Daniel Herreros| 540002818
David Miranda | 540001818
Nestor Uriarte| 540000817
MAT300-SP23
Project 2

Cubic Splines

Octave Project 2

Description of the problem:

This project's goal is to code routines to construct cubic spline curves in both 2D and 3D. By definition a cubic spline curve is defined by given $n + 1$ points $(t_0, P_0), (t_1, P_1), \dots, (t_n, P_n)$ with $t_i < t_{i+1}$, there is a unique cubic spline $p \in P^n_{3,2}[t_0, t_1, \dots, t_n]$ through the given points satisfying $p''(t_0) = p''(t_n) = 0$. The concept is to select $n + 1$ points in $\mathbb{R}^2$ or $\mathbb{R}^3$, then using a regular mesh $[0, n] \in \mathbb{R}$, finally construct and evaluate a cubic spline whose output will be plotted.

Explanation of the method:

A cubic spline is a piecewise polynomial $p \in P^n_{3,2[t_0,t_1,t_2 \dots t_n]}$. Let

$$p(t) = \begin{cases} p_1(t) & t \in [t_0, t_1) \\ p_2(t) & t \in [t_1, t_2) \\ p_3(t) & t \in [t_2, t_3) \\ & \vdots \\ p_n(t) & t \in [t_{n-1}, t_n) \end{cases}$$

Where $p_i(t) \in \mathbb{P}_3, i = 1, \dots, n$. Moreover we assume that p is continuous with first and second continuous in $[t_0, t_n]$. A basis for the space $P^n_{3,2[t_0,t_1,t_2 \dots t_n]}$

$$B = \{1, t, t^2, t^3, (t - t_1)^3_+, (t - t_2)^3_+, \dots, (t - t_{n-1})^3_+\}$$

Therefore p can be expressed as a linear combination of elements in B in the following way.

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

Given $n + 1$ points $P_0(x_0, y_0), P_1(x_1, y_1), P_y(x_2, y_2), \dots, P_n(x_n, y_n)$ and with $(t_0, P_0), (t_1, P_1), (t_2, P_2), \dots, (t_n, P_n)$ with $t_{i-1} < t_i$, The cubic spline is determined by the solution in $(a_0, a_1, a_2, \dots, a_{n+2})$ of the system of equations

$$\begin{cases} a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 + a_4(t_0 - t_1)^3_+ + a_5(t_0 - t_2)^3_+ + \cdots + a_{n+2}(t_0 - t_{n-1})^3_+ = P_0 \\ a_0 + a_1 t_1 + a_2 t_1^2 + a_3 t_1^3 + a_4(t_1 - t_1)^3_+ + a_5(t_1 - t_2)^3_+ + \cdots + a_{n+2}(t_1 - t_{n-1})^3_+ = P_1 \\ a_0 + a_1 t_2 + a_2 t_2^2 + a_3 t_2^3 + a_4(t_2 - t_1)^3_+ + a_5(t_2 - t_2)^3_+ + \cdots + a_{n+2}(t_2 - t_{n-1})^3_+ = P_2 \\ \vdots \\ a_0 + a_1 t_n + a_2 t_n^2 + a_3 t_n^3 + a_4(t_n - t_1)^3_+ + a_5(t_n - t_2)^3_+ + \cdots + a_{n+2}(t_n - t_{n-1})^3_+ = P_2 \end{cases}$$

This system of equation has $n + 2$ unknowns and n equations, to fore a unique solution we must impose two more conditions $p''(t_0) = p''(t_n) = 0$.

$$p''(t_i) = 2a_2 + 6a_3 t_i + 6a_4(t_i - 1)_+ + 6a_5(t_i - 2)_+ + \cdots + 6a_{n+2}(t - t_{n-1})^3_+$$

Additionally, we have to remove all the shifted elements that do not satisfy the conditions in each of the equations. The final system is.

