# Optimization in Function Fitting and Interpolation

AI2101 PROJECT

GROUP - 30

## Group Members

Boda Devaki – ES20BTECH11008

Chemikala Bhanu Teja – ES20BTECH11010

Muvva Thanmai – ES20BTECH11019

N. Dilip Kumar Reddy – ES20BTECH11020

Narayana Banu Bharadwaj – ES20BTECH11021

## What is Function Fitting and Interpolation?

**Curve fitting** is the process of constructing a curve, or mathematical function, that has the best fit to a series of data points, possibly subject to constraints.

**Interpolation** is a statistical method by which related known values are used to estimate an unknown price or potential yield of a security.

## Why study function fitting?

Fitted curves can be used as an aid for data visualization, to estimate values of a function where no data are available, and to summarize the relationships among two or more variables.

Instead of saving complete data in the form of points and tables, it is memory efficient to save it as a function.

## Methods of solving Function Fitting

Using splines as basis functions

Minimum norm function fitting

Gradient descent algorithms

Using genetic algorithms

# Minimum Norm Function Fitting

In this problem, a function f ∈ F that matches the given data, $(u_1, y_1)$, .., $(u_m, y_m)$ with $u_i \in D$ and $y_i \in R$ as closely as possible is obtained.

Number of constraints can be added to the problem like linear inequalities satisfying the function at various points, derivative constraints, monotonicity constraints.

Example: $minimize \sum_{(i=1)}^{m} (f(u_i) - y_i)^2$

# Gradient Descent Algorithms

In this problem of finding smooth curves to the given set of data points $p_0$, $p_1$, ..., $p_n$, on non-linear manifolds at distinct and ordered instants of time, the obtained smooth curve γ involves two goals of conflicting nature.

- The curve best approximates the data. $E_d(\gamma) = \sum_{i=0}^{n} d^2(\gamma(t_i), p_i)$ where d denotes the distance function on the Riemannian manifold M.
- The curve should be sufficiently regular, i.e., the changes in the velocity or the acceleration is minimized.

Thus, it results in a optimization problem with two objective functions, fitting function $E_d$ and regularity function $E_s$, whose domain is the suitable set of curves in the Riemannian manifold M.

-

## Genetic Algorithms

Genetic Algorithms are the efficient search algorithms working with the bit strings based on the mechanics of natural genetics. These GAs are used in curve fitting to avoid the complicated and unreliable process of finding the fitting parameters.

Its goal is to minimize the curvature integral in order to reach an optimal shape. A real-coded genetic algorithm is developed in to find good knots of a fitting spline.

These genetic algorithms are run simultaneously and cooperatively, consisting of a coevolutionary genetic algorithm. The final solution of the fitting algorithm is derived by combining the partial solutions of the GAs.

# Function Fitting using B-splines

**Basis**: Consider a family of functions $f_1, \ldots, f_n : R^k \to R$, with common domain dom $f_i = D$.

With each $x \in R^n$ we associate the function $f : R^k \to R$ given by $f(u) = x_1 f_1(u) + \cdots + x_n f_n$ with dom $f = D$.

The family $\{f_1, \ldots, f_n\}$ is called the set of basis functions

**Splines**: A function s(x) defined on a finite interval $[x_{min}, x_{max}]$ whose degree $k(\geq 0)$ with knots denoted by a strictly increasing sequence $\lambda_i$ where $i = 0$ to $g+1$,
->On each knot interval, s(x) is a polynomial of at-most degree k.
->S(x) and it's derivatives up to order (k-1) are continuous on the interval $[x_{min}, x_{max}]$.

**B-splines**: B-splines or Basis-splines are used as a basis to represent polynomial splines.

A B-spline of degree l having order (l+1) can be computed by the De-Boor recursive formula, given by:

$$B_{i,l+1}(x) = \frac{x-\lambda_i}{\lambda_{i+l}-\lambda_i} B_{i,l}(x) + \frac{\lambda_{i+l+1}-x}{\lambda_{i+l+1}-\lambda_{i+1}} B_{i+1,l}(x)$$

with the base case of degree 0 as:

$$B_{i,1}(x) = \begin{cases} 1, if x \in [\lambda_i, \lambda_{i+1}). \\ 0, \quad otherwise. \end{cases}$$

For a given sequence of knots($\lambda_i$ ,i=0,1,….,g+1), the dimension of vector space of spline functions is g+k+1, but only g-k+1 splines are generated from the given knots.

To solve this, 2k new knots are added coinciding with the boundaries of the given interval.

