UNAM. Facultad de Ciencias
Física Computacional
Profesor: Ernesto Arturo Gutiérrez González
Alumno: Mario Tonatiuh Zamarron

Trabajo Final:
Recreación de una figura por medio de Interpolación de Trazadores Cúbicos Sujetos

RESUMEN

El objetivo de este proyecto fue recrear por computadora una imagen a partir de un número discreto de puntos, aproximándola por medio de Interpolación de Trazadores Cúbicos Sujetos. Para esto se requirió principalmente un editor de texto, un compilador de lenguaje C, un software de graficación y una imagen de silueta bien definida sobre un fondo cuadriculado.

En particular, se utilizó una computadora con el sistema operativo Ubuntu 6.06 (Dapper Drake), el editor *vi*, el compilador GCC y el graficador Gnuplot. Se imprimió la imagen en papel cuadriculado utilizando LogPaper, PDFCreator y Adobe PhotoShop CS1 (en Windows XP).

Básicamente lo que se hizo fue dividir la imagen en secciones, de tal manera que cada una fuera función $f(x)$. De cada sección se tomó una cantidad discreta de puntos y se realizó una interpolación de trazadores cúbicos sujetos mediante un programa en C. En cada sección interpolada se utilizó un polinomio cúbico de la forma:

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$$

para $j = 0, 1, ..., n - 1$ ($n = k - 1$, donde k es el número total de puntos conocidos por sección).

El programa en C generó un archivo de datos con las coordenadas de la función interpolante $S(x)$ -para una sección específica-. Una vez obtenidos los archivos de datos de todas las secciones se graficó la imagen interpolada mediante Gnuplot.

Los resultados se aprecian claramente al comparar la imagen original (figura A) con la interpolada (figura B). La figura C muestra a las figuras A y B sobrepuestas.

Figura A.

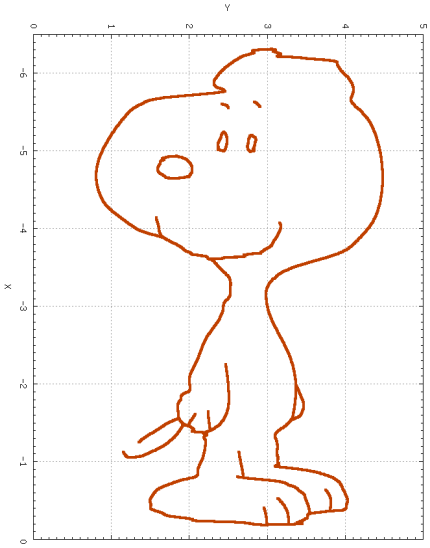


Figura B.

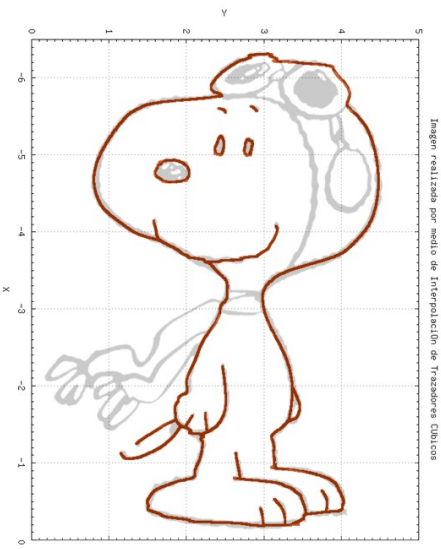


Figura C.

INTRODUCCIÓN

Interpolación

En general, interpolar consiste en encontrar una función, S, escogida dentro de determinado conjunto (por ejemplo, los polinomios) que satisfaga ciertas condiciones locales (por ejemplo, $S(x_i) = y_i$ para determinados puntos (x_i, y_i), $i = 0, 1, 2, ..., n$).

La interpolación resulta útil cuando se disponen de algunos valores, $y_i = f(x_i)$, de una función desconocida, f, y se quiere estimar su valor en puntos intermedios. La interpolación puede aplicarse para simplificar el cálculo de funciones complicadas o para modelar el comportamiento de un sistema a partir de algunos datos experimentales. En ciertos problemas, resulta útil aproximar una función por un polinomio tal que ambos tomen los mismos valores en ciertos puntos concretos. Así, la teoría de la interpolación polinómica se usa para diversos métodos numéricos, como los de la integración o derivación.

