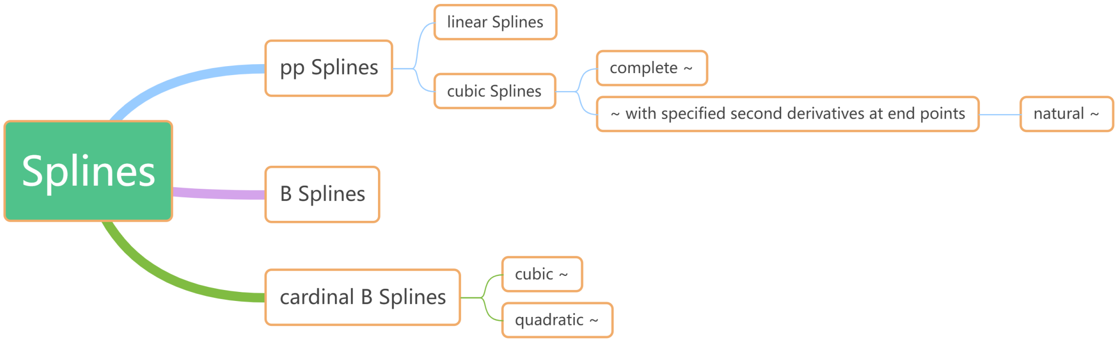
Program Design for Splines

Pingchuan Li

November 2023

1 Inheritance Relationship of Different Splines



2 Objects&Functions in Different Classes

2.1 class Splines

This is the head class, and there should be objects and functions which are applicable in all sub-classes.

Objects:

`int n`: the number of knots

`double* x`: the list of knots' values, `x[1]`,`x[2]`,$\cdots$,`x[n]`

`double* fx`: the list of functioned knots' values, `fx[1]`,`fx[2]`,$\cdots$,`fx[n]`

`double a`: left end-point of the whole interval

`double b`: right end-point of the whole interval

Functions:

`spline(int n, double* x, double* fx, double a, double b)`: the constructor

`double solve(double t)`: a virtual function, get the spline's value of t (written as spline(t)) after forming the spline

`void interpolate_for_graph(string file_name)`: generate data, where `x[]` are uniformly-spaced picked on $[a, b]$ and `y[]` are values of `solve(x[])`, for later graph in txt file

2.2 class `linear_spline`

This class can be simply built.

2.3 class `cubic_spline`

This class derives from class `spline`.

All extra objects are intermediate for the process of spline's solution.

`cubic_spline(int n, double* x, double* fx, double a, double b, double fa_diff, double fb_diff)`: the last two parameters are given for divided difference table

`double spline_s(int i, double t)` & `double solve(double t)`: t is first processed in `solve()` to obtain the correspondent formula in `spline_s()`, by which `spline_s()` generates the answer

Considering different boundary conditions, class `cubic_spline` can be classified into class `complete_cubic_spline`, class `specified_2d_cubic_spline`, class `natural_cubic_spline`, etc.

2.3.1 class `complete_cubic_spline`

No extra parameters are needed for spline's solution.

2.3.2 class `specified_2d_cubic_spline`

Extra $f"(a)$ and $f"(b)$ are need for spline's solution.

2.3.3 class `natural_cubic_spline`

This class can simply derive from the upper class since it is a particular type with $f"(a) = 0$ and $f"(b) = 0$.

2.4 class `B_spline`

Extra objects:

`int d`: spline of degree d and smoothness $d-1$, written as $\mathbb{S}_d^{d-1}$

`double* u`: knot points, which differ from knots. Knots are given in initialized parameters, while knot points are generated in the constructor (there are many methods to generate knot points, and uniformly-spaced picking in $[a,b]$ is most simple). There should be n knots and $n+d+1$ knot points.

Extra function:

`double B_Basis(int i, int d, double t)`: constructed by recursion

2.5 class `cardinal_B`

Extra objects:

`double move`: Since the two types of cardinal B spline below are resolved by assuming knots in interval $[1,?]$, to deal with splines with interval $[a,b]$, we need to first move the spline right to $[1,?]$. (the amount of move is $1 - a$)

`double* _a`: intermediate for the process of spline's solution

2.5.1 class `cubic_cardinal_B`

`double Basis(int i, double x)`: Simplified from that in B spline by given knots $[1, 2, \cdots, n]$ and degree 3

`double solve(double t)`: Since the spline moves right for the solution (we keep the actual spline not move as we do not change `x[]`), we need to moves t right with the same amount to get the value.

2.5.2 class `quadratic_cardinal_B`

Similar as above.