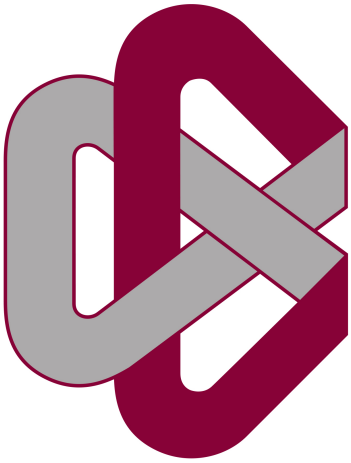
Tarea 10



CIMAT

Nombre:	**Jairo Saul Diaz Soto**
Maestría:	Ciencias Computacionales
Modulo:	Métodos Numéricos
Instructor:	Dr. Luis Daniel Blanco Cocom
Fecha de entrega:	2023 - 10 - 29

1. Introducción

Retomando la importancia de los métodos de interpolación, en ocasiones, no necesariamente se tienen curvas continuas, quizá, aparezcan curvas que solo son continuas a trozos. Para esto existen otro tipo de interpolaciones, las cuales toman secciones e interpolan estas mismas, pero cuidando la congruencia con las demás secciones de la curva que se trata de interpolar. Esto se logra mediante la información que aportan las derivadas de las funciones en determinados puntos.

A continuación algunos métodos que ayudan a resolver problemas mencionados anteriormente

1.1. Polinomios de Hermite

Este método trabaja con diferencias divididas y permite una alta precisión aun con pocos puntos, a continuación un algoritmo que implementa este método

Algorithm 1: Algoritmo de interpolación de Hermite con diferencias dividas

Data: Arreglos x, y, evals, valor x0, entero sz
Result: Estructura solInterpol

```
1  Q ← genMatriz((2 · sz) + 1, (2 · sz) + 1);
2  z ← crearVector((2 · sz) + 1);
3  for i ← 0 to sz do
4      z [2 · i] ← x[i];
5      z [(2 · i) + 1] ← x[i];
6      Q [2 · i][0] ← y[i];
7      Q [(2 · i) + 1][0] ← y[i];
8      Q [(2 · i) + 1][1] ← evals[i];
9      if i > 0 then
10         Q [2 · i][1] ← (Q [2 · i][0] - Q [(2 · i) − 1][0]) / (z [2 · i] - z [(2 · i) − 1]);
11 for i ← 1 to (2 · sz) + 1 do
12     for j ← 1 to i do
13         Q [i][j] ← (Q [i][j − 1] - Q [i − 1][j − 1]) / (z [i] - z [i − j]);
14 sol ← crearEstructuraSolInterpol();
15 sol.matriz ← Q;
16 sol.sol ← 0.0;
17 for i ← 0 to (2 · sz) + 1 do
18     temp ← 1.0;
19     for j ← 0 to i do
20         temp ← temp · (x0 - z [j]);
21     for j ← 0 to i − 1 do
22         temp ← temp · (x0 - z [j]);
23     sol.sol ← sol.sol + Q [i][i] · temp;
24 return sol;
```

1.2. Interpolación cubica natural

Este método es bastante bueno puesto que genera curvas bastante suavizadas, mantiene la concavidad o convexidad de la misma. A continuación un algoritmo que implementa este método

Algorithm 2: Algoritmo de interpolación cúbica natural

```
   Data: Arreglos x, y, valor x0, entero sz
 1 S ← genMatriz(sz, 4) ;
 2 h ← crearVector(sz-1) ;
 3 for i ← 0 to sz do
 4 |   S [i][0] ← y[i];
 5 |   if i < sz − 1 then
 6 |   |   h [i] ← x[i + 1] - x[i];
 7 alpha ← crearVector(sz-1) ;
 8 for i ← 1 to sz-1 do
 9 |   alpha [i] ← (3 * (S [i + 1][0] - S [i][0]) / h [i]) - (3 * (S [i][0] - S [i − 1][0]) / h [i − 1]);
10 l ← crearVector(sz) ;
11 z ← crearVector(sz) ;
12 mu ← crearVector(sz) ;
13 l [0] ← 1.0;
14 z [0] ← mu [0] ← 0.0;
15 for i ← 1 to sz-1 do
16 |   l [i] ← (2 * (x[i + 1] - x[i − 1])) - (h [i − 1] * mu [i − 1]);
17 |   mu [i] ← h [i] / l [i];
18 |   z [i] ← (alpha [i] - (h [i − 1] * z [i − 1])) / l [i];
19 l [sz − 1] ← 1.0;
20 z [sz − 1] ← S [sz − 1][2] ← 0.0;
21 for j ← sz − 2 to 0 do
22 |   S [j][2] ← z [j] - (mu [j] * S [j + 1][2]);
23 |   S [j][1] ← ((S [j + 1][0] - S [j][0]) / h [j]) - (h [j] * (S [j + 1][2] + (2 * S [j][2])) / 3);
24 |   S [j][3] ← (S [j + 1][2] - S [j][2]) / (3 * h [j]);
25 i ← 0;
26 while x0 ≥ x[i] do
27 |   i + +;
28 if i > 0 then
29 |   i − −;
30 temp ← S [i][0] + (S [i][1] * (x0 - x[i])) + (S [i][2] * (x0 − x[i])^2) + (S [i][3] * (x0 − x[i])^3);
31 return temp;
```

