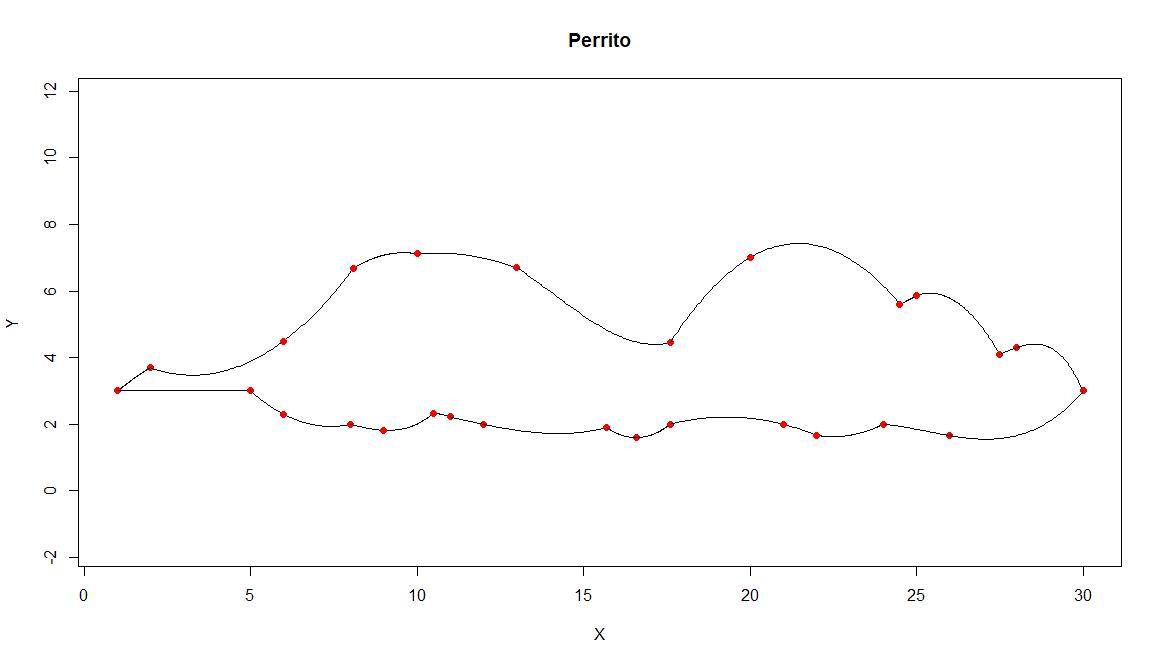
**Pontificia Universidad Javeriana**

**Juan Felipe Vanegas Patiño**

**Diego Alejandro Mateus Cruz**

**Concurso interpolación del perrito**



**Polinomios usados para interpolar el contorno superior**

* 2.1 + x - 0.1\*x^2 para **x = [1,2]**
* 4.95808 - 0.9053864\*x + 0.1381733\*x^2 para **x = [2,8.1]**
* -11.67812 + 3.921165\*x - 0.2041353\*x^2 para **x = [8.1,10]**
* -1.198582 + 1.579441\*x - 0.07475832\*x^2 para **x = [10,13]**
* -58.67878 + 15.22781\*x - 1.132393\*x^2 + 0.02675997\*x^3 para **x = [13,17.6]**
* -84.32407 + 8.547685\*x - 0.1990741\*x^2 para **x = [17.6,24.5]**
* 96.63111 - 7.886\*x + 0.1702222\*x^2 para **x = [24.5,25]**
* -262.43 + 21.132\*x - 0.416\*x^2 para **x = [25,27.5]**
* 3316.42 - 360.6953\*x + 13.0776\*x^2 - 0.1578667\*x^3 para **x = [27.5,30]**

**Polinomios usados para interpolar el contorno inferior**

* 3 para **x = [1,5]**
* 12 - 2.716667\*x + 0.1833333\*x^2 para x = [5,8]
* -18.208 + 8.419333\*x - 1.124222\*x^2 + 0.04844444\*x^3 para **x = [8,10.5]**
* -27.7 + 5.8\*x - 0.28\*x^2 para **x = [10.5,11]**
* -6.303557 + 2.729076\*x - 0.2645088\*x^2 + 0.007895784\*x^3 para **x = [11,15.7]**
* 107.7235 - 12.8\*x + 0.3859649\*x^2 para **x = [15.7,17.6]**
* -25.72 + 2.895\*x - 0.075\*x^2 para **x = [17.6,22]**
* 85.16 - 7.425\*x + 0.165\*x^2 para **x = [22,24]**
* -334.325 + 40.20354\*x - 1.595\*x^2 + 0.02098958\*x^3 para **x = [24,30]**

**Criterios del concurso:**

**1.** Metodología que explique cómo se seleccionaron k puntos con k<n con n el total de puntos dados (Selección de más puntos o de los puntos de la parte de abajo)

Cuando se estaban escogiendo los puntos la idea principal era que cada segmento de curva que uniera los puntos dados, fuera máximo grado 2, además de ello, se omitieron puntos de la tabla original ya que eran muy cercanos y al usar el algoritmo poly.calc() la función retornada pasaba por dicho punto cercano, por lo cual se hizo la selección de los siguientes puntos

