Project 2

Kerman Munitxa, Júlia Gumà, Markel Pisano
kerman.m@digipen.edu j.gum@digipen.edu markel.p@digipen.edu
540002517 540000117 540002615

1. Description of the problem

It is a method for, given some points, creating a curve (2D or 3D) that passes through them. The input data are the points for the curve. The output curve is twice differentiable, so it is smooth.

2. Mathematical explanation of numerical methods

mat300 slides (a) mat300 slides (b)

We are given $n+1$ points: $P_0, P_1, \dots, P_n$. First of all, we create a regular mesh of nodes in the interval $[0, n]$ satisfying $0 = t_0 < t_1 < t_2 < \cdots \geq t_n = n$, and same distance between nodes.

When using a mesh of nodes instead of points, we are making sure that all elements in the mesh are different. If there were at least two elements with the same value, we couldn't find a curve.

A cubic spline interpolation consists of finding an interpolation polynomial $p \in P^n_{3,2}t_0, t_1, \dots, t_n$ through the given points, where n is the number of nodes + 1. This means that we will get a set of n piecewise polynomials of degree at most 3 that are at least twice differentiable. These polynomials will go from t_{n-1} to t_n, being:

$$p(t) = \begin{cases} \mathrm{p}_1(t) & t \in [t_0, t_1] \\ p_2(t) & t \in [t_1, t_2] \\ \vdots & \vdots \\ p_n(t) & t \in [t_{n-1}, t_n] \end{cases}$$

For doing so, out basis will be $B = \{1, t, t^2, t^3, (t-t_1)^3_+, (t-t_2)^3_+, \dots, (t-t_{n-1}{}^3_+\}$. Having that, we can express p as a linear combination of elements of B like this:

$$p(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 (t-t_1)^3_+ + a_5 (t-t_2)^3_+, \dots, a_{n+2}(t-t_{n-1})^3_+$$

Given $n+1$ nodes and points $(t_0, P_0), (t_1, P_1), \dots, (t_n, P_n)$, the cubic spline is determined by the solution in $(a_0, a_1, \dots, a_n, a_{n+1}, a_{n+2})$ of the system of equations. We only have n constraints, and for obtaining a unique solution we need to impose two constraints more.

This is why we impose the condition for the second derivatives of the first and last point are 0:

$$p''(t_0) = 0$$
$$\text{p''}(\text{t}_n) = 0$$

We will have to solve the system of equations, that looks like this

$$\begin{cases} \text{P}_0 = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 + a_4 (t_0 - t_1)_+^3 + a_5 (t_0 - t_2)_+^3, \ldots, a_{n+2}(t_0 - t_{n-1})_+^3 \\ P_1 = a_0 + a_1 t_1 + a_2 t_1^2 + a_3 t_1^3 + a_4 (t_1 - t_1)_+^3 + a_5 (t_1 - t_2)_+^3, \ldots, a_{n+2}(t_1 - t_{n-1})_+^3 \\ P_2 = a_0 + a_1 t_2 + a_2 t_2^2 + a_3 t_2^3 + a_4 (t_2 - t_1)_+^3 + a_5 (t_2 - t_2)_+^3, \ldots, a_{n+2}(t_2 - t_{n-1})_+^3 \\ \vdots \\ P_n = a_0 + a_1 t_n + a_2 t_n^2 + a_3 t_n^3 + a_4 (t_n - t_1)_+^3 + a_5 (t_n - t_2)_+^3, \ldots, a_{n+2}(t_n - t_{n-1})_+^3 \end{cases}$$

For x,y and, if we are in 3D, z. This gives us the following matrix:

$$\left(\begin{array}{cccccccc|c|c|c} 1 & t_0 & t_0^2 & t_0^3 & (t_0 - t_1)_+^3 & (t_0 - t_2)_+^3 & \cdots & (t_0 - t_{n-1})_+^3 & x_0 & y_0 & z_0 \\ 1 & t_1 & t_1^2 & t_1^3 & (t_1 - t_1)_+^3 & (t_1 - t_2)_+^3 & \cdots & (t_1 - t_{n-1})_+^3 & x_1 & y_1 & z_1 \\ 1 & t_2 & t_2^2 & t_2^3 & (t_2 - t_1)_+^3 & (t_2 - t_2)_+^3 & \cdots & (t_2 - t_{n-1})_+^3 & x_2 & y_2 & z_2 \\ \vdots \\ 1 & t_n & t_n^2 & t_n^3 & (t_n - t_1)_+^3 & (t_n - t_2)_+^3 & \cdots & (t_n - t_{n-1})_+^3 & x_n & y_n & z_n \end{array}\right)$$

After solving the system (applying Reduced Row Echelon Form), we obtain the vector of coefficients for each coordinate. Replacing each coordinate into the polynomials gives us the resulting curve.