$(\lambda_{-k}= \ldots\ldots = \lambda_{-1} = \lambda_0 = X_{min})$ and $(\lambda_{g+1}= \lambda_{g+1} \ldots\ldots = \lambda_{g+k+1} = X_{max})$

Now, s(x) can be represented as linear combination of above generated basis functions.

$\mathrm{s}(x) = \sum_{i=-k}^{g}\left(c_i B_{i,k+1}(x)\right)$, where $c_i$ are spline coefficients.

- **Derivative of spline function:** The v'th derivative of s(x) can be written as:

$$s^v(x) = \prod_i^v (k+1-i) \sum_{i=-k+v}^{g} (c_i^v B_{k+1-v}(v))$$

with

$$c_j^{(i)} = \begin{cases} c_j\,, & \text{if } i = 0\,, \\ \frac{c_j^{(i-1)} - c_{j-1}^{(i-1)}}{\lambda_{j+k+1-i} - \lambda_j} & \text{if } i > 0\,. \end{cases}$$

# Optimization Problem

Now, lets see how an optimization problem is constructed. Given m points(x) and their function values($y_r$) in an interval,

=>The spline coefficients of function and its k derivatives are taken as optimization variables in the form of a vector of dimension (k+1)(g + 1 +k/2)

$$c = [\, c_{-k}^{(0)} \ldots\ldots\ldots c_g^{(0)} \; c_{-k+1}^{(1)} \ldots\ldots\ldots\ldots c_g^{(1)} \ldots\ldots\ldots c_0^{(k)} \ldots\ldots\ldots\ldots c_g^{(k)} ]^T$$

=> Since there are only g + k + 1 independent variables, (k+1)(g + 1 + k/2 ) constraints must be added. These constraints are found in derivative equations of $c_i$ , which are simple linear equality constraints that are added to the problem as $C \cdot c = 0$ where C is a sparse banded matrix.

For a spline with knots $\lambda_i$ and degree k and for measurements $(x_r , y_r )$, the basic approximation problem we want to solve is:

