GEOMETRIC ALGORITHM FOR CURVE INTERPOLATION WITH NON UNIFORM B-SPLINES

Edson Kenji Ueda* Marcos de Sales Guerra Tsuzuki* Emilio Carlos Nelli Silva* Thiago de Castro Martins* Giulliano Batelochi Gallo† Marco Antonio Marques† Rogério Yugo Takimoto*

*Laboratório de Geometria Computacional
Escola Politécnica da Universidade de São Paulo
São Paulo, São Paulo, Brazil

†*Companhia Energética do Estado de São Paulo*

Email: ueda.edson@gmail.com, mtsuzuki@usp.br, ecnsilva@usp.br, thiago@usp.br, giulliano.gallo@cesp.com.br, marco.marques@cesp.com.br, takimotoyugo@gmail.com

Abstract— The interpolation of a sequence of points is an important task in Engineering. In this work, three different interpolation methods are studied and expanded. The first method is the conventional interpolation Spline. The second method is a subdivision based geometric algorithm. The third method interpolates a given set of points with additional point normal constraints. The last two methods were implemented with uniform B-Splines curves. In this work, both methods are expanded to use non uniform B-Spline curves. Three critical curves are used to test the developed methods: circle involute, bowditch and epitrochoid. The results show that the non uniform B-Spline implementations have better quality with smaller errors, once the value of the distance error of the curve is in the order of 10^{-15} % of the bound box diagonal of the initial input data points and the normal error is around 10^{-4} rad in the worst case.

Keywords— Non Uniform B-Spline, Interpolation, Curves, Geometric Algorithm, Point Normal Constraint.

1 INTRODUCTION

The interpolation of a sequence of points is an extensive tool used in Engineering, mainly in reverse engineering, image segmentation, robot path planning and others. There are two approaches to fit a curve in a set of data points, one is curve approximation and the other is curve interpolation that is the object of this study. In the first case an approximated curve is generated and it passes near the data points. A way to do it is using optimization methods like Simulated Annealing (Sobrinho et al., 2009) and Genetic Algorithm (Hasegawa et al., 2013). In the interpolation case a curve that passes through all the data points is generated. In this work, apart from passing through all the points, it will also be guaranteed the interpolation of the tangent and the normal of all the points. Maekawa et al. (2007) and Gofuku et al. (2009) proposed a geometric based interpolation algorithm that is expanded to use uniform and non uniform B-Spline.

The paper is organized as the following. In section 2, a review of B-Spline curve is made and two conventional B-Spline interpolations will be presented, the standard point interpolation and the Piecewise cubic Hermite Interpolation. In section 3, the Geometric Algorithm for B-Spline is presented (the point interpolation and the point-normal, point-tangent B-Spline interpolation). In section 4, a modification of the geometric algorithm is presented, in which a non uniform B-Spline is used to interpolate the curves. In section 5, results will be compared and finally in section 6 conclusion will be presented.

2 BASIC CONCEPTS

A review of B-Splines curves and two methods of conventional B-Spline curve interpolation is made.

2.1 Non Uniform B-Spline Curves

B-Spline is a Spline function in which a curve of order k is formed by the joint of several pieces of polynomials of degree $k-1$ with at most C^{k-2} continuity at breakpoints. And these breaking points $t_0 \leq t_1 \leq \ldots \leq t_n$ define a knot vector $\mathbf{T} = (t_0, t_1, \ldots, t_n)$, which determines the parametrization of the basis functions $N_{i,k}(t)$ of a B-Spline. The B-Spline basis function is defined as

$$N_{i,1}(t) = \begin{cases} 1, & \text{for } t_i \leq t < t_{i+1}, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

for $k = 1$, and

$$\begin{aligned} N_{i,k}(t) =& \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) \\ &+ \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t) \end{aligned} \tag{2}$$

for $k > 1$, and $i = 0, 1, \ldots, n-k$.

