Abstract

This paper presents the Graphics Constructor 3.0 application. It is an extension version of the Graphics Constructor 2.0 which were elaborated in Selcuk University.

Keywords: Computer Dialogue application, Spline functions.

2010 Mathematics Subject Classification: 65D07

1. Introduction

The Research Center of Applied Mathematics of Selçuk University developed an application called Graphics Constructor 2.0 (GC2.0). The application allows to draw an approximate graph of a function defined by a number of separate fixed points, “knots”. The graph can be calculated as Lagrange interpolation polynomial, a linear, a special quadratic and a cubic Hermite spline functions. The application calculates several values, namely “Integral”, “Curve”, “Area” and “Volume”.

Let \( y = f(x) \), \( a < x < b \) be a continuous real function then the following values

\[
I = \int_a^b f(x) \, dx, \quad L = \int_a^b \sqrt{1 + (f'(x))^2} \, dx, \quad V = \int_a^b f^2(x) \, dx, \quad S = 2\pi \int_a^b |f(x)| \sqrt{1 + (f'(x))^2} \, dx
\]

are known as the integral “Integral”, length of the arc of the graph “Curve”, volume and area of the solid of revolution of the graph rotated around the X-axis “Area” and “Volume”.

The “Integral” is computed for all spline interpolation functions. But another three values are computed only for the first degree spline functions.

In the paper the exact formulas for these values in case of the special quadratic spline functions are given. The exact formula for compute “Volume” for the Hermite cubic spline is also given. A trapezoidal rule is suggested for computing “Curve” and “Area” values for Hermite cubic spline functions. All the results are displayed in the Graphics Constructor 3.0 (GC3.0). The application shows the values as “Curve” (L), “Volume” (V) and “Area” (S).

GC3.0 is implemented in C++ using MFC and compiled in Visual Studio 2013. The user interface of the GC3.0 is very similar to the interface of the GC2.0 with as new natural parameter “0 < M < 65”, which is used as a control value for the third degree spline function application.

Please refer to the full description of the GC2.0 which is given in [1]. A number applications of the GC2.0 one can find in the textbook [2]. Let’s recall some results of the GC2.0 application. The application works with table 1. A unique linear spline function exists

\[
P_i(x) = a_j x + b_j, \quad x_j < x < x_{j+1}, \quad j = 0, 1, \ldots, n - 1
\]

<table>
<thead>
<tr>
<th>( X )</th>
<th>( x_0 )</th>
<th>( x_1 )</th>
<th>\cdots</th>
<th>( x_n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Y )</td>
<td>( y_0 )</td>
<td>( y_1 )</td>
<td>\cdots</td>
<td>( y_n )</td>
</tr>
<tr>
<td>( F )</td>
<td>( f_0 )</td>
<td>( f_1 )</td>
<td>\cdots</td>
<td>( f_n )</td>
</tr>
</tbody>
</table>

Table 1
which is satisfied to table 1. The reals \( a_j, b_j \) are found by solving equations

\[
\begin{align*}
& P_i \left( x_j \right) = y_j, \\
& P_i \left( x_{j+1} \right) = y_{j+1}.
\end{align*}
\]

A unique special quadratic spline function exists

\[
P_j(x) = a_j x^2 + b_j x + c_j, \quad x_j < x < x_{j+1}, \quad j = 0, 1, \ldots, n - 1
\]

which is satisfied to the table 1. The reals \( a_j, b_j, c_j \) are found by solving the equations

\[
\begin{align*}
& P_j(x_0) = y_0, \\
& P_j(x_1) = y_1, \\
& P_j(x_{n+1}) = y_{n+1}.
\end{align*}
\]

A unique Hermite cubic spline function exists

\[
P_j(x) = a_j x^3 + b_j x^2 + c_j x + d_j, \quad x_j < x < x_{j+1}, \quad j = 0, 1, \ldots, n - 1
\]

which is satisfied to the table 1. The reals \( a_j, b_j, c_j, d_j \) are satisfied to the systems

\[
\begin{align*}
P_j(x_j) &= y_j, \\
P_j'(x_j) &= f_j, \\
P_j(x_{j+1}) &= y_{j+1}, \\
P_j'(x_{j+1}) &= f_{j+1}.
\end{align*}
\]

2. Special Quadratic Spline Functions

It is clear that

\[
L_j = \int_{x_j}^{x_{j+1}} \sqrt{1 + (P_j'(x))^2} \, dx = \int_{x_j}^{x_{j+1}} \sqrt{1 + (2a_j x + b_j)^2} \, dx,
\]

guarantee that \( L = L_0 + L_1 + \cdots + L_{n-1} \).