**Objective** : $\underset{c}{\text{minimize}} \sum_{i=1}^{g} w_i |s^{(k)}(\lambda_i +) - s^{(k)}(\lambda_i -)|$

**Constraints** : $\sum_{r=1}^{m} \left(v_r\left(y_r - s(x)\right)\right)^2 \leq S$

$$C.\boldsymbol{c} = 0$$

$$g\left(s(x), \dots \dots \dots s^k(x)\right) = 0$$

$$h\left(s(x), \dots \dots \dots s^k(x)\right) \leq 0$$

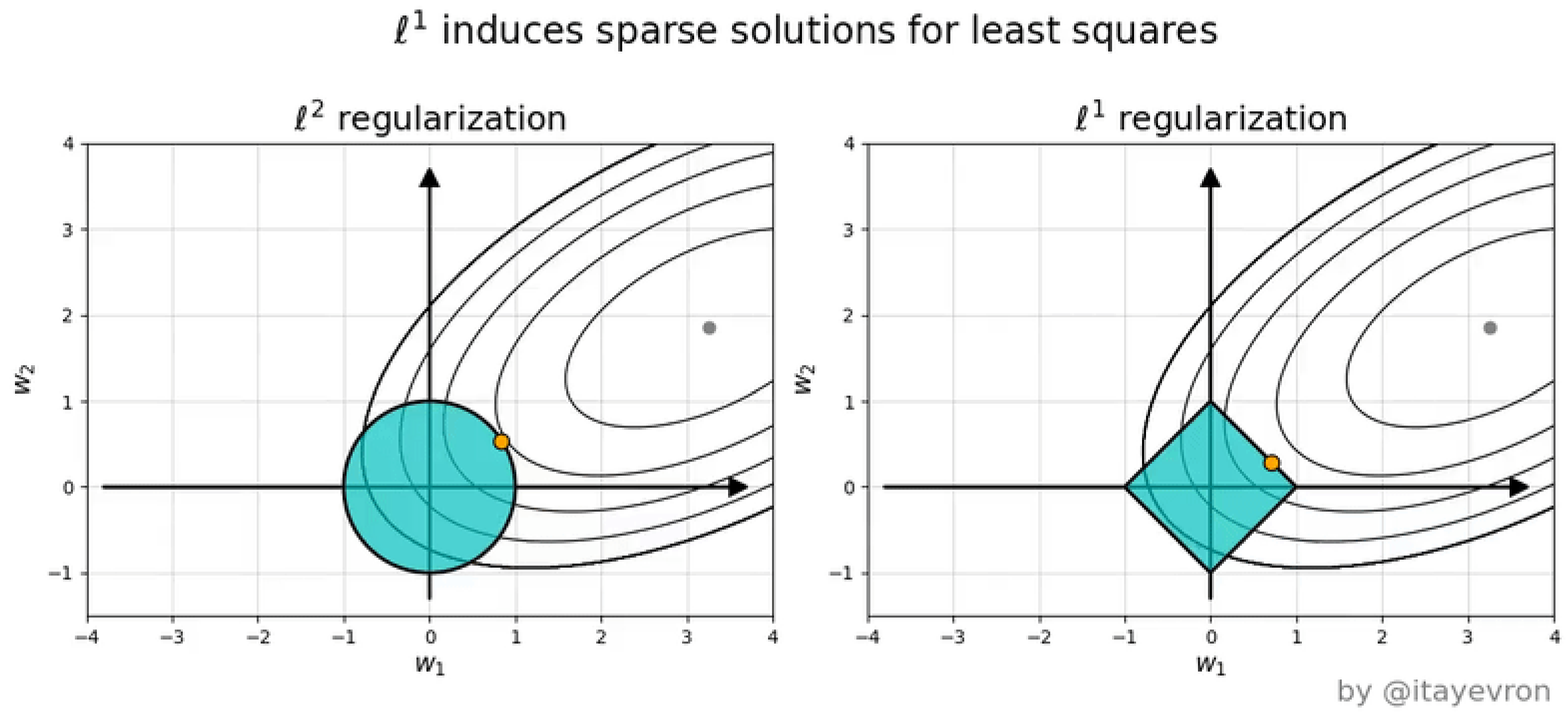
Where $w_i$ , $v_i$ are chosen weights, S is a fixed parameter to control quality of fit, and g , h are additional constraints for the problem.

## L1 Regularization vs L2 Regularization

Although L2 norm smoothens the curve fit, L1 norm is preferred in objective of optimization over it because:

- L1 norm measures the non-smoothness of s(x)
- L1 norm yields sparse solutions i.e., it yields a solution which has few non-zero elements in $s^{(k)}(\lambda_i +) - s^{(k)}(\lambda_i -)$.

# l1 norm vs l2 norm



## Improving sparsity of solution:

Initially, weights wi , for I = 1,……,g are taken to be ‘1’.

If still too many active knots are observed in the solution, a reweighted L1 minimization is performed to improve sparsity of the solution. In each reweighting iteration j, the `1 optimization problem (11) is solved and the weights $w_i$ are updated as:

$w_i^{(j+1)} = \frac{1}{\left|s^{(k)}(\lambda_i+) - s^{(k)}(\lambda_i-)\right| + \epsilon}$ where ε should be chosen slightly smaller than the expected nonzero magnitudes of $s^{(k)}(\lambda_i +) - s^{(k)}(\lambda_i -)$.

**Note :** A knot is called active if the kth order derivative of s(x) is discontinuous at that point ,and is inactive otherwise.

## EXAMPLES

# 1.Unconstrained Optimization

In this example, the titanium heat data (from de boor and rice) is fitted.

- This problem is solved by cubic splines(degree, k= 3) and S(our parameter to control quality of fit) = 0.0073, and the weights $v_r$ are taken as 1, and 1000 equidistant candidate knots.
- Non-smoothness obtained from this convex optimization(0.0433) method is less than the non-smoothness obtained from Dierickx (0.0899) which uses L2 norm in objective and Jupp’s method(0.0742).

## 2.Constrained Optimization

A constraint of the form $f < s < g$ is added to the above problem where f,s and g are some functions of optimization variables.

**Problem:** Consider measurements of n marker positions $P_i$ , $i = 1,2,3$ as ends of a rigid body moving in 3D let the co ordinates be $x_i$, $y_i$, $z_i$ of a point i and the function $S_{xi}$, $S_{yi}$, $S_{zi}$ are spline functions of fitted values of $x_i$, $y_i$, $z_i$ as a function of time. Measurement noise is considered $\pm\delta$.