3. Code implementation

First of all we create an array of $t : [0, 1, 2, \ldots, n-1, n]$, which acts as a mesh of nodes. After this, we create a matrix of polynomials of the form:

$$p(t) = a_0 + a_1 t_i + a_2 t_i^2 + a_3 t_i^3 + a_4 (t_i - t_1)_+^3 + a_5 (t_i - t_2)_+^3, \ldots, a_{n+2}(t_i - t_{n-1})_+^3,$$

begin $i = 0, \ldots, n$ the row number. We fill the last columns with the corresponding value at t for each of the axis (line 28).

After this, we add another two rows to the matrix, which corresponds to the second derivatives to the extreme points, equaling to 0 on the augmented part (line 55).

Then we apply *Reduced Row Echelon Form* to extract the values of a for x, y and z (line 72).

After this we substitute resulting a on the polynomial coefficients and using a mesh of t values $\in [0, n]$ with *outputnodes* (this variable changes the precision on the drawing of the curve). This gives points in the curve to be plotted (line 86).

4. Examples

Lets take for instance the set of points $S = \{P_0, P_1, P_2\}$, being $P_0 = (0, 1)$, $P_1 = (1, 2)$, $P_2 = (0, 3)$.

Our set of t are $T = \{0, 1, 2\}$.

Our basis will be $B = \{1, t, t^2, t^3, (t-1)^3_+\}$

We get the augmented matrix of polynomials:

$$\left[\begin{array}{ccccc|c|c} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 2 \\ 1 & 2 & 4 & 8 & 1 & 0 & 3 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 12 & 6 & 0 & 0 \end{array}\right] \tag{1}$$

Now we apply *RREF* to it and we get the coefficients for X and Y from the last two columns.

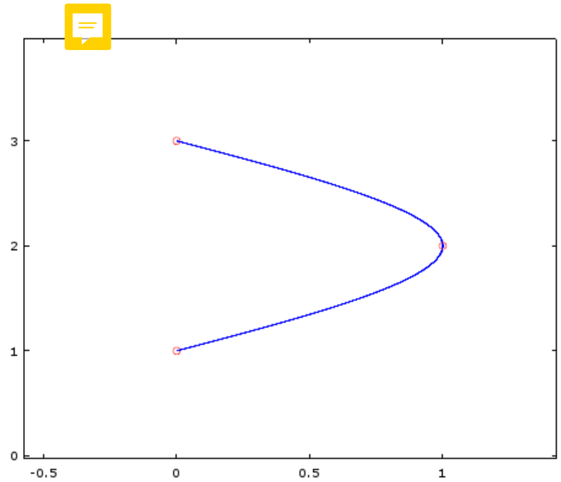
$$\left[\begin{array}{ccccc|c|c} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & \frac{3}{2} & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{array}\right] \tag{2}$$

Now we substitute this coefficients in the polynomial

$P(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 (t-1)^3_+$

giving

$P_x(t) = \frac{3}{2}t - \frac{1}{2}t^3 + (t-1)^3$ and $P_y(t) = 1 + t$.



5. Observations

Cubic Splines are useful because changing points changes very little the resultant curve, not like interpolation methods such as *Newton method*. The curves are smooth because are twice differentiable. However, when having a lot of points, our maximum t to the cube will end up being so big that it will start losing precision.

References

mat300 slides. Osculating polynomials. https://distance.digipen.edu/2020-spring/pluginfile.php/68872/mod_resource/content/1/lecture7.pdf, a.

mat300 slides. Cubic spline. https://distance.digipen.edu/2020-spring/pluginfile.php/72175/mod_resource/content/1/lecture9.pdf, b.