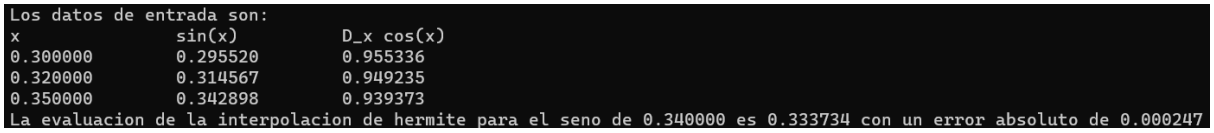


Figura 1: Aproximación de la función seno con 3 puntos de entrada mediante polinonios de Hermite.

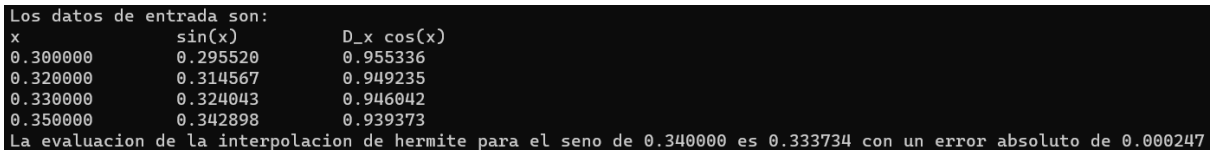


Figura 2: Aproximación de la función seno con 4 puntos de entrada mediante polinonios de Hermite.

1.3. Interpolación cubica fija

Similar a lo anterior, sin embargo, esta permite mayor flexibilidad pues puede interpolar muy bien datos que contenga ruido, permite interpolar curvas que sean continuas a trozos. A continuación un algoritmo que implementa este método.

2. Resultados

A continuación se muestran los resultados obtenidos por los métodos en algunas pruebas

3. Conclusión

Del método de Hermite, se puede apreciar que no cambia en lo absoluto si se hace con 3 o 4 puntos de entrada, puesto que el valor calculado con el método es exactamente igual, lo que nos indica que no es necesario agregar más puntos para mejorar la precisión.

Para el caso de la curva del Pato, se dividio en 3 secciones con los puntos en el eje x 4.4 y 11.6 como los delimitantes de cada región, sin embargo, no se aprecia un cambio radical en las gráficas obtenidas por ambos métodos.

Ahora, para la curva del perro, es inapreciable los puntos donde se unen las secciones que se eligieron para resolver el mismo, además de que es una curva que aproxima de forma bastante fiel a la curva original.

4. Referencias

[1] Burden, R.L., Faires, J.D. and Burden, A.M. 2015. *Numerical analysis.* Cengage learning.
[2] Quarteroni, A., Sacco, R. and Saleri, F. 1999. *Numerical Mathematics*. Springer.

Algorithm 3: Interpolation using Cubic Splines

Data: Arrays x and y, values f_0 and f_n, target value x_0, and array size sz
Result: Interpolated value at x_0

Allocate memory for matrix S of size $sz \times 4$;
Allocate memory for array h of size $sz - 1$;
for $i = 0$ *to* $sz - 1$ **do**
    $S[i][0] \leftarrow y[i]$;
    if $i < sz - 1$ **then**
        $h[i] \leftarrow x[i+1] - x[i]$;
Allocate memory for array *alpha* of size $sz - 1$;
$alpha[0] \leftarrow 3\left(\frac{S[1][0]-S[0][0]}{h[0]} - f_0\right)$;
$alpha[sz-2] \leftarrow 3\left(f_n - \frac{S[sz-1][0]-S[sz-2][0]}{h[sz-2]}\right)$;
for $i = 1$ *to* $sz - 2$ **do**
    $alpha[i] \leftarrow \frac{3(S[i+1][0]-S[i][0])}{h[i]} - \frac{3(S[i][0]-S[i-1][0])}{h[i-1}$;
Allocate memory for arrays l, z, and mu of size sz;
$l[0] \leftarrow 2h[0]$;
$mu[0] \leftarrow 0{,}5$;
$z[0] \leftarrow \frac{alpha[0]}{l[0]}$;
for $i = 1$ *to* $sz - 1$ **do**
    $l[i] \leftarrow 2(x[i+1] - x[i-1]) - h[i-1] \cdot mu[i-1]$;
    $mu[i] \leftarrow \frac{h[i]}{l[i]}$;
    $z[i] \leftarrow \frac{alpha[i]-h[i-1]\cdot z[i-1]}{l[i]}$;
$l[sz-1] \leftarrow h[sz-2](2 - mu[sz-2])$;
$z[sz-1] \leftarrow \frac{alpha[sz-1]-h[sz-2]\cdot z[sz-2]}{l[sz-1]}$;
$S[sz-1][2] \leftarrow z[sz-1]$;
for $j = sz - 2$ ***downto*** 0 **do**
    $S[j][2] \leftarrow z[j] - mu[j] \cdot S[j+1][2]$;
    $S[j][1] \leftarrow \frac{S[j+1][0]-S[j][0]}{h[j]} - \frac{h[j](S[j+1][2]+2S[j][2])}{3}$;
    $S[j][3] \leftarrow \frac{S[j+1][2]-S[j][2]}{3h[j]}$;
$i \leftarrow 0$;
while $x_0 \geq x[i]$ **do**
    $i \leftarrow i + 1$;
if $i > 0$ **then**
    $i \leftarrow i - 1$;
$temp \leftarrow S[i][0] + S[i][1](x_0 - x[i]) + S[i][2](x_0 - x[i])^2 + S[i][3](x_0 - x[i])^3$;
Free memory for arrays l, mu, z, *alpha*, and h;
return *temp*;