Daniel Herreros| 540002818
David Miranda | 540001818
Nestor Uriarte| 540000817
MAT300-SP23
Project 2

$$\begin{cases} a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 = P_0 \\ a_0 + a_1 t_1 + a_2 t_1^2 + a_3 t_1^3 + a_4 (t_1 - t_1)_+^3 = P_1 \\ a_0 + a_1 t_2 + a_2 t_2^2 + a_3 t_2^3 + a_4 (t_2 - t_1)_+^3 + a_5 (t_2 - t_2)_+^3 = P_2 \\ \vdots \\ a_0 + a_1 t_n + a_2 t_n^2 + a_3 t_n^3 + a_4 (t_n - t_1)_+^3 + a_5 (t_n - t_2)_+^3 + \cdots + a_{n+2} (t_n - t_{n-1})_+^3 = P_2 \\ 2a_2 + 6a_3 t_0 + 6a_4 (t_0 - 1)_+ + 6a_5 (t_0 - 2)_+ + \cdots + 6a_{n+2} (t_0 - t_{n-1})_+^3 = (0,0) \\ 2a_2 + 6a_3 t_n + 6a_4 (t_n - 1)_+ + 6a_5 (t_n - 2)_+ + \cdots + 6a_{n+2} (t_n - t_{n-1})_+^3 = (0,0) \end{cases}$$

```
% create a zero matrix which will hold the result
% create n + 2 rows and  n + 4 (or 5 if we have z)
% as the solution is given by
% a_0 -> a_(n + 2) and we add two more constrains
% so that is a unique solution
if(Dimension == 3)
  resultMatrix = zeros(n + 2, n + 5);
else
  resultMatrix = zeros(n + 2, n + 4);
endif
% insert the standard basis components
% a0 + a1*t + a2*t^2 + a3*t^3
for(i = 1 : n)
  for(j = 1 : 4)
    resultMatrix(i, j) = t(i)^(j - 1);
  endfor
endfor

% insert the right shifted basis components
for(i = 1 : n)
  for(j = 5 : n + 3)
    resultMatrix(i, j) = max(t(i) - t(j - 3), 0)^3;
  endfor
endfor

% add the additional constrains
% p''(t0) = p''(tn) = 0
% as p''(t0) is p''(0) we can simplify it as 2*a_2 = 0
resultMatrix(n + 1, 3) = 2;

% to simplify p''(tn)
resultMatrix(n + 2, 4) = 6 * t(n);
resultMatrix(n + 2, 3) = 2;
```

Daniel Herreros| 540002818
David Miranda | 540001818
Nestor Uriarte| 540000817
MAT300-SP23
Project 2

We can solve the system of equations by doing a RREF

$$\begin{pmatrix} 1 & t_0 & t_0^2 & t_0^3 & 0 & 0 & \cdots & 0 & x_0 & y_0 \\ 1 & t_1 & t_1^2 & t_1^3 & (t_1-t_1)^3 & 0 & & 0 & x_1 & y_1 \\ 1 & t_2 & t_2^2 & t_2^3 & (t_2-t_1)^3 & (t_2-t_2)^3 & & 0 & x_2 & y_2 \\ \vdots & & & & & & \ddots & & & \vdots \\ 1 & t_n & t_n^2 & t_n^3 & (t_n-t_1)^3 & (t_n-t_2)^3 & & (t_n-t_{n-1})^3 & x_n & y_n \\ 0 & 0 & 2 & 6t_0 & 0 & 0 & & 0 & 0 & 0 \\ 0 & 0 & 2 & 6t_n & 6(t_n-t_1) & 6(t_n-t_2) & \cdots & 6(t_n-t_{n-1}) & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & x'_0 & y'_0 \\ 0 & 1 & 0 & 0 & 0 & 0 & & 0 & x'_1 & y'_1 \\ 0 & 0 & 1 & 0 & 0 & 0 & & 0 & x'_2 & y'_2 \\ \vdots & & & & & & \ddots & & & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & & 0 & x'_n & y'_n \\ 0 & 0 & 0 & 0 & 0 & 0 & & 0 & x'_{n+1} & y'_{n+1} \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 & x'_{n+2} & y'_{n+1} \end{pmatrix}$$