The following algorithm can be used for computing \( L_j \).

**Algorithm 1**

\[
\text{if } |a_j| < 10^{-14} \text{ then } L_j = (x_{j+1} - x_j) \sqrt{1 + b_j^2} \text{ else }
\]

\[
W_j(x) = \frac{1}{4a_j} \left( 2a_j x + b_j \right) \sqrt{1 + (2a_j x + b_j)^2} \\
+ \ln \left( \sqrt{1 + (2a_j x + b_j)^2} + 2a_j x + b_j \right);
\]

\[
L_j = W_j(x_{j+1}) - W_j(x_j); \quad \text{end if.}
\]

It is clear that

\[
V_j = \pi \int_{x_j}^{x_{j+1}} P_j^2(x) \, dx = \pi \int_{x_j}^{x_{j+1}} \left( a_j x^2 + b_j x + c_j \right)^2 \, dx,
\]

guarantee that \( V = V_0 + V_1 + \cdots + V_{n-1} \).

The following formulas can be used for computing \( V_j \).

\[
W(x) = \frac{1}{3} d_j^2 x^4 + \frac{1}{2} a_j b_j x^2 + \frac{1}{3} (2a_j c_j + b_j^2) x^3 + c_j b_j x^2 + c_j^2 x, \quad x_j < x < x_{j+1};
\]

\[
V_j = \pi \left( W(x_{j+1}) - W(x_j) \right).
\]

It is clear that

\[
S_j = \pi \int_{x_j}^{x_{j+1}} |P_j(x)| \sqrt{1 + (P_j'(x))^2} \, dx
= 2 \pi \int_{x_j}^{x_{j+1}} |2a_j x + b_j x + c_j| \sqrt{1 + (2a_j x + b_j)^2} \, dx,
\]

guarantee that \( S = S_0 + S_1 + \cdots + S_{n-1} \).
The following Algorithm 2 can be used for computing $S_j$. This algorithm uses the following functions:

$$
G(u) = u\sqrt{1 + u^2}, \quad H(u) = \ln(1 + u^2),
$$

$$
g(u) = \frac{u}{2u^2 + 1}\sqrt{1 + u^2},
$$

$$
I_j(\alpha, \beta) = \frac{4a_jc_j - b_j^2}{16a_j^2}\left(G(2a_j\beta + b_j) - G(2a_j\alpha + b_j)\right)
+ \frac{4(4a_jc_j - b_j^2)}{64a_j^2}\left(H(2a_j\beta + b_j) - H(2a_j\alpha + b_j)\right)
+ \frac{1}{64a_j^2}\left(g(2a_j\beta + b_j) - g(2a_j\alpha + b_j)\right).
$$

**Algorithm 2**

if $|a_j| < 10^{-14}$ then Applying linear spline function for the points $(x_j, y_j), (x_j + 1, y_{j+1})$ one can compute $S_j$ by the GC2.0 algorithm.

end if.

if $a_j > 0$ then

$$
t_j = \frac{-b_j - s}{2a_j}, \quad T_j = \frac{-b_j + s}{2a_j} \text{ else}
$$

$$
t_j = \frac{-b_j - s}{2a_j}, \quad T_j = \frac{-b_j + s}{2a_j} \quad \text{end if.}
$$

if $t_j < x_j < T_j < x_{j+1}$ then

$$
S_j = 2\pi\left([I_j(x_j, t_j)] + [I_j(T_j, x_{j+1})]\right) \quad \text{end if.}
$$

if $x_j < t_j$ and $T_j < x_{j+1}$ then

$$
S_j = 2\pi\left([I_j(x_j, t_j)] + [I_j(T_j, x_{j+1})]\right) \quad \text{end if.}
$$

if $x_j < t_j < x_{j+1} < T_j$ then

$$
S_j = 2\pi\left([I_j(x_j, t_j)] + [I_j(T_j, x_{j+1})]\right) \quad \text{end if.}
$$

if $x_j < t_j$ and $T_j < x_{j+1}$ then

$$
S_j = 2\pi\left([I_j(x_j, t_j)] + [I_j(T_j, x_{j+1})]\right) \quad \text{end if.}
$$

if $x_j < t_j < x_{j+1} < T_j$ then

$$
S_j = 2\pi\left([I_j(x_j, t_j)] + [I_j(T_j, x_{j+1})]\right) \quad \text{end if.}
$$

end if.

end if.