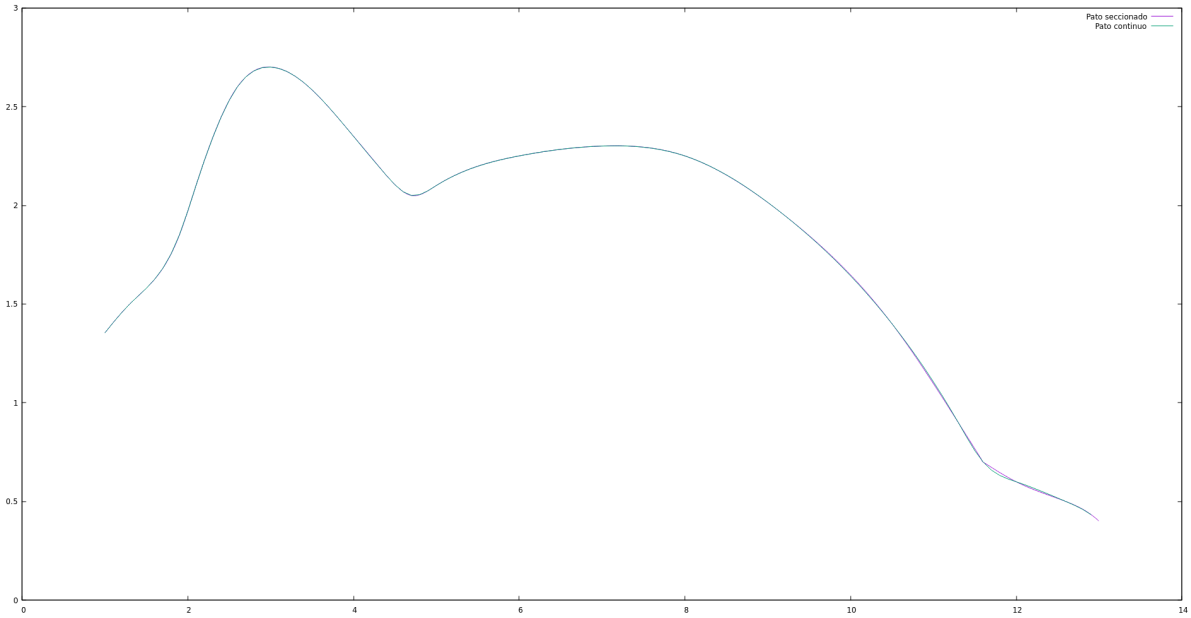


Figura 3: Interpolación de la curva superior de la figura del Pato mediante interpolación cubica natural, la curva morada se genero mediante 3 secciones, mientras que la curva azul se genero con todos los puntosa la vez.

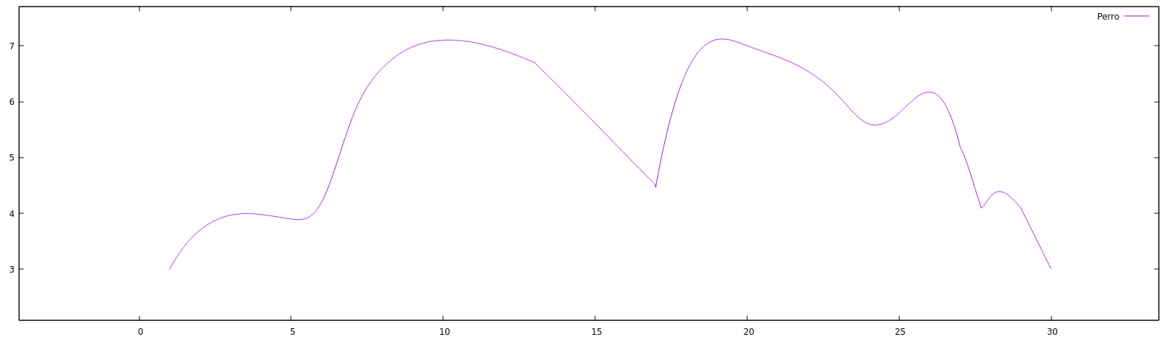


Figura 4: Interpolación de la curva superior de la figura Perro mediante la interpolación cubica fija.