Algunas formas de interpolación que se utilizan con frecuencia son la interpolación lineal, la interpolación polinómica, la interpolación por medio de spline o la interpolación polinómica de Hermite. [1]

Lenguaje C

C es un lenguaje de programación de alto nivel, además de ser uno de los lenguajes de programación de propósito general más populares debido en gran medida a su simplicidad, a su consistencia de diseño [2] y a que está ligado uno de los grandes fenómenos de la informática de los años 80: el sistema UNIX, dado que C es el lenguaje nativo de este sistema. [3]

En el mundo de la computación, entre más alejado esté un lenguaje de programación de la arquitectura de la computadora, su nivel será más alto. Los lenguajes de nivel más bajo son los lenguajes de máquina que las computadoras entienden y ejecutan directamente. Por otra parte, los lenguajes de programación de alto nivel se asemejan más al lenguaje humano.

Los lenguajes de programación de alto nivel, incluyendo a C, tienen las siguientes ventajas:

- *Legibilidad.* Los programas son fáciles de leer.
- *Facilidad de mantenimiento.* Es fácil dar mantenimiento a los programas.
- *Portabilidad.* Es fácil portar los programas a través de diferentes plataformas de cómputo.

La legibilidad y facilidad de mantenimiento del lenguaje C se debe precisamente a su semejanza con el lenguaje humano, en especial con el inglés.

Cada lenguaje de alto nivel necesita de un *compilador* o un *intérprete* para traducir las instrucciones escritas en el lenguaje de programación de alto nivel a un lenguaje de máquina que la computadora pueda entender y ejecutar. Cada máquina podría necesitar de un compilador o intérprete distinto para el mismo lenguaje de programación. Por lo tanto, la portabilidad de los programas escritos en C se logra recompilando estos programas con diferentes compiladores para distintas máquinas.

El lenguaje C tiene otras ventajas. Los programas escritos en este lenguaje pueden ser reutilizados. Se pueden guardar los programas de C en un archivo de biblioteca e invocarlos en cualquier otro programa simplemente incluyendo dicho archivo. Muchas tareas de programación comúnes y útiles ya están implementadas en bibliotecas que vienen incluidas con los compiladores. Además, las bibliotecas permiten desencadenar con facilidad el poder y la funcionalidad del sistema operativo que se esté empleando.

C es un lenguaje de programación relativamente pequeño. No hace falta memorizar muchas palabras clave o comandos antes de empezar a escribir programas de C que resuelvan problemas reales. Para quienes buscan velocidad conservando la conveniencia y la elegancia de un lenguaje de alto nivel, C es una de las mejores opciones.

Varios lenguajes de alto nivel han sido desarrollados con base en C. Por ejemplo, Perl es un conocido lenguaje de programación en el diseño de World Wide Web (WWW) a través de internet. Perl tiene muchas características de C. Si uno entiende C, aprender Perl será más fácil. Otro ejemplo es el lenguaje C++, el cual es simplemente una versión ampliada de C, aunque C++ facilita la programación orientada a objetos. Incluso aprender Java es más fácil si ya se conoce C.

Para escribir programas de C en una máquina basada en UNIX, se necesita un editor de texto como *vi* o *emacs*.

Si se tiene una PC con un sistema operativo Windows, se necesitará instalar un compilador de C y un editor de texto. Sin embargo, la mayoría de los compiladores de C vienen con un editor de texto integrado. También se puede utilizar cualquier editor de texto ya instalado en la PC.

En la mayoría de los casos, los paquetes de Linux contienen un compilador de C. Se puede instalar dicho compilador en la computadora al momento de instalar el sistema operativo Linux, o puede agregarse después.

Se puede elegir el compilador de C que se desee, pero se recomienda que sea compatible con el ANSI C (la versión estandarizada de C). [4]

Gnuplot

Gnuplot es una aplicación de libre distribución que permite hacer representaciones gráficas de funciones matemáticas y datos experimentales, en 2 y 3 dimensiones. El programa no tiene interfaz gráfica de usuario sino una línea de comandos que acepta órdenes y produce resultados de forma interactiva.

Además de dibujar la gráfica en pantalla es posible guardarla en multitud de formatos entre los que se encuentran los usuales, como jpg, png, pdf, svg; y otros, menos usuales, pero muy interesantes para los usuarios de LaTeX como metafont, eps, pstricks, picture...