**3. Hermite Cubic Spline Functions**

It is clear that

$$
V_j = \pi \int_{x_j}^{x_{j+1}} P_3^2(x) dx = \pi \int_{x_j}^{x_{j+1}} \left(ax^3 + bx^2 + cx + d\right)^2 dx,
$$

guarantee that $V = V_0 + V_1 + \cdots + V_{n-1}$. The following formulas can be used for computing $V_j$.

$$
W(x) = \frac{1}{7}a_j^2x^7 + \frac{1}{3}a_jb_jx^6 + \frac{1}{5}(2a_jc_j + b_j^2)x^5
+ \frac{1}{2}(a_jd_j + c_jb_j)x^4 + \frac{1}{3}(2b_jd_j + c_j^2)x^3
+ c_jd_jx^2 + d_j^2x,
$$

$$
V_j = \pi(W(x_{j+1}) - W(x_j)).
$$

It is clear that

$$
L_j = \int_{x_j}^{x_{j+1}} \sqrt{1 + (P_3(x))^2} dx, \quad S_j = 2\pi \int_{x_j}^{x_{j+1}} |P_3(x)|\sqrt{1 + (P_3(x))^2} dx,
$$

guarantee that $L = L_0 + L_1 + \cdots + L_{n-1}$ and $S = S_0 + S_1 + \cdots + S_{n-1}$.

The Algorithm 3 can be used for computing $L_j(M)$ and $S_j(M)$ which approximate $L_j$ and $S_j$ respectively. Here, a natural number $0 < M < 65$ is a control parameter.

**Algorithm 3**

step 1. $h_j = \frac{x_{j+1} - x_j}{M}$,
step 2. for $k = 0, 1, \ldots, M$ \(g_{k,j} = P_j(x_j + kh_j);\)

The approximations “Curve” $L_j(M)$ and “Area” $S_j(M)$ can be computed for the first degree spline function which is defined by the virtual table 2.

<table>
<thead>
<tr>
<th>Vectors</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>\cdots</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X$</td>
<td>$x_j$</td>
<td>$x_j + h_j$</td>
<td>$x_j + 2h_j$</td>
<td>\cdots</td>
<td>$x_{j+1}$</td>
</tr>
<tr>
<td>$Y$</td>
<td>$g_{0,j}$</td>
<td>$g_{1,j}$</td>
<td>$g_{2,j}$</td>
<td>\cdots</td>
<td>$g_{M,j}$</td>
</tr>
</tbody>
</table>

Table 2

Hence

\[
L(M) = L_0(M) + L_1(M) + \cdots + L_{M-1}(M),
\]

\[
S(M) = S_0(M) + S_1(M) + \cdots + S_{M-1}(M)
\]

are the approximations of the required values.

4. Tests

Using GC3.0 application we computed the following two tests for special quadratic spline functions.

<table>
<thead>
<tr>
<th>Vectors</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X$</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>$Y$</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>$F$</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3

<table>
<thead>
<tr>
<th>Vectors</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X$</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>$Y$</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$F$</td>
<td>-3</td>
<td></td>
</tr>
</tbody>
</table>

Table 4

Test 1

According the table 3

“Curve”: 11.0085, “Area”: 185.096 and “Volume”: 45.2389

Test 2

According the table 4


Using GC3.0 we computed the following two tests for Hermite cubic spline functions with $M = 10$ control parameter.

<table>
<thead>
<tr>
<th>Vectors</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X$</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>$Y$</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>$F$</td>
<td>0</td>
<td>3</td>
<td>-6</td>
</tr>
</tbody>
</table>

Table 5

<table>
<thead>
<tr>
<th>Vectors</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X$</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>$Y$</td>
<td>1</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>$F$</td>
<td>1</td>
<td>-1</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 6

Test 3

According the table 5


Test 4

According the table 6

“Curve”: 8.23543, “Area”: 166.63 and “Volume”: 117.585

Conclusion

The presented algorithms are realised in application GC3.0 which is available in the address: www.selcuk.edu.tr/lenematik/Web/Sayfa/Ayrinti/65078/tr. The authors are thankful to Dr. Ayse Bulgak and Dr. Oguzer Sinan for assistance and useful discussions.

References