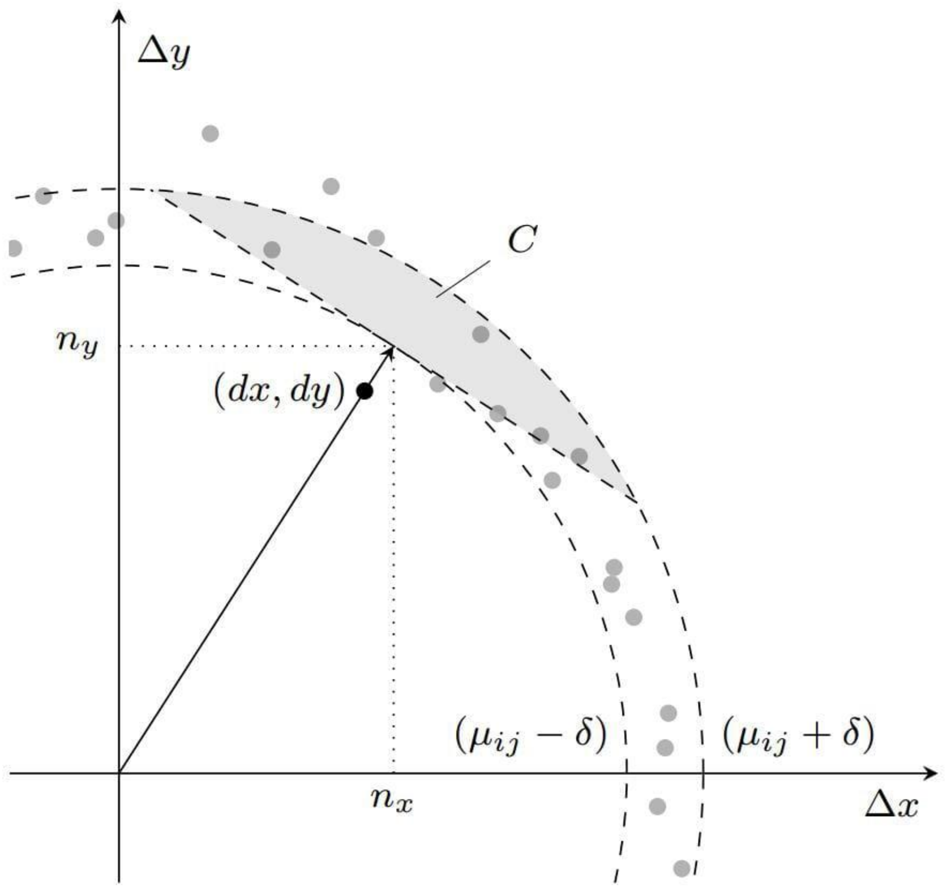
Constraint on the given problem :

$$(\mu_{ij} - \delta)^2 \leq \sum(S_{li} - S_{lj})^2 \leq (\mu_{ij} + \delta)^2$$