After the RREF the values we get in the last two columns will be the values for $a_0, a_1, a_2, \ldots, a_{n+2}$, the value of the x' column represent the values of *a* for $p(x)$ and the values of y' column the ones of $p(x)$.

```
for(j = 5 : n + 2)
  resultMatrix(n + 2, j) = (t(n) - t(j - 3))*6;
endfor

resultMatrix(1:n, n + 3) = PX';
resultMatrix(1:n, n + 4) = PY';
if(Dimension == 3)
  resultMatrix(1:n, n + 5) = PZ';
endif

resultMatrix = rref(resultMatrix);

xCoef = resultMatrix(:, n + 3)';
yCoef = resultMatrix(:, n + 4)';

if(Dimension == 3)
  zCoef = resultMatrix(:, n + 5)';
endif
```

Daniel Herreros| 540002818
David Miranda | 540001818
Nestor Uriarte| 540000817
MAT300-SP23
Project 2

Examples:

Cubic spline though $(0,1),(1,3),(2,-1),(4,0)$

With a regular mesh $[0,1,2,3]$

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

$$\begin{aligned}p_i\,(t_i)&=a_0+a_1t_i+a_2t_i^2+a_3t_i^3+a_4(t_i-1)_+^3+a_5(t_i-2)_+^3\\ p_i{}'(t_i)&=a_1+2a_2t_i+3a_3t_i^2+3a_4(t_i-1)_+^2+3a_5(t_i-2)_+^2\\ p_i''(t_i)&=2a_2+6a_3t_i+6a_4(t_i-1)_++6a_5(t_i-2)_+\end{aligned}$$

$$\begin{cases}p_0(t_0)=a_0+a_1t_0+a_2t_0^2+a_3t_0^3+a_4(t_0-1)_+^3+a_5(t_0-2)_+^3\\ p_1(t_1)=a_0+a_1t_1+a_2t_1^2+a_3t_1^3+a_4(t_1-1)_+^3+a_5(t_1-2)_+^3\\ p_2(t_2)=a_0+a_1t_2+a_2t_2^2+a_3t_2^3+a_4(t_2-1)_+^3+a_5(t_2-2)_+^3\\ p_3(t_3)=a_0+a_1t_3+a_2t_3^2+a_3t_3^3+a_4(t_3-1)_+^3+a_5(t_3-2)_+^3\\ p_i''(t_0)=2a_2+6a_3t_0+6a_4(t_0-1)_++6a_5(t_0-2)_+\\ p_i''(t_3)=2a_2+6a_3t_3+6a_4(t_3-1)_++6a_5(t_3-2)_+\end{cases}$$

$$\begin{cases}a_0=(0,1)\\ a_0+a_1+a_2+a_3=(1,3)\\ a_0+2a_1+4a_2+8a_3+a_4=(2,-1)\\ a_0+3a_1+9a_2+27a_3+8a_4+a_5=(4,0)\\ 2a_2=(0,0)\\ 2a_2+18a_3+12a_4+6a_5=(0,0)\end{cases}$$

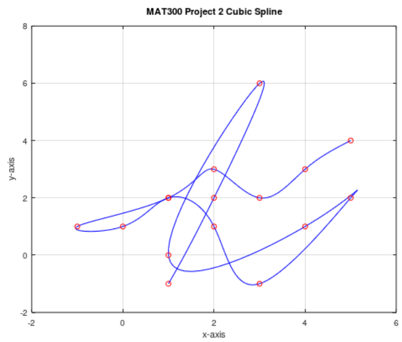
$$\begin{pmatrix}1&0&0&0&0&0&0&1\\1&1&1&1&0&0&1&3\\1&2&4&8&1&0&2&-1\\1&3&9&27&8&1&4&0\\0&0&2&0&0&0&0&0\\0&0&2&18&12&6&0&0\end{pmatrix}\overset{RREF}{\Rightarrow}\begin{pmatrix}1&0&0&0&0&0&0&1\\0&1&0&0&0&0&\frac{16}{15}&\frac{59}{15}\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&-\frac{1}{15}&-\frac{29}{15}\\0&0&0&0&1&0&\frac{2}{5}&\frac{28}{5}\\0&0&0&0&0&1&-\frac{3}{5}&-\frac{27}{5}\end{pmatrix}$$

