何选取这些额外的型值点并不会影响结果，为简单起见并且兼容 cardinal B 样条的型值点，选取的额外型值点为 $t_{1-i} = t_1 - i, t_{N+i} = t_N + i, i = 1, 2, \cdots, N$。

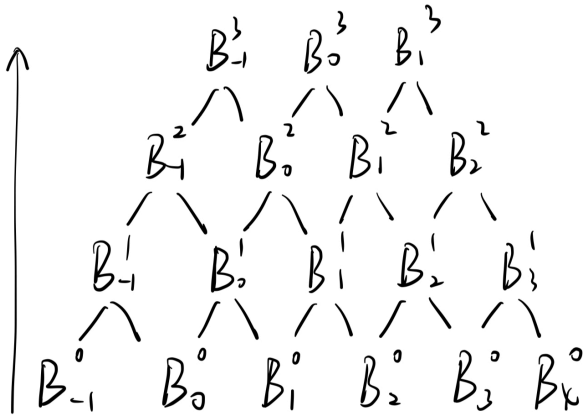
2.3.2.2 基函数 B_i^n 的计算 确定了额外的型值点后，我们就可以计算基函数了，首先我们有基函数的递推公式

$$B_i^{n+1} = \frac{x - t_{i-1}}{t_{i+n} - t_{i-1}} B_i^n(x) + \frac{t_{i+n+1} - x}{t_{i+n+1} - t_i} B_{i+1}^n(x)$$

那我们可以想到，只要定义好 0 阶的基函数就可以直接通过递归调用来实现基函数的计算，而 0 阶的基函数的表达式又是十分简单的。但是这会产生一个问题，这个方式会引起大量的重复计算，使得算法的时间复杂度大大增加，如下图所示 (红色数字为计算次数)：



可以看到仅仅是三阶基函数就会产生大量的重复运算，所以我们运用动态规划的思想，将每次运算的基函数都存储下来，代码中的实际实现是将递归循环展开，从底层往上计算，如图所示



这样算法的时间复杂度为 $O(n(n+N))$，并且如果从左到右按顺序计算各阶基函数，每次计算后就可以直接覆盖掉低阶基函数，这样空间复杂度仅为 $O(n+2N-1)$。

2.3.2.3 $\mathbb{S}_n^{n-1}$ 的实现（其实还没写完 有了上面的算法我们就可以很轻松的得到 $\mathbb{S}_n^{n-1}$ 的基，但是对于 $\mathbb{S}_n^{n-1}$ 的计算除了各个型值点处的值之外我们还缺少 $n-1$ 个条件，即边界条件。一开始的想法是对左右端点的其中一个做限制，使样条的 $i(<n)$ 阶导数与拟合函数相等，但实际实现测试后发现，拟合的结果十分不理想，在三次样条时在远离做了限制的端点的区域内就会出现极大的波动，对比讲义中的边值计算的条件数发现一个为个位数，一个为一万多，并且随着样条阶数的增加会急速增长，在 $n=6$ 时已经大于 10^{16}（大概是这个，之前测试的数据没有及时保存，丢失了，暴风哭泣），并且这部分实现在对样条进行模板化并支持高维散点的调整的时候不再适配了，也没有时间再修改，目前注释在一维 B 样条的 fitCurve() 对边界条件为 $myDefault2$ 的实现中。目前有两种其他的边界条件的想法

- 对两个端点都做限制，条件为两个端点处的 $i \leq \lfloor \frac{n}{2} \rfloor$ 阶导数都与原函数相等
- 将条件分散在各个型值点上，尽量使条件所需导数的阶数尽可能低，并且在条件少于型值点个数时，优先对靠近两侧的型值点做限制

这两种条件在三次样条时是完全样条或者特定二阶导数的样条的边界条件，至少保证了在三次样条时的条件数并不大，但是已经没有时间去实现并做相关测试了，烂尾了呜呜呜

3 代码文档

代码文档通过添加注释后由 Doxygen 生成，见 design document.pdf