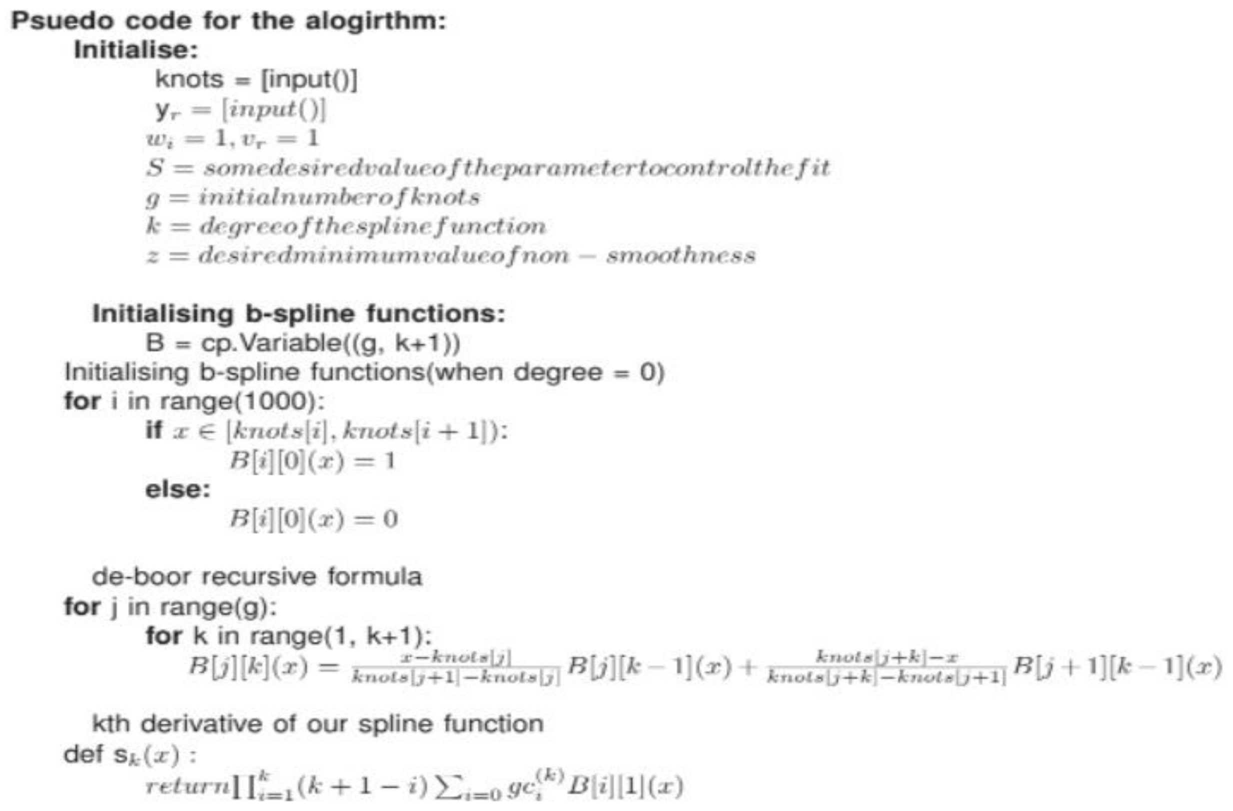
The LHS constraint

$(\mu_{ij} - \delta)^2 \leq \sum(S_{li} - S_{lj})^2$ is concave(outside part of circle of radius).

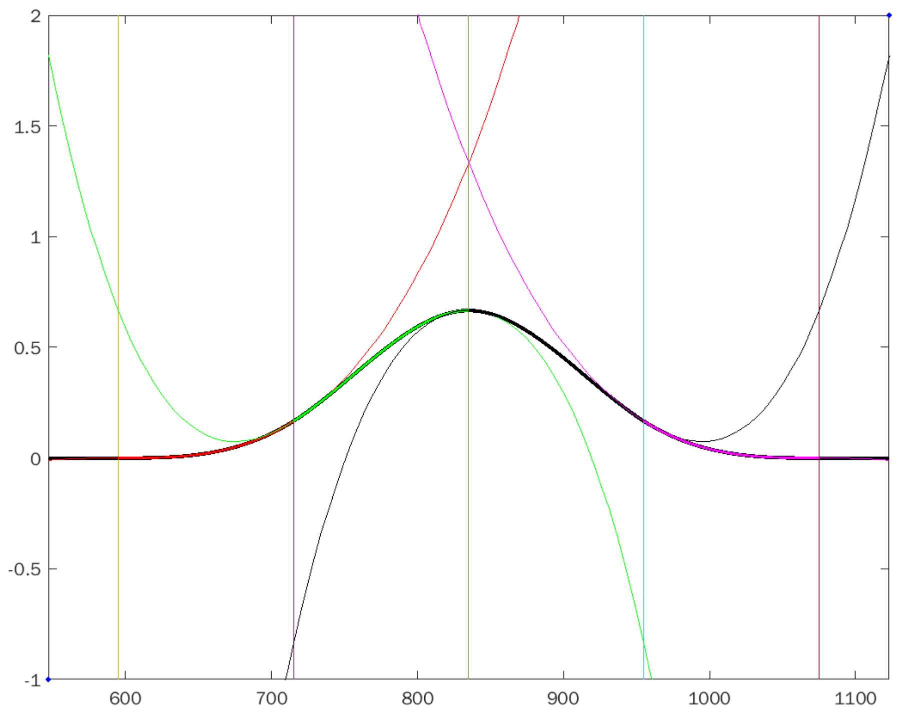
We use linearization of the domain to solve the problem.