Gnuplot está disponible para una gran variedad de plataformas: estaciones UNIX, Vax/VMS, MS-DOS, Windows y GNU-Linux. Este graficador suele venir en cualquier distribución GNU-Linux, aunque no siempre se instala de forma predeterminada.

En Internet se encuentra abundante documentación sobre este programa. [5]

MARCO TEÓRICO

Interpolación de Trazadores Cúbicos

La aproximación de una función arbitraria por medio de un polinomio en un intervalo cerrado presenta serios inconvenientes debido a la naturaleza oscilatoria de los polinomios de alto grado y a la propiedad de que una fluctuación en una parte pequeña de un intervalo puede ocasionar importantes fluctuaciones en todo el rango.

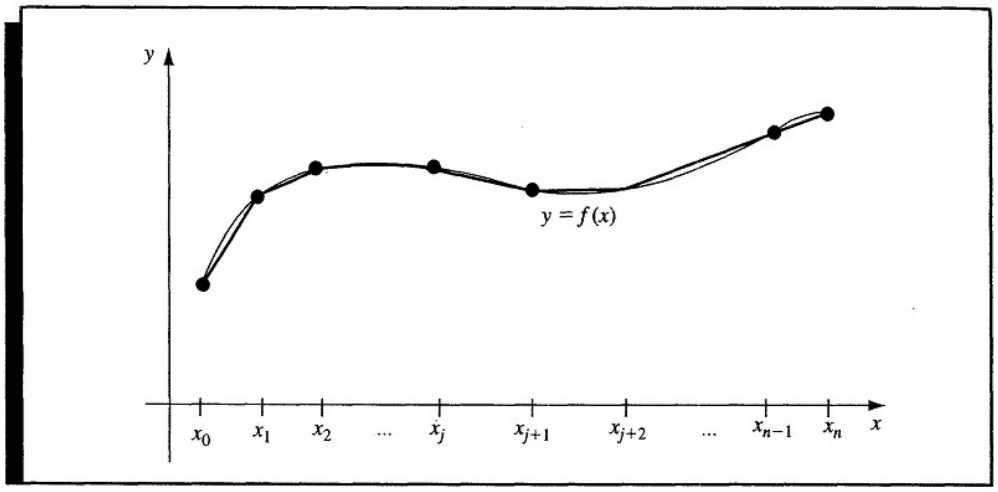
Un procedimiento alterno consiste en dividir el intervalo en una serie de subintervalos, y en cada subintervalo construir un polinomio (generalmente) diferente de aproximación. A esta forma de aproximar por medio de funciones se le conoce como *aproximación polinómica fragmentaria*.

La aproximación polinómica fragmentaria más simple es la interpolación lineal fragmentaria que consiste en unir una serie de puntos de datos

$$\{(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))\}$$

mediante una serie de segmentos de rectas, como los que aparecen en la figura 1.

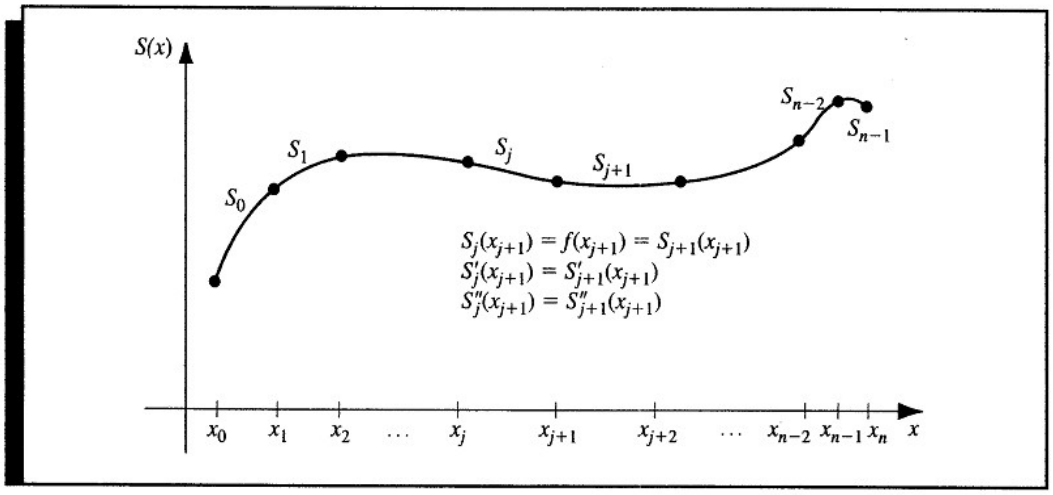
Figura 1



La aproximación por funciones lineales muestra una desventaja: no se tiene la seguridad de que haya diferenciabilidad en los extremos de los subintervalos, lo cual dentro de un contexto geométrico significa que la función interpolante no es "suave" en dichos puntos. A menudo las condiciones físicas indican claramente que se requiere esa condición y que la función aproximante debe ser continuamente diferenciable.