A B-Spline is defined as a linear combination of control points and B-Spline basis functions $N_{i,k}(t)$ expressed by $\mathbf{r}(t) = \sum_{i=0}^{n} \mathbf{P}_i N_{i,k}(t)$, $t \in [t_{k-1}, t_{n+1}]$. A recursive method to obtain the first and second derivatives of the B-Spline function is given by (Piegl and Tiller, 1997; Patrikalakis and Maekawa, 2001)

$$\dot{\mathbf{r}}(t) = \sum_{i=1}^{n} (k-1) \left(\frac{\mathbf{P}_i - \mathbf{P}_{i-1}}{t_{i+k-1} - t_i} \right) N_{i,k-1}(t),$$

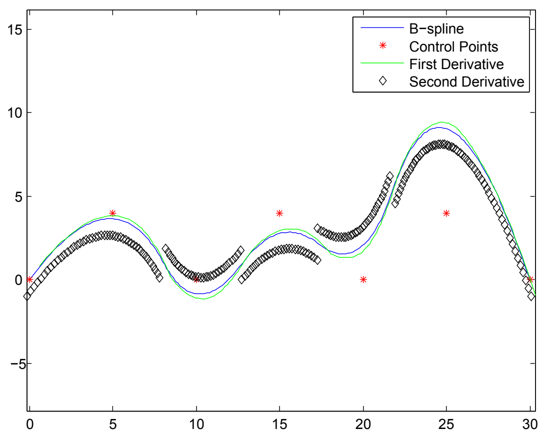


Figure 1: B-Spline curve with first and second derivatives.

$$\ddot{\mathbf{r}}(t) = \sum_{i=2}^{n}\left(\frac{k-2}{t_{i+k-2}-t_i}\frac{k-1}{t_{i+k-1}-t_i}(\mathbf{P}_i-\mathbf{P}_{i-1}) - \frac{k-1}{t_{i+k-2}-t_{i-1}}(\mathbf{P}_i-\mathbf{P}_{i-1})\right)N_{i,k-2}(t).$$

At the endpoints the derivatives is given by

$$\dot{\mathbf{r}}(0) = \frac{(k-1)}{t_k}(\mathbf{P}_1-\mathbf{P}_0), \qquad \dot{\mathbf{r}}(1) = \frac{k-1}{t_{n+k-1}-1}(\mathbf{P}_n-\mathbf{P}_{n-1}), \tag{3}$$

for the first derivative, and

$$\ddot{\mathbf{r}}(0) = \frac{(k-1)(k-2)}{t_k}\left[\frac{\mathbf{P}_0}{t_k} - \frac{t_{k+1}+t_{k+2}}{t_{k+1}.t_{k+2}}\mathbf{P}_1 + \frac{\mathbf{P}_2}{t_{k+2}}\right],$$
$$\ddot{\mathbf{r}}(1) = \frac{(k-2)(k-1)}{1-t_n}\left[\frac{\mathbf{P}_n}{1-t_n} + \frac{\mathbf{P}_{n-2}}{1-t_{n-1}} - \frac{2-t_n-t_{n-1}}{(1-t_n)(1-t_{n-1})}\mathbf{P}_{n-1}\right], \tag{4}$$

for the second derivative. Fig. 1 shows an example of a B-Spline with the first and second derivative.

2.2 Standard point Interpolation Method

Given $n+1$ data points, $\mathbf{Q}_0, \ldots, \mathbf{Q}_n$, it is needed to create a curve that interpolates all the data points. First, it is needed to create a knot vector given by

$$\tau_0 = 0, \qquad \tau_i = \tau_{i-1} + \frac{|\mathbf{Q}_i - \mathbf{Q}_{i-1}|}{d}, \; i = 1, \ldots, n \tag{5}$$

where $d = \sum_{i=1}^{n} |\mathbf{Q}_i - \mathbf{Q}_{i-1}|$.

For sharp curves, Piegl and Tiller (1997) used a centripetal method to create the knot vector, given by

$$\tau_0 = 0, \qquad \tau_i = \tau_{i-1} + \frac{\sqrt{|\mathbf{Q}_i - \mathbf{Q}_{i-1}|}}{d}, \; i = 1, l\ldots, n \tag{6}$$

where $d = \sum_{i=1}^{n} \sqrt{|\mathbf{Q}_i - \mathbf{Q}_{i-1}|}$. If the B-Spline curve is a clamped curve, it is used an average technique to generate the knot vector that is given by the following rules

$$t_0 = \ldots = t_{k-1} = 0,$$
$$t_{r-k+1} = \ldots = t_r = 1, \; r = n+k,$$
$$t_{j+k-1} = \frac{1}{k-1}\sum_{i=j}^{j+k-2}\tau_i, \; j = 1, \ldots, n-k+1.$$

The standard point interpolation method is the solution of the following system of linear equations $\mathbf{r}(\tau_i) = \sum_{j=0}^{n} \mathbf{P}_j N_{j,k}(\tau_i) = \mathbf{Q}_i, \; 0 \le i \le n$. This system has $n+1$ equations with $n+1$ variables, the control points $\mathbf{P}_j$.