Now the polynomial for x would be the values for $a_{0\ldots5}$ from the first columns and the one for y the values from the second column.

$$\begin{aligned}p(x)&=\frac{16}{15}x-\frac{1}{15}x^3+\frac{2}{5}(x-1)^3-\frac{3}{5}(x-2)^3\\ p(y)&=1+\frac{59}{15}y-\frac{29}{15}y^3+\frac{28}{5}(y-1)^3-\frac{27}{5}(y-2)^3\end{aligned}$$

Daniel Herreros| 540002818
David Miranda | 540001818
Nestor Uriarte| 540000817
MAT300-SP23
Project 2

Observations:



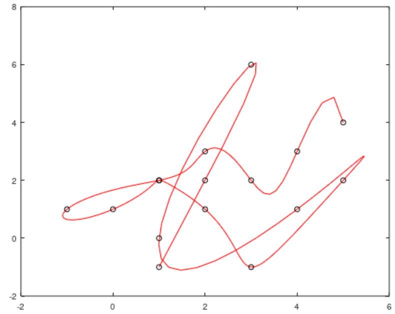


Figure 1: Gauss-Jordan

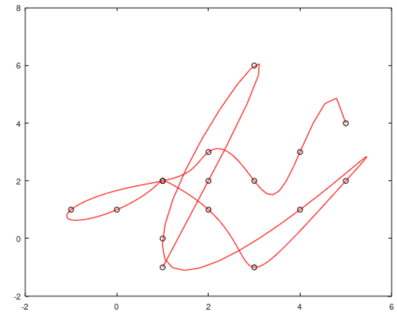


Figure 2: Lagrange Method

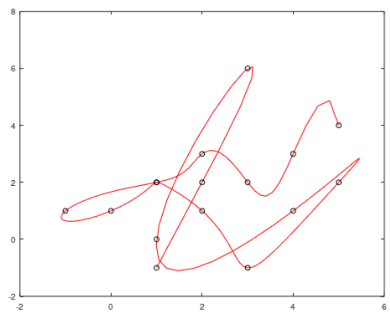


Figure 1: Newton Method

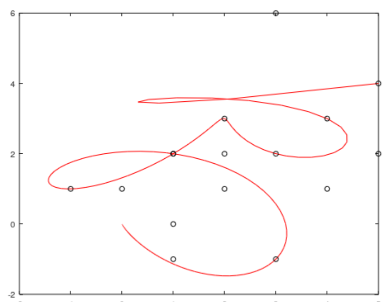


Figure 4: Gauss-Jordan

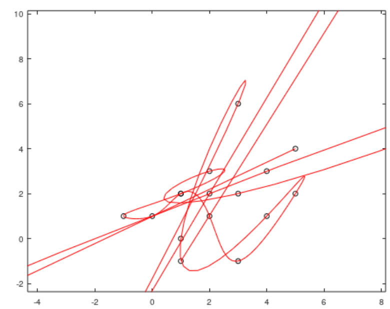


Figure 5: Lagrange

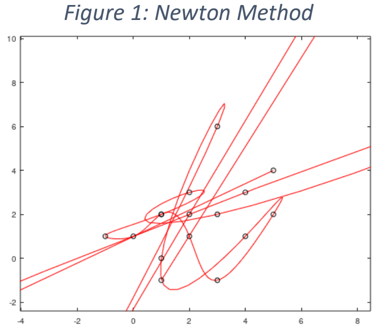


Figure 6: Newton

As we can see in figures 1, 2 and 3, previous methods using Chebyshev meshes still work but they start doing strange things in the endpoints, as expected. However, once we change the mesh to a regular mesh to make a fair comparison with cubic splines, we can see in figures 4, 5 and 6 that Gauss-Jordan completely breaks and the other two methods create very strange curves. Therefore, we can conclude that cubic spline is a much more reliable interpolation method.

Bibliography:

MAT300 Lecture Notes: lecture9.