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | 1 | 2 | 6 | 8.1 | 10 | 13 | 17.6 | 20 | 24.5 | 25 | 27.5 | 28 | 30 |
| y | 3 | 3.7 | 4.5 | 6.69 | 7.12 | 6.7 | 4.45 | 7 | 5.6 | 5.87 | 4.1 | 4.3 | 3 |

Para la realización del contorno inferior se utilizaron los siguientes puntos, estos fueron obtenidos mediante un dibujo en cuaderno en el cual se dibujaba en el plano cartesiano la figura del perrito y se tenían en cuenta los puntos más relevantes como por ejemplo curvas suaves, cambios de dirección, etc.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | 1 | 5 | 6 | 8 | 9 | 10.5 | 11 | 12 | 15.7 | 16.6 | 17.6 | 21 | 22 | 24 | 26 | 30 |
| y | 3 | 3 | 2.3 | 2 | 1.82 | 2.33 | 2.22 | 2 | 1.9 | 1.6 | 2 | 2 | 1.67 | 2 | 1.66 | 3 |

**2.** Algoritmo que se aplicó (justificación) aplico, por ejemplo, interpolación polinómica y como soluciono el sistema

El algoritmo que se implementó para poder hacer gráficamente el contorno del perrito fue la interpolación polinómica, se guardaban los datos X y Y en pequeños intervalos para obtener (mediante la función poly.calc()) una función que conectara los puntos del intervalo dado, haciendo esto sucesivamente hasta conectar todos los puntos de la tabla original.

**Productos**

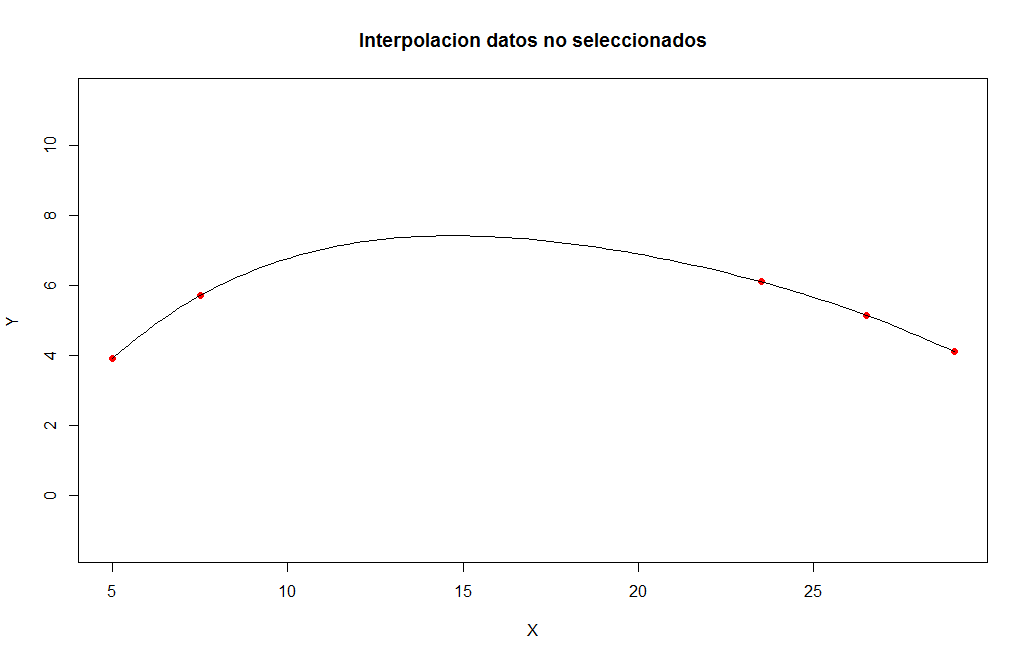
**2. 1)** Codificación, tabla donde está la interpolación en los n-k puntos (no seleccionados), el polinomio o la función interpolante. En un plano los puntos originales, los utilizados, el contorno y el interpolado (utilice el grosor mínimo para la curva)

Tabla de datos no seleccionados

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X | 5 | 7.5 | 23.5 | 26.5 | 29 |
| Y | 3.9 | 5.7 | 6.1 | 5.15 | 4.1 |

Polinomio interpolante de los n-k puntos:

-2.83438 + 1.873047\*x - 0.1211499\*x^2 + 0.003380719\*x^3 - 3.951654e-05\*x^4



**2.2)** Calcular la cota de error de su método con los datos experimentales y compárela con la cota teórica

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X | 5 | 7.5 | 23.5 | 26.5 | 29 |
| Y | 3.9 | 5.7 | 6.1 | 5.15 | 4.1 |