2.3 Piecewise Cubic Hermite Interpolation

Piecewise cubic Hermite interpolation can be obtained in the Bézier form by

$$\mathbf{r}(t) = \sum_{j=0}^{3} \mathbf{b}_{3i+j} B_{j,3}\left(\frac{t-\tau_i}{\tau_{i+1}-\tau_i}\right), \; t \in [\tau_i, \tau_{i+1}]$$

where $\mathbf{B}_{j,3}$ is the Bernstein polynomials of degree 3 and $\mathbf{b}_{3i} = \mathbf{P}_i$, $\mathbf{b}_{3i+1} = \mathbf{P}_i + \frac{\tau_{i+1}-\tau_i}{3}\mathbf{d}_i$, $\mathbf{b}_{3i+2} = \mathbf{P}_{i+1} - \frac{\tau_{i+1}-\tau_i}{3}\mathbf{d}_{i+1}$ and $\mathbf{b}_{3i+3} = \mathbf{P}_{i+1}$. From (Prautzsch et al., 2002) $\mathbf{d}_i$ is estimated by

$$\mathbf{d}_i = (1-\alpha_i)\frac{\mathbf{Q}_i - \mathbf{Q}_{i-1}}{\tau_i - \tau_{i-1}} + \alpha_i \frac{\mathbf{Q}_{i+1} - \mathbf{Q}_i}{\tau_{i+1} - \tau_i}$$

with $\alpha_i = \frac{\tau_i - \tau_{i-1}}{\tau_{i+1} - \tau_{i-1}}$. At endpoints, $\mathbf{d}_0$ and $\mathbf{d}_n$ are given by

$$\mathbf{d}_0 = \frac{2(\mathbf{Q}_1 - \mathbf{Q}_0)}{\tau_1 - \tau_0} - \mathbf{d}_1,$$
$$\mathbf{d}_n = \frac{2(\mathbf{Q}_n - \mathbf{Q}_{n-1})}{\tau_n - \tau_{n-1}} - \mathbf{d}_{n-1}.$$

This method uses $3(n+1)$ control points, making this method requiring three times more control points compared with the standard point interpolation.

3 INTERPOLATION BY GEOMETRIC ALGORITHMS

In this section, it is presented the geometric algorithms developed by Maekawa et al. (2007) and Gofuku et al. (2009). The first, interpolates the given data points, and the second also secures the angle of the normal vector in each given data point. It is used the uniform quadratic B-Spline given by the expression $\mathbf{r}(t) = \frac{1}{2}(1-t)^2\mathbf{P}_{i-1} + \frac{1}{2} + t - t^2\mathbf{P}_i + \frac{1}{2}t^2\mathbf{P}_{i+1}, \; i = 1, \ldots, n-1$ to interpolate the points .

3.1 Point Interpolation

The interpolation algorithm introduced by Maekawa et al. (2007) is divided in two steps. First, it is assumed that the given data points $(\mathbf{Q}_0, \ldots, \mathbf{Q}_n)$ are the control points of the B-Spline curve $\mathbf{r}^{(1)}(t) = \sum_{i=0}^{n} \mathbf{P}_i^{(1)} N_{i,k}(t)$, where $\mathbf{P}_i^{(1)} = \mathbf{Q}_i, i = 0, \ldots, n$.

It is apparent that the B-Spline curve does not interpolate all the given data points. For each given point $\mathbf{Q}_i, i = 1, \ldots, n-1$ the closest point in the curve is given by finding the orthogonal projection onto the curve. To compute this point it is necessary to solve the nonlinear equation

$$(\mathbf{Q}_i - (\mathbf{r}^{(1)}(t))) \cdot \dot{\mathbf{r}}^{(1)}(t) = 0. \qquad (7)$$

It is used the Newton's method to solve (7) and, according to Gofuku et al. (2009), in the j-step of Newton's method, the curve could be linearized by projecting $\mathbf{Q}_i$ onto the tangent line of the curve at $t_{i,j}$, determining the update Δt by solving

$$(\mathbf{Q}_i - (\mathbf{r}^{(1)}(t_{i,j}) + \dot{\mathbf{r}}(t_{i,j})\Delta t)) \cdot \dot{\mathbf{r}}^{(1)}(t_{i,j}) = 0.$$