La aproximación polinómica fragmentaria más común utiliza polinomios entre cada par consecutivo de nodos y recibe el nombre de *interpolación de trazadores cúbicos*. Un polinomio cúbico general contiene cuatro constantes; así pues, el procedimiento del trazador cúbico ofrece suficiente flexibilidad para garantizar que el interpolante no sólo sea continuamente diferenciable en el intervalo, sino que además tenga una segunda derivada continua en el intervalo. Sin embargo, en la construcción del trazador cúbico no se supone que las derivadas del interpolante concuerdan con las de la función, ni siquiera en los nodos. (Ver figura 2).

Figura 2



Definición. Dada una función *f* definida en [*a*, *b*] y un conjunto de nodos $a = x_0 < x_1 < \ldots < x_n = b$ un *interpolante de trazador cúbico S* para *f* es una función que cumple con las condiciones siguientes:

- **a.** $S(x)$ es un polinomio cúbico, denotado $S_j(x)$, en el subintervalo $[x_j, x_{j+1}]$ para cada $j = 0, 1, \ldots, n-1$;
- **b.** $S_j(x) = f(x_j)$ para cada $j = 0, 1, \ldots, n$;

c. $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$ para cada $j = 0, 1, \ldots, n-2$;
d. $S'_{j+1}(x_{j+1}) = S'_j(x_{j+1})$ para cada $j = 0, 1, \ldots, n-2$;
e. $S''_{j+1}(x_{j+1}) = S''_j(x_{j+1})$ para cada $j = 0, 1, \ldots, n-2$;
f. Una de las siguientes condiciones de frontera se satisface:
(**i**) $S''(x_0) = S''(x_n) = 0$ (**frontera libre o natural**);
(**ii**) $S'(x_0) = f'(x_0)$ y $S'(x_n) = f'(x_n)$ (**frontera sujeta**).

Aunque los trazadores cúbicos se definen con otras condiciones de frontera, las condiciones dadas en (**f**) son suficientes es este caso. Cuando se presentan las condiciones de frontera libre, el trazador recibe el nombre de *trazador natural* y su gráfica se aproxima a la forma que adoptaría una varilla larga y flexible si se le hiciera pasar por los puntos $\{(x_0, f'(x_0)), (x_1, f'(x_1)), \ldots, (x_n, f'(x_n))\}$.

En términos generales, en las condiciones de frontera sujeta se logran aproximaciones más exactas, ya que abarcan más información acerca de la función. Pero para que se cumpla este tipo de condición de frontera, se requiere tener los valores de la derivada en los extremos o bien una aproximación precisa de ellos.

Si se quiere construir el interpolante del trazador cúbico de determinada función *f*, se aplican las condiciones de la definición a los polinomios cúbicos:

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3,$$

para cada $j = 0, 1, \ldots, n-1$. Está claro que

$$S_j(x) = a_j = f(x_j),$$

Y si se aplica la condición (**c**),

$$a_{j+1} = S_{j+1}(x_{j+1}) = S_j(x_{j+1}) = a_j + b_j(x_{j+1} - x_j) + c_j(x_{j+1} - x_j)^2 + d_j(x_{j+1} - x_j)^3,$$

para cada $j = 0, 1, \ldots, n-2$. Puesto que los términos $x_{j+1} - x_j$ se utilizarán varias veces en este desarrollo, conviene introducir la notación más simple

$$h_j = x_{j+1} - x_j,$$

para cada $j = 0, 1, \ldots, n-1$. Si también se define $a_n = f(x_n)$, entonces la ecuación

$$a_{j+1} = a_j + b_j h_j + c_j h^2_j + d_j h^3_j, \qquad (1)$$

será válida para cada $j = 0, 1, \ldots, n-1$. De manera análoga, se define $b_n = S'(x_n)$ y obsérvese que