* Con x = 5 y f(x) = - 4.95808 - 0.9053864\*x + 0.1381733\*x^2
  + f(5) = 3.88548
  + cota de error = 0.0142
  + error relativo = 0.00364103
* Con x = 7.5 y f(x) = - -11.67812 + 3.921165\*x - 0.2041353\*x^2
  + f(7.5) = 5.93993
  + cota de error = 0.23993
  + error relativo = 0.04209298
* Con x = 23.5 y f(x) = -84.32407 + 8.547685\*x - 0.1990741\*x^2
  + f(23.5) = 5.315333
  + cota de error = 0.784667
  + error relativo = 0.12863393
* Con x = 26.5 y f(x) = -262.43 + 21.132\*x - 0.416\*x^2
  + f(26.5) = 5.432
  + cota de error = 0.282
  + error relativo = 0.05475728
* Con x = 29 y f(x) = - 3316.42 - 360.6953\*x + 13.0776\*x^2 - 0.1578667\*x^3
  + f(29) = 4.3068
  + cota de error = 0.2068
  + error relativo = 0.05043902

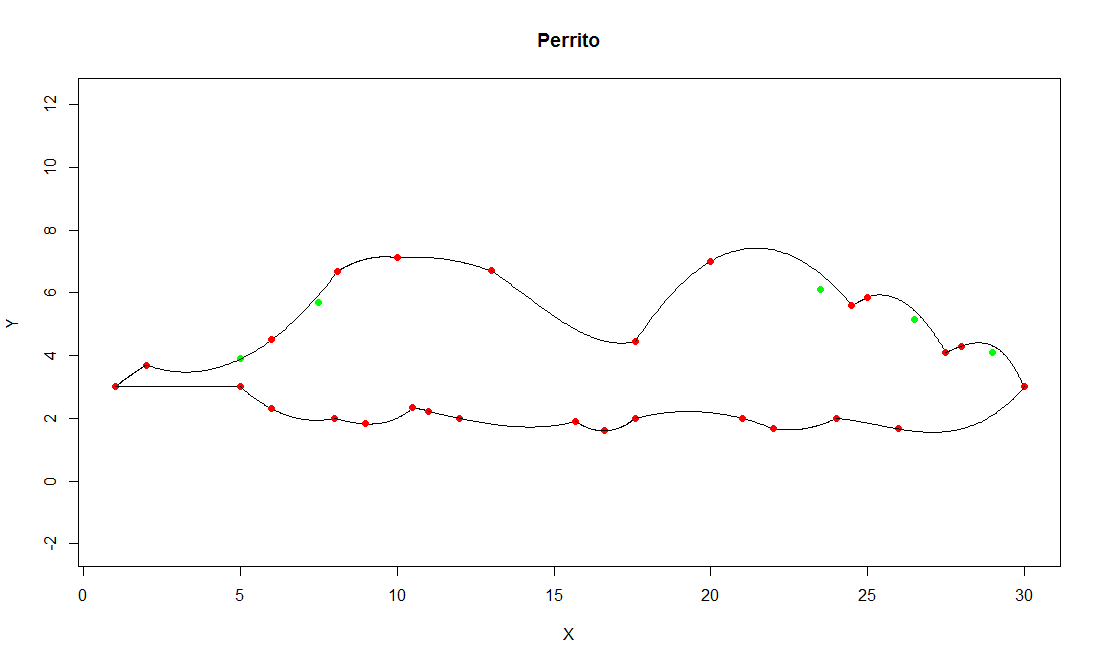
En total

3.1 tabla donde estén los valores interpolados (tenga en cuenta los que no utilizo), los originales y el error relativo, calcule un error relativo total como la suma de los errores relativos

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | 1 | 2 | 5 | 6 | 7.5 | 8.1 | 10 | 13 | 17.6 | 20 | 23.5 | 24.5 | 25 | 26.5 | 27.5 | 29 | 28 | 30 |
| y | 3 | 3.7 | 3.88 | 4.5 | 5.93 | 6.69 | 7.12 | 6.7 | 4.45 | 7 | 5.31 | 5.6 | 5.87 | 5.42 | 4.1 | 4.3 | 4.3 | 3 |

Error relativo total = 0.00364103 + 0.04209298 + 0.12863393 + 0.05475728 + 0.05043902

Error relativo total = 0.27956424

3.2 

**Preguntas**

* **¿El origen se puede modificar?**

Si, el origen puede ser modificado, al cambiar el origen el programa sigue dando como resultado un polinomio interpolante que pase por el origen dado

* **Si se tiene mas información (es decir mas nodos) ¿Cómo se implementarian con el algoritmo usado?**

Al tener nueva información el programa generará una función que cruce por dichos puntos, sin embargo, hay que cambiar los intervalos de interpolación, ya que con nuevos datos las funciones generadas no se conectarían.

* **¿Su método es robusto? En el sentido que si se tienen más puntos ¿la exactitud no disminuye?**

Si se tienen mas puntos habrán mas particiones y por ende mas funciones, si hablamos de exactitud las funciones interpolantes pasarán por todos los puntos lados, pero se convierte un poco tedioso ya que estos datos son cambiados manualmente en codigo.

* **Suponga que tiene más puntos con más cifras significativas ¿como se comporta su algoritmo ? ¿la exactitud decae?**

La exactitud no decae si se hace con particiones que generen polinomios de grado 2 y 3.