A second order algorithm developed by Hu and Wallner (2005) can also be used to find the projection of the control point. As an initial value for both methods, the values of the knot vector in (5) or (6) are used. The resulting parameter value which satisfy (7) is denoted as $\bar{t}_i$. The second step of the algorithm is to offset each control point $\mathbf{P}_i^{(1)}$ by an error vector $\mathbf{e}_i^{(1)}$ generating new control points $\mathbf{P}_i^{(2)}$. So the control points of the second iteration are given by

$$\mathbf{e}_i^{(1)} = \mathbf{Q}_i - \mathbf{r}^{(1)}(\bar{t}_i),$$

$$\mathbf{P}_i^{(2)} = \mathbf{P}_i^{(1)} + \mathbf{e}_i^{(1)}, \; i = 1, \ldots, n-1.$$

In the α-th iteration, for each given point $\mathbf{Q}_i, i = 1, \ldots, n-1$, it is found its projection onto the B-Spline curve $\mathbf{r}^{(\alpha)}(t)$, defined by its control points $\mathbf{P}_i^{(\alpha)}, i = 0, \ldots, n$, and the knot vector.

$$\begin{aligned}\mathbf{P}_i^{(\alpha+1)} &= \mathbf{P}_i^{(\alpha)} + \mathbf{e}_i^{(\alpha)} \\ &= \mathbf{P}_i^{(\alpha)} + \mathbf{Q}_i - \mathbf{r}^{(\alpha)}(\bar{t}_i), \; i = 1, \ldots, n-1\end{aligned}$$

with $\mathbf{P}_0^{(\alpha)} = \mathbf{Q}_0$ and $\mathbf{P}_n^{(\alpha)} = \mathbf{Q}_n$.

This geometric algorithm repeat this loop until the magnitude of the error becomes smaller than a tolerance. Maekawa et al. (2007) set a tolerance as 0.05% of the bounding box diagonal of the initial input data points.

3.2 Point-Normal Interpolation

The 2D point-normal interpolation problem can be solved from the point-tangent interpolation by taking the cross product of $\mathbf{e}_z = [0 \; 0 \; 1]$ and $\mathbf{t}_i$ generating the normal vector $\mathbf{n}_i = \mathbf{e}_z \times \mathbf{t}_i$, for

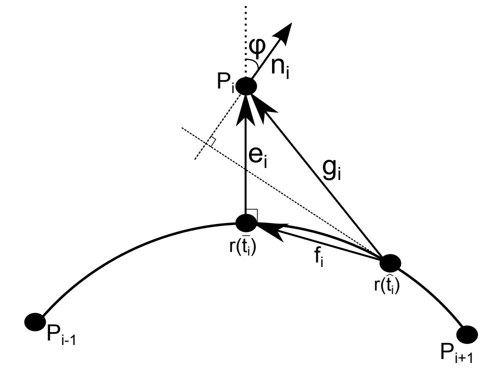


Figure 2: Translation vectors $\mathbf{f}_i$, $\mathbf{g}_i$ and $\mathbf{e}_i$.

$i = 0, \ldots, n$. The point interpolation algorithm introduced the $\mathbf{e}_i$ error vector that translate the control points to make the curve pass through to all given points. For the point-normal problem a new error vector named $\mathbf{f}_i$ is introduced; it translates the control points to correct the angle of the normal of each point in the curve. Gofuku et al. (2009) also introduced a new error vector named $\mathbf{g}_i$ that is the resulting vector of adding $\mathbf{e}_i$ with $\mathbf{f}_i$ (see Fig. 2). This resulting vector translates the control points to make the curve tangent to the given data points and also corrects the angle of the normal, speeding up the algorithm. $\mathbf{f}_i$ and $\mathbf{g}_i$ vectors are given by the following

$$\mathbf{f}_i = \mathbf{r}(\bar{t}_i) - \mathbf{r}(\hat{t}_i), \; i = 1, \ldots, n-1$$

$$\mathbf{g}_i = \mathbf{f}_i + \mathbf{e}_i = \mathbf{Q}_i - \mathbf{r}(\hat{t}_i), \; i = 1, \ldots, n-1.$$