$$S'_j(x) = b_j + 2c_j(x - x_j) + 3d_j(x - x_j)^2$$

Significa que $S'_j(x) = b_j$ para cada $j = 0, 1, \ldots, n-1$. Al aplicar la condición (**d**) obtenemos

$$b_{j+1} = b_j + 2c_j h_j + 3d_j h^2_j, \qquad (2)$$

para cada $j = 0, 1, \ldots, n-1$. Al definir $c_n = S''(x_n)/2$ y aplicar la condición (**e**), se obtiene otra relación entre los coeficientes de S_j. En este caso, para cada $j = 0, 1, \ldots, n-1$.

$$c_{j+1} = c_j + 3d_j h_j, \qquad (3)$$

Al despejar d_j en la ecuación (3) y sustituir este valor en las ecuaciones (1) y (2), para cada $j = 0, 1, \ldots, n-1$ se obtienen las ecuaciones

$$a_{j+1} = a_j + b_j h_j + \frac{h^2{}_j}{3}(2c_j + c_{j+1}) \tag{4}$$

y

$$b_{j+1} = b_j + h_j(c_j + c_{j+1}) \tag{5}$$

La relación final que incluye los coeficientes se obtiene resolviendo la ecuación correspondiente en la forma de la ecuación (4), primero para b_j,

$$b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1}), \tag{6}$$

Y entonces, con una reducción del índice, para b_{j-1}:

$$b_{j-1} = \frac{1}{h_{j-1}}(a_j - a_{j-1}) - \frac{h_{j-1}}{3}(2c_{j-1} + c_j).$$

Al sustituir estos valores en la ecuación obtenida de la ecuación (5), con el índice reducido en 1, se obtiene el sistema de ecuaciones lineales

$$h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_j c_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j + a_{j-1}), \tag{7}$$

para cada $j = 0, 1, \ldots, n-1$. Este sistema contiene sólo $\{c_j\}_{j=0}^{n}$ como incógnitas, ya que los valores de $\{h_j\}_{j=0}^{n-1}$ y de $\{a_j\}_{j=0}^{n}$ están dados por el espaciado de los nodos $\{x_j\}_{j=0}^{n}$ y los valores de f en éstos.

Nótese que una vez que se conocen los valores de $\{c_j\}_{j=0}^{n}$ encontrar el resto de las constantes $\{b_j\}_{j=0}^{n-1}$ partiendo de la ecuación (6) y $\{d_j\}_{j=0}^{n-1}$ de la ecuación (3) para construir los polinomios cúbicos $\{S_j(x)\}_{j=0}^{n-1}$ es fácil.

El interrogante principal que se plantea en relación con esta construcción es si se pueden determinar los valores de $\{c_j\}_{j=0}^{n}$ por medio del sistema de ecuaciones dado en (7) y, de ser así, si estos valores son únicos. La respuesta es sí, esto es posible cuando se establece una de las condiciones de frontera de la parte (**f**) de la definición. La demostración del siguiente teorema es específica para la condición (**ii**) de frontera sujeta:

Teorema. Si f está definida en $a = x_0 < x_1 < \ldots < x_n = b$, y es diferenciable en a y b, entonces f tiene un único trazador sujeto que interpola los nodos $x_0, x_1, \ldots, x_n$, es decir, un interpolante de trazador que cumple las condiciones de frontera $S'(a) = f'(a)$ y $S'(b) = f'(b)$.

Demostración. Puesto que $f'(a) = S'(a) = S'(x_0) = b_0$, se puede ver que la ecuación (6) con $j = 0$ implica que

$$f'(a) = \frac{1}{h_0}(a_1 - a_0) - \frac{h_0}{3}(2c_0 + c_1).$$

En consecuencia,

$$2h_0c_0 + h_0c_1 = \frac{3}{h_0}(a_1 - a_0) - 3f'(a).$$

De manera semejante,

$$f'(b) = b_n = b_{n-1} + h_{n-1}(c_{n-1} + c_n),$$

De modo que la ecuación (6) con $j = n - 1$ implica que

$$f'(b)=\frac{a_n-a_{n-1}}{h_{n-1}}-\frac{h_{n-1}}{3}(2c_{n-1}+c_n)+h_{n-1}(c_{n-1}+c_n)$$
$$=\frac{a_n-a_{n-1}}{h_{n-1}}+\frac{h_{n-1}}{3}(c_{n-1}+2c_n),$$

