设计文档

Wenchong Huang

日期：2022 年 12 月 24 日

1 设计思路

- `class Function`: 函数基类。
- `class T_Polynomial<T>`: 多项式模板类。
- `class ppForm_base`: 多项式样条插值基类。
- `class ppForm_linear`: 线性多项式样条插值类，继承`ppForm_base`。
- `class ppForm_cubic`: 三阶多项式样条插值类，继承`ppForm_base`。
- `class BSpline_base`: B 样条插值基类。
- `class BSpline_linear`: 线性 B 样条插值类，继承`BSpline_base`。
- `class BSpline_cubic`: 三阶 B 样条插值类，继承`BSpline_base`。

此外，本项目还引用了作者上学期在《优化实用算法》中完成的矩阵库，包括类 `Matrix` 及其派生类 `ColVector`，`RowVector`，封装了矩阵的一些常用运算。篇幅所限，在这篇设计文档中作者不介绍矩阵库的设计思路。

1.1 class Function

函数基类规定一个抽象函数，需要具有求值（小括号运算）、求导、求二阶导三项基本功能，其中求值运算定义为虚函数，求导定义小差商为默认方法，允许函数实例用导函数的解析式替换默认方法。原型如下。

```
class Function{
  public:
    virtual double operator () (const double &x) const = 0;
    double diff (const double &x) const;
    double diff2 (const double &x) const;
};
```

1.2 class T_Polynomial<T>

多项式模板类。用一个 `std::vector<T>` 存储各项系数，支持多项式的求值、求导、加减乘运算、不同格式的输出。原型如下。

```
template<class T> class T_Polynomial{
  private:
```

```
    static int outputMode;

  protected:
    std::vector<T> coef;
    int n;

  public:
    T_Polynomial();
    T_Polynomial(const T &x);
    T_Polynomial(const T &k0, const T &k1);
    T_Polynomial(const std::vector<T> & coef);
    T_Polynomial(const T_Polynomial & p);

    static const int OUTPUT_LATEX = 0;
    static const int OUTPUT_TIKZ = 1;
    static void setOutput(const int & style);
    friend std::ostream & operator << (std::ostream &out, const T_Polynomial &ni);

    T operator () (const T &vx) const;
    T_Polynomial operator + (const T_Polynomial &rhs);
    T_Polynomial operator - (const T_Polynomial &rhs);
    T_Polynomial operator * (const T &rhs);
    T_Polynomial operator * (const T_Polynomial &rhs);
    T_Polynomial diff();
};
```

定义 Polynomial 为模板类的生成类 T_Polynomial<double>。

```
typedef T_Polynomial<double> Polynomial;
```

1.3 class ppForm_base

主要用于存储分段插值多项式，保存插值结点、每一段的多项式。实现求值（小括号运算）。原型如下。

```
class ppForm_base{
  protected:
      int n;
      std::vector<double> knots;
      std::vector<Polynomial> poly;
  public:
      ppForm_base();
      ppForm_base(const ppForm_base &rhs);
      double operator () (const double &x) const;
};
```

1.4 class ppForm_linear

继承ppForm_base，实现“给定插值点和对应值”的基本构造函数，调用基本构造函数实现“给定插值点和被拟合函数”、“给定区间、插值点数量、被拟合函数”的构造函数。原型如下。

```
class ppForm_linear : public ppForm_base{
  public:
    ppForm_linear(const std::vector<double> &x, const std::vector<double> &f);
    ppForm_linear(const std::vector<double> &x, Function & func);
    ppForm_linear(const int &n, const double &l, const double &r, Function &func);
};
```

1.5 class ppForm_cubic

继承ppForm_base，实现“给定插值点和对应值、边界条件”的基本构造函数，调用基本构造函数实现“给定插值点和被拟合函数、边界条件”、“给定区间、插值点数量、被拟合函数、边界条件”的构造函数，同时对三种构造函数给出边界条件缺省时的构造函数，默认边界条件为*natural*。原型如下。

```
class ppForm_cubic : public ppForm_base{
  private:
    // 初始化端点处的函数值条件、端点处的一阶导数与二阶导数连续条件
    void basic_init(Matrix &A, ColVector &b, const std::vector<double> &x, const
        std::vector<double> &f);

  public:
    // “给定插值点和对应值、边界条件”的基本构造函数。调用basic_init初始化基本条
        件，再根据bondary添加边界条件，然后求解
    ppForm_cubic(const std::vector<double> &x, const std::vector<double> &f, const
        std::string &bondary);
    ppForm_cubic(const std::vector<double> &x, const std::vector<double> &f):
        ppForm_cubic(x, f, "natural") {}

    // “给定插值点和被拟合函数、边界条件”的构造函数
    ppForm_cubic(const std::vector<double> &x, Function &f, const std::string &
        bondary);
    ppForm_cubic(const std::vector<double> &x, Function &f): ppForm_cubic(x, f, "
        natural") {}

    // “给定区间、插值点数量、被拟合函数、边界条件”的构造函数
    ppForm_cubic(const int &n, const double &l, const double &r, Function & func,
        const std::string &bondary);
    ppForm_cubic(const int &n, const double &l, const double &r, Function & func):
        ppForm_cubic(n, l, r, func, "natural") {}
};
```

下面给出一个三阶多项式样条插值的使用实例。以拟合螺旋线的 x 坐标为例，边界条件为 *complete*。以点值格式输出。

```
class Helix_X : public Function{
  double operator () (const double &x) const{
      return x*cos(x);
  }
  double diff (const double &x) const{
      return cos(x)-x*sin(x);
  }
  virtual double diff2 (const double &x) const{
      return -2*sin(x)-x*cos(x);
  }
} hx;
const double pi = acos(-1);
ppForm_cubic pp_x(17, 0, 4*pi, hx, "complete");
cout << "x1 = [";
for(double t = 0; t <= 4*pi; t += 0.01)
  cout << pp_x(t) << " ";
```

1.6 class BSpline_base

B 样条插值基类。用两个 `std::vector<double>` 分别存储插值点和每个 B 样条基对应的系数。实现递归求 B 样条基 $B_i^k(x)$，用公式

$$\frac{\mathrm{d}}{\mathrm{d}x}B_i^k(x)=\frac{kB_i^{k-1}(x)}{t_{i+k+1}-t_{i-1}}-\frac{kB_{i+1}^{n-1}(x)}{t_{i+k}-t_i}$$

实现 B 样条基的求导、二阶导。另外定义虚函数 $B_i(x)$ 和对应的导数、二阶导，在线性插值类中该虚函数返回 $B_i^1(x)$，在三阶插值类中该虚函数返回 $B_i^3(x)$。

最后还要实现求点值（小括号运算）。

```
class BSpline_base{
  protected:
    vector<double> coef;
    vector<double> knots;

    double B(const int &i, const int &k, const double &x) const;
    double dB(const int &i, const int &k, const double &x) const;
    double d2B(const int &i, const int &k, const double &x) const;

    virtual double B(const int &i, const double &x) const = 0;
    virtual double dB(const int &i, const double &x) const = 0;
    virtual double d2B(const int &i, const double &x) const = 0;

  public:
    BSpline_base(){}
```

```
    BSpline_base(const BSpline_base & rhs);
    double operator () (const double &x) const;
  };
```

最后，`class BSpline_linear` 与 `class BSpline_cubic` 提供的接口与对应的 ppForm 插值类完全一致，其设计思路不再赘述。