The $\mathbf{r}(\hat{t}_i)$ point is the place in the curve where the tangent of this point is orthogonal to the normal vector $\mathbf{n}_i$. It can be computed by solving the following

$$\mathbf{n}_i \cdot \dot{\mathbf{r}}^{\alpha}(t) = 0. \qquad (8)$$

To solve (8), it is used the Newton's method given by

$$t_i^{\beta+1} = t_i^{\beta} - \frac{\mathbf{n}_i \cdot \dot{\mathbf{r}}^{\alpha}(t_i^{\beta})}{\mathbf{n}_i \cdot \ddot{\mathbf{r}}^{\alpha}(t_i^{\beta})}$$

It is noticed that in straight regions of $\mathbf{r}^{\alpha}(t)$, there are infinite solution making the method unstable. In such cases, these points are not computed in the α-th iteration. Then the point-normal geometric algorithm is updated as following

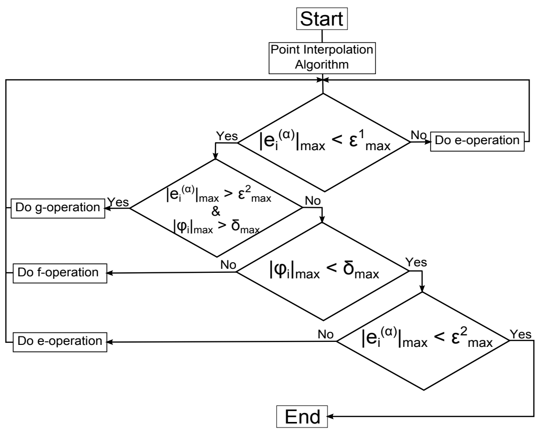
$$\mathbf{P}_i^{(\alpha+1)} = \mathbf{P}_i^{(\alpha)} + \mathbf{r}(\bar{t}_i) - \mathbf{r}(\hat{t}_i), \; i = 1, \ldots, n-1$$

for $\mathbf{f}_i$ error vector, and

$$\mathbf{P}_i^{(\alpha+1)} = \mathbf{P}_i^{(\alpha)} + \mathbf{Q}_i - \mathbf{r}(\hat{t}_i), \; i = 1, \ldots, n-1$$

for $\mathbf{g}_i$ error vector. To control the angle error it is introduced the angle between the normal of the curve ($\mathbf{n}_i$) and the rotation of the tangent vector of the orthogonal projection of the control point at the curve. And it is given by

$$\varphi_i = \arccos\left\{\mathbf{n}_i \cdot \left(\mathbf{e}_z \times \frac{\dot{\mathbf{r}}^{(\alpha)}(\bar{t}_i)}{|\dot{\mathbf{r}}^{(\alpha)}(\bar{t}_i)|}\right)\right\}. \qquad (9)$$



Start
Point Interpolation Algorithm
$|e_i^{(\alpha)}|_{max} < \varepsilon^1_{max}$
Yes
No
Do e-operation
$|e_i^{(\alpha)}|_{max} > \varepsilon^2_{max}$ & $|\varphi_i|_{max} > \delta_{max}$
Do g-operation
$|\varphi_i|_{max} < \delta_{max}$
Do f-operation
$|e_i^{(\alpha)}|_{max} < \varepsilon^2_{max}$
Do e-operation
End

Figure 3: Global iteration of the point-normal interpolation algorithm.

For each iteration the maximum angle is compared to an angle tolerance. Gofuku et al. (2009) named this tolerance as $\delta_{\max}$ and set a tolerance of one hundredth of the maximum angle computed in the standard point interpolation method and also warned that if this tolerance is set too high the algorithm will diverge. The global iteration is illustrated in Fig. 3 and it starts with the point-tangent algorithm by translating the control points by the error vector $\mathbf{e}_i^{(\alpha)}$ until $|\mathbf{e}_i^{(\alpha)}|_{\max} < \varepsilon^1_{\max}$, this $\varepsilon^1_{\max}$ tolerance is the same used at the point-tangent interpolation that is 0.05% of the bounding box diagonal of the given data points. Once this is satisfied, the control points is translated with the $\mathbf{g}_i^{(\alpha)}$ error vector until $|\mathbf{e}_i^{(\alpha)}|_{\max} < \varepsilon^2_{\max}$ and $|\varphi_i|_{\max} < \delta_{\max}$, where $\varepsilon^2_{\max}$ is used as 10^{-12}% of the bounding box diagonal of the given data points. If $|\mathbf{e}_i^{(\alpha)}|_{\max} < \varepsilon^2_{\max}$ is satisfied, but $|\varphi_i|_{\max} < \delta_{\max}$ is not satisfied, the control points are translated by $\mathbf{f}_i^{(\alpha)}$ error vector, and if $|\varphi_i|_{\max} < \delta_{\max}$ is satisfied and $|\mathbf{e}_i^{(\alpha)}|_{\max} < \varepsilon^2_{\max}$ is not satisfied, we translate the control points with $\mathbf{e}_i^{(\alpha)}$. When both conditions are satisfied, the algorithm is finished.