y que

$$h_{n-1}c_{n-1}+2h_{n-1}c_n=3f'(b)-\frac{3}{h_{n-1}}(a_n+a_{n-1}).$$

Las ecuaciones (7) junto con las ecuaciones

$$2h_0c_0+h_0c_1=\frac{3}{h_0}(a_1-a_0)-3f'(a).$$

y

$$h_{n-1}c_{n-1}+2h_{n-1}c_n=3f'(b)-\frac{3}{h_{n-1}}(a_n+a_{n-1}).$$

determinan el sistema lineal Ax = b, donde

$$A=\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & & \vdots \\ 0 & h_1 & 2(h_1+h_2) & h_2 & 0 \\ \vdots & & \ddots & \ddots & \ddots \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{b}=\begin{bmatrix} 0 \\ \frac{3}{h_1}(a_2-a_1)-\frac{3}{h_0}(a_1-a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n-a_{n-1})-\frac{3}{h_{n-2}}(a_{n-1}-a_{n-2}) \\ 0 \end{bmatrix} \quad \text{y} \quad \mathbf{x}=\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix}.$$

La matriz *A* es estrictamente dominante en sentido diagonal, y por tanto, el sistema lineal tiene solución única para $c_0, c_1, \ldots, c_n$. (Ver demostración en Burden, pp. 398 – 400)

La solución del problema de los trazadores cúbicos con condiciones de frontera $S'(x_0) = f'(x_0)$ y $S'(x_n) = f'(x_n)$ se puede obtener usando el algoritmo de la página 11. [6]

DESARROLLO

El primer paso fue eligir una imagen adecuada para recrearla por interpolación. Esta debería tener una silueta bien definida y no demasiados detalles. El Snoopy de la figura 3 es la imagen escogida, pero sin considerar la bufanda ni los detalles del casco.

Figura 3. Imagen escogida para recrearla por medio de interpolación.

Posteriormente, se agregó una cuadrícula a la figura 3 (ver figura 4) para determinar las coordenadas de los puntos entre los cuales se deseaba interpolar.

Para agregar la cuadrícula se decidió utilizar los siguientes programas (en Windows XP): LogPaper, Adobe PhotoShop CS1 y PDFCreator. Con LogPaper se creó una hoja cuadriculada y se guardó en formato pdf por medio de PDFCreator. Después, se abrió con PhotoShop, se le agregó la figura 3, se le añadió color, brillo, contraste, la numeración de los ejes, y se guardó en formato png.