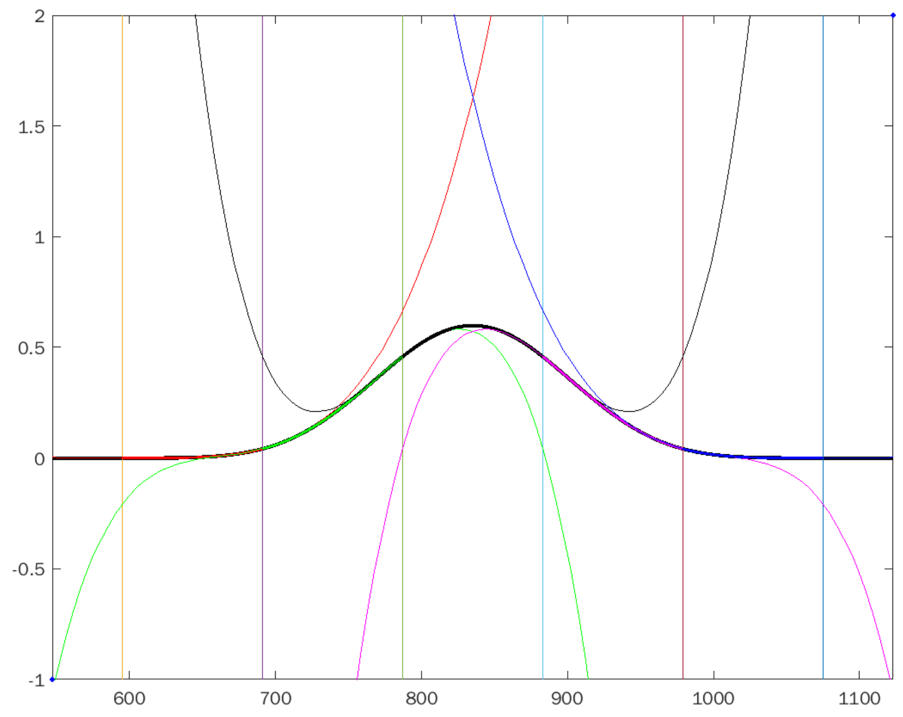
## Generalized Pseudo Code



```
Psuedo code for the alogirthm:
    Initialise:
        knots = [input()]
        $y_r = [input()]$
        $w_i = 1, v_r = 1$
        $S = somedesiredvalueoftheparametertocontrolthefit$
        $g = initialnumberofknots$
        $k = degreeofthesplinefunction$
        $z = desiredminimumvalueofnon - smoothness$

    Initialising b-spline functions:
        B = cp.Variable((g, k+1))
    Initialising b-spline functions(when degree = 0)
    for i in range(1000):
        if $x \in [knots[i], knots[i+1])$:
            $B[i][0](x) = 1$
        else:
            $B[i][0](x) = 0$

    de-boor recursive formula
    for j in range(g):
        for k in range(1, k+1):
            $B[j][k](x) = \frac{x - knots[j]}{knots[j+1] - knots[j]} B[j][k-1](x) + \frac{knots[j+k] - x}{knots[j+k] - knots[j+1]} B[j+1][k-1](x)$

    kth derivative of our spline function
    def $s_k(x)$ :
        $return \prod_{i=1}^{k} (k+1-i) \sum_{i=0} gc_i^{(k)} B[i][1](x)$
```



A cubic(k=3)  spline with 4 active knots



A cubic(k=3) spline with 5 active knots

```
 //The optimisation problem
while $\sum_{i=1}^{g} w_i|s^{(k)}(knots_i+) - s^{(k)}(knots_i-)| \geq z$ :
    objective = cp.Minimize($\sum_{i=1}^{g} w_i|s^{(k)}(knots_i+) - s^{(k)}(knots_i-)|$)
    constraints = [$\sum_{r=1}^{m}(v_r(y_r - s(x_r)))^2 \leq S$,
    $g(s(x), ..., s^{(k)}(x)) = 0$,
    $h(s(x), ..., s^{(k)}(x)) \leq 0$]
    //g and h are problem specific constraints
    //c's of b-splines recursively using c's of previous order of b-splines:
    for i in range(k):
        for j in range(g + k + 1 - i):
            if $i == 0$:
                constraints.append($c_j^i = c_j$)
            else if $i > 0$:
                constraints.append($c_j^i = \frac{c_j^(i-1) - c_(j-1)^(i-1)}{knots[j+k+1-i] - knots[j]}$)

     //weights are updated
    $w_i = \frac{1}{|s^{(k)}(knots[i]+) - s^{(k)}(knots[i]-)| + \epsilon}$
```

## References

- A Convex Optimization Approach to Curve Fitting with B-Splines by W.Van Loock ,G.Pipeleers,J.De Schutter and J. Swevers, Proceedings of the 18th World Congress ,IFAC, Milano(Italy), August28-September 2,2011.

  Interactive Spline Approximation by Marian Merchant, Simon Fraser University,1971.

  Least Squares Cubic Spline Approximation I-Fixed Knots by Carl de Boor and John R.Rice, April 1968, Purdue University.

  Chapter 6.5 in Convex Optimization by Stephen Boyd, Stanford University

## THANK YOU