4 EXPANSION TO NON UNIFORM B-SPLINE

Maekawa et al. (2007) and Gofuku et al. (2009) algorithms utilizes uniform B-spline to interpolate a set of data point. However, if the distance of the points is not uniform, it is better to use the equations of the section 2.1 in both algorithms for better results, that is the equations of the definition of B-spline curves and can be used for uniform and noun-uniform B-splines, the distance of the given points do not need to be equidistant. The calculus of the basis function of the B-spline curve have a high cost of processing time and also the majority of the basis functions are null. Piegl and Tiller (1997) utilizes a method to calculate only

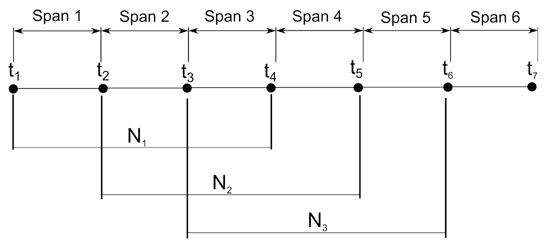
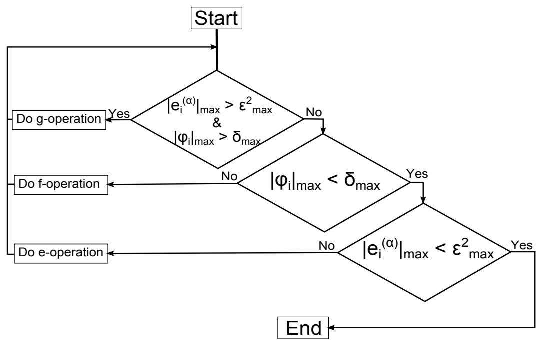


Figure 4: Intervals in which the value of the basis function is not null.



Start
$|e_i^{(\alpha)}|_{max} > \varepsilon^2_{max}$ & $|\varphi_i|_{max} > \delta_{max}$
Yes
No
Do g-operation
$|\varphi_i|_{max} < \delta_{max}$
Do f-operation
$|e_i^{(\alpha)}|_{max} < \varepsilon^2_{max}$
Do e-operation
End

Figure 5: Modified iteration of the point-normal interpolation algorithm.

the basis functions that are not null, this method is a binary search that search in which span of the knot vector $\mathbf{T}$, the value of t is. After finding the span, the basis functions of the same index of the span and $k-1$ previews index are calculated.

As an example for a quadratic B-spline ($k = 3$), in Fig. 4 is possible to see that if the value of t is located in the span 3, the basis function N_1, N_2 and N_3 are not null and all the others basis functions are null. Once the value of the B-spline and the first and second derivatives for a specific value of t is possible to be calculated, these values can be used at Maekawa et al. (2007) algorithm and Gofuku et al. (2009) algorithm in place of the variables $\mathbf{r}(t)$, $\dot{\mathbf{r}}(t)$ and $\ddot{\mathbf{r}}(t)$. The use of a non uniform quadratic B-Spline to interpolate a set of data points that is not equal distanced improved the algorithm, because if the the control points of a B-spline curve is not equidistant, the knot vector $\mathbf{T}$ is not uniform and an uniform B-spline curve can not be used.

Comparing the processing time to calculate the values of the B-spline and the derivative the uniform curve is much faster than the non uniform one, due to this difference of time the algorithm got slower. Because of it, a modification are made to make the algorithm faster. The initial part of Gofuku et al. (2009) algorithm is a first approximation of the interpolative curve to the given points and it is not necessary, once moving the control points with the $\mathbf{g}$ vector not only the curve is approximated but also the correction of the angle of the normal vector is made. Eliminating the first part of the algorithm, the number of iteration

Table 1: Comparing the variation of the $\varepsilon^1_{\max}$ parameter

$\varepsilon^1_{\max}$	time(s)	$\lvert\mathbf{e}_i^{(\alpha)}\rvert_{\max}$	$\lvert\varphi_i\rvert_{\max}$(rad)
1E-6	73.65	6.4E-15	2.9E-08
1E-4	54.22	5.6E-15	2.9E-08
1E-2	46.98	5.6E-15	2.9E-08
1E+0	42.91	6.6E-15	2.9E-08