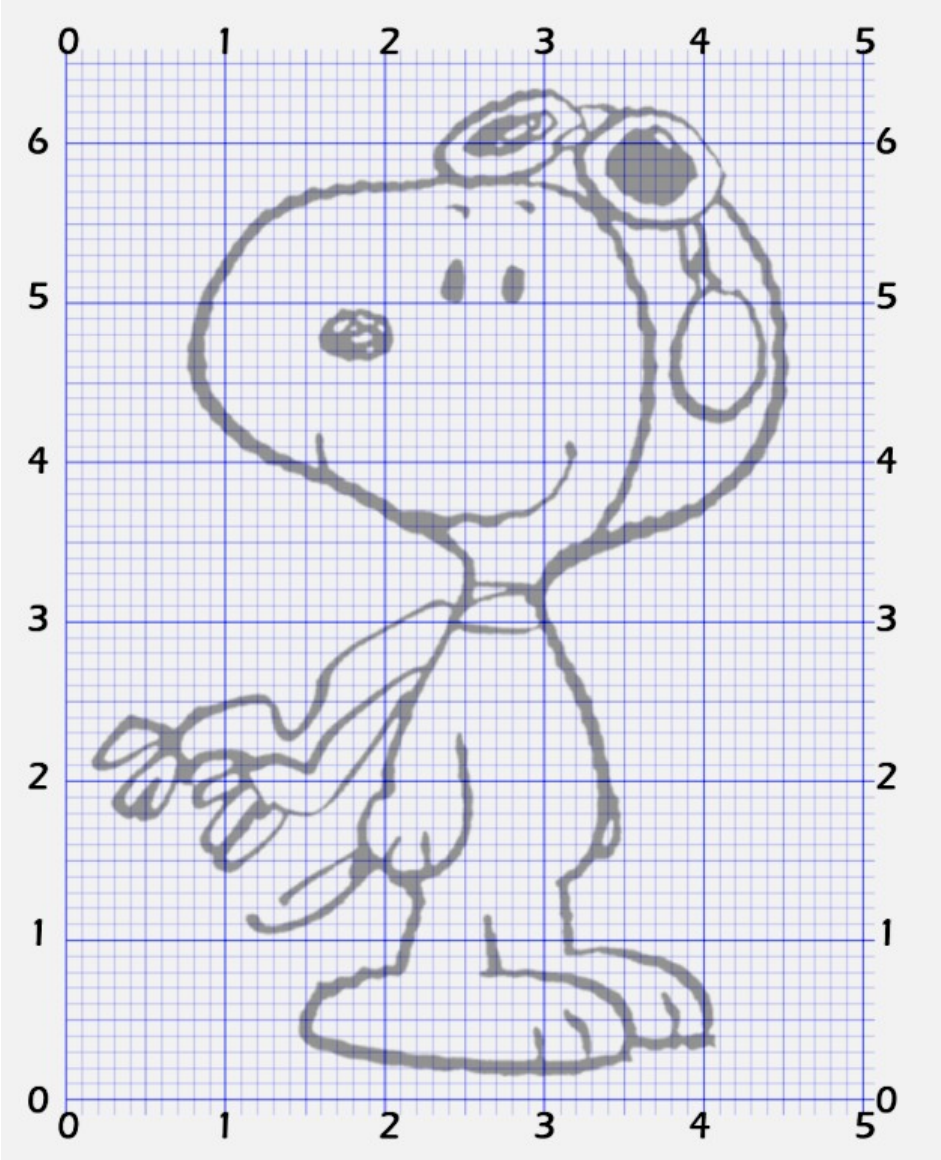


Figura 4. Imagen de la figura 3 sobre fondo cuadriculado.

Para emplear el método de Interpolación de Trazadores Cúbicos Sujetos (MITCS) se requiere que la figura que se espera obtener por este método sea una función $f(x) = y$, es decir, para cada valor de x debe existir y corresponder sólo un valor de y. Esto obligó, más adelante, a "cortar" la figura 3 en diferentes secciones, de manera tal que cada una fuera una función $f(x)$, permitiendo aplicar a cada sección por separado el MITCS.

Para reducir el número de "cortes" en la figura 4 se rotaron los ejes de coordenadas 90° hacia la izquierda (ver figura 5). Con esta rotación el anterior eje X (eje horizontal) pasó a ser el eje Y', y el anterior eje Y (eje vertical) pasó a ser el eje X' (agregando un signo – a todos sus valores para conservar el sentido de la numeración).

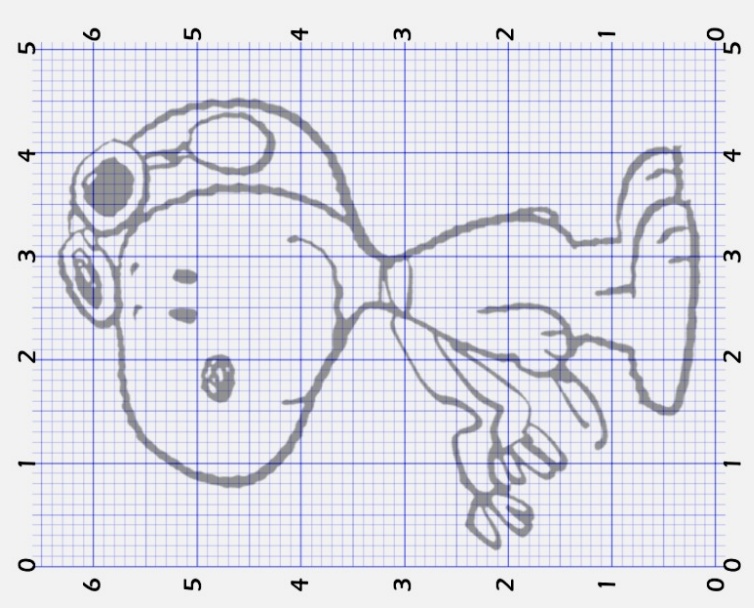


Figura 5. Rotación de 90° de la figura 4 (imagen reducida).