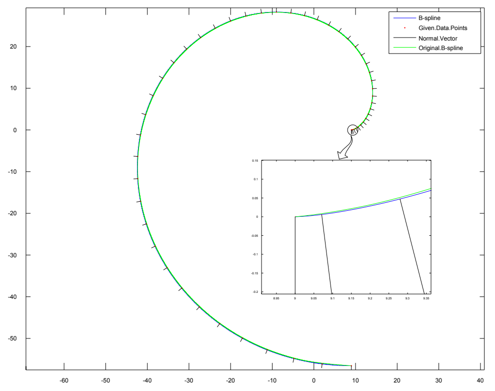


Figure 6: Circle involute curve.

to generate the interpolative curve decrease and it decrease the processing time. It happens because each iteration, in a case of a quadratic B-spline, it is needed to calculate $3n$ values of basis function with n being the number of points.

Table 1 compares the values of time of the execution, the max angle($|\varphi_i|_{\max}$) and distance error($|\mathbf{e}_i^{(\alpha)}|_{\max}$) from 25 points that generated an circumference. The variation of $\varepsilon^1_{\max}$ parameter from $1 * 10^-6$ to 1 shows that the values of max angle($|\varphi_i|_{\max}$) and distance error($|\mathbf{e}_i^{(\alpha)}|_{\max}$) is not interfered and the time of execution decreased. With this results the first part of the algorithm is removed and the final algorithm used in this work is shown is Fig. 5.

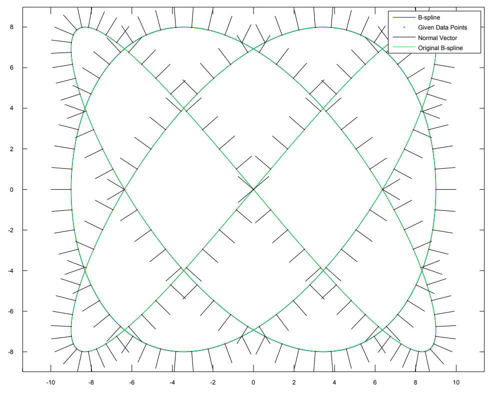


Figure 7: Bowditch curve.

5 RESULTS

In this section, 3 curves are interpolated and compared with the results from (Gofuku et al., 2009).

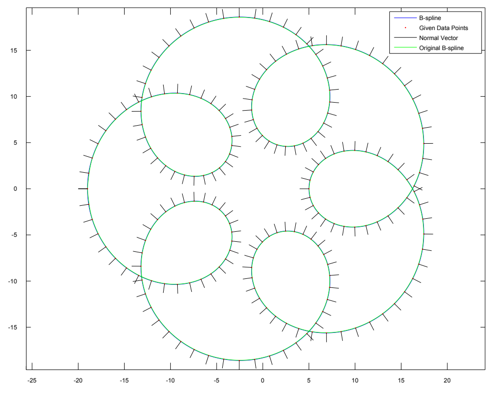


Figure 8: Epitrochoid curve.

The first curve is a circle involute curve given by

$$x(t) = a(cos(t) + t\ sin(t)),$$
$$y(t) = a(sin(t) - t\ cos(t))$$

with $a = 9$, and $0 \leq t \leq 2\pi$. Fig. 6 shows the 51 given data points together with the curve and the unit normal vector.

The second curve is a Bowditch curve given by

$$x(t) = a\ sin(\omega\ t + c),$$
$$y(t) = b\ sin(t)$$

with $a = 9$, $b = 8$, $c = 0$, $\omega = 0.75$ and $-4\pi \leq t \leq 4\pi$. Fig. 7 shows the 201 given data points together with the curve and the unit normal vector.

Finally the third curve is an epitrochoid curve, given by

$$x(t) = (a + b)\ cos(t) - h\ cos(\frac{a+b}{b}t),$$
$$y(t) = (a + b)\ sin(t) - h\ sin(\frac{a+b}{b}t)$$

with $a = 10$, $b = 2$, $h = 7$ and $-\pi \leq t \leq \pi$. Fig. 8 shows the 201 given data points together with the curve and the unit normal vector.

From Table 2 it is possible to see that the results from the uniform quadratic B-Spline was as fast as the results from Gofuku and slower than the conventional method that is so fast that it was not possible to compute the time. However the max angle error was worse, so an implementation using non uniform quadratic B-Spline was made and also compared. The result was better: the max angle error and max distance error got smaller, however the time of execution was much bigger.

To the second example was made a variation of the order of the non uniform B-Spline to evaluate the influence of this parameter over the results. The time of execution, max angle and distance error were presented in Table 3.

The max angle error got smaller with the increasing of the B-Spline order, however the time

Table 2: Results comparing the time of execution, max distance and angle error of the curves

Max distance error				
Curve	Gofuku point-normal	conventional point	Uniform B-Spline	Non Uniform B-Spline
Circle involute	7.8E-15	3.2E-13	4.7E-14	7.2E-15
Bowditch	8.6E-12	5.4E-12	2.2E-14	5.7E-15
Epitrochoid	4.8E-14	1.0E-14	9.27E-15	7.2E-15
Max angle error				
Curve	Gofuku point-normal	conventional point	Uniform B-Spline	Non Uniform B-Spline
Circle involute	2.1E-03	8.2E-01	6.98E-02	2.98E-08
Bowditch	2.4E-02	2.0E+00	1.0E-01	9.56E-04
Epitrochoid	5.0E-03	1.4E-01	2.19E-01	2.98E-08
Time(s)				
Curve	Gofuku point-normal	conventional point	Uniform B-Spline	Non Uniform B-Spline
Circle involute	2.34	*	0.8	557.64
Bowditch	1.72	*	2.4	75.66
Epitrochoid	1.78	*	2.38	1400.75

Table 3: Comparing the order of the B-Spline

Order	time(s)	$\|\mathbf{e}_i^{(\alpha)}\|_{\max}$	$\|\varphi_i\|_{\max}$(rad)
3	75.66	5.7E-15	9.56E-04
4	247.34	5.8E-15	2.9E-04
5	875.09	6.6E-15	1.9E-04

of execution increased a lot. This increase is due to the number times that the B-Spline basis function is calculated and also the curve is hard to control, once each point in the curve is influenced by k numbers of control points, and according to Piegl and Tiller (1997) in a clamped curve, each control point have less influence over the curve, due to the averaging technic to generate the knot vector.

6 CONCLUSION

The algorithms proposed by Maekawa et al. (2007) and Gofuku et al. (2009) were successfully implemented. Both algorithms were expanded to use non uniform B-Splines. Three curves with geometric difficulties (circle involute, bowditch and epitrochoid) were used as tests. The conventional interpolation method, and the uniform and the non uniform B-Spline implementations for the geometric algorithms were compared. The results show that the conventional interpolation method is the fastest algorithm. However, the smallest error was obtained by the non uniform B-Spline implementation.

ACKNOWLEDGEMENTS

EK Ueda is supported by CAPES. MSG Tsuzuki is partially supported by CNPq (grant 309570/2010-7). RY Takimoto is supported by FAPESP (grant 2011/22402-8). This research is supported by ANEEL project PD-0061-0033/2011.

References

Gofuku, S, Tamura, S and Maekawa, T (2009). Point-tangent/point-normal b-spline curve interpolation by geometric algorithms, *CAD* **41**: 412–422.

Hasegawa, AY, Rosso, RU and Tsuzuki, MSG (2013). Bezier curve fitting with a parallel differential evolution algorithm, *11th IFAC Workshop on Intelligent Manufacturing Systems.*, São Paulo, Brazil.

Hu, SM and Wallner, J (2005). A second order algorithm for orthogonal projection onto curves adn surfaces, *CAGD* **22**: 251–260.

Maekawa, T, Matsumoto, Y and Namik, K (2007). Interpolation by geometric algorithm, *CAD* **39**: 313–323.

Patrikalakis, NM and Maekawa, T (2001). *Shape Interrogation for Computer Aided Design and Manufacturing*, Springer-Verlag.

Piegl, L and Tiller, W (1997). *The NURBS book*, Springer-Verlag.

Prautzsch, H, Boehm, W and Paluszny, M (2002). *Bezier and B-spline Techniques*, Springer.

Sobrinho, E, Sanomya, R, Ueda, R, Tiba, H, Tsuzuki, MSG, Adamowski, JC, Silva, ECN, carnobari, RC and Buiochi, F (2009). Development of a methodology for evaluation of a structural damage in turbine blades from hydropower generators, *COBEM 2009.*, Gramado, Brazil.