El siguiente paso fue "cortar" la figura 5 en secciones (cada una función $f(x)$) y marcar los puntos entre los que se deseaba interpolar. Hecho esto ya era posible crear un programa para interpolar por medio del MITCS.

El siguiente programa en C se aplica por separado a cada sección de la figura 5. Este genera un archivo de datos para la sección en particular. De aquí que se tengan tantos archivos de datos como secciones de la imagen utilizadas.

Para graficar la imagen completa bastó un script para Gnuplot que reúne todas las secciones de la imagen en una sola.

Descripción del programa en C

En base al siguiente algoritmo, el programa calcula los valores de b_j, c_j y d_j (para $j = 0, 1, ..., n - 1$), correspondientes a los puntos $\{(x_i, y_i)\}_{i=0}^{n}$ (entre los que se desea interpolar) de una sección determinada de la figura 5. Así mismo, con estos datos el programa determina una cantidad adecuada de puntos de la función interpolante $S_j(x)$ para posteriormente graficarla, trazando la interpolación de la sección deseada por medio de Gnuplot.

Datos necesarios (incluidos en el código fuente del programa)

- Coordenadas $\{(x_i, y_i)\}_{i=0}^{n}$ de los puntos entre los que se interpolará la imagen.
- Cantidad $n = k - 1$, donde k es el número total de puntos (recordar que el primer punto es el punto 0).
- Las pendientes de los puntos inicial (*FP0*) y final (*FPN*).

Resultados que muestra el programa

El programa muestra en pantalla los valores de j, x_j, $a_j (= y_j)$, b_j, c_j y d_j. Así mismo, genera un archivo de datos con las coordenadas interpoladas $(x, S_j(x))$ (para $x_j \leq x \leq x_{j+1}$).

Algoritmo para trazar una figura por medio de Interpolantes de Trazadores Cúbicos Sujetos

Los siguientes pasos permiten construir el interpolante de trazador cúbico S para la función f que se define en los números $x_0<x_1<...<x_n$, y que satisface $S'(x_0) = f'(x_0)$ y $S'(x_n) = f'(x_n)$:

ENTRADA n; $x_0, x_1, ..., x_n$; $a_0 = f(x_0), a_1 = f(x_1), ..., a_n = f(x_n)$;
$FP0 = f'(x_0)$; $FPN = f'(x_n)$; (*FP0 y FPN son las pendientes en* $(x_0, f'(x_0))$ *y* $(x_n, f'(x_n))$, *respectivamente, es decir, en los extremos de f*)

SALIDA (a_j, b_j, c_j, d_j) para $j = 0, 1, ..., n - 1$;
$(x, S(x))$.

Paso 1 Para $i = 0, 1, ..., n-1$ tomar $h_i = x_{i+1} - x_i$.

Paso 2 Tomar $\alpha_0 = 3(a_1 - a_0)/h_0 - 3FP0$;
$\alpha_n = 3FPN - 3(a_n - a_{n-1})/h_{n-1}$

Paso 3 Para $i = 0, 1, ..., n - 1$
tomar $\alpha_i = \frac{3}{h_i}(a_{i+1} - a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1})$

Paso 4 Tomar $l_0 = 2h_0$; (*Los pasos 4, 5, 6 y parte del 7 resuelven un sistema lineal tridiagonal utilizando el método de factorización de Crout*)(*Ver Burden, pp. 408*)
$\mu_0 = 0.5$;
$z_0 = \alpha_0 / l_0$.

Paso 5 Para $i = 0, 1, ..., n - 1$
tomar $l_i = 2(x_{i+1} - x_{i-1}) - h_{i-1}\,\mu_{i-1}$;
$\mu_i = h_i / l_i$;
$z_i = (\alpha_i - h_{i-1}\,z_{i-1}) / l_i$;

Paso 6 Tomar $l_n = h_{n-1}(2 - \mu_{i-1})$;
$z_n = (\alpha_n - h_{n-1}\,z_{n-1}) / l_n$;
$c_n = z_n$.

Paso 7 Para $j = n - 1, n - 2, ..., 0$
tomar $c_j = z_j - \mu_j\,c_{j+1}$;
$b_j = (a_{j+1} - a_j)h_j - h_j(c_{j+1} + 2c_j)/3$;
$d_j = (c_{j+1} - c_j)/(3h_j)$.

Paso 8 Para $j = 0, 1, ..., n - 1$ (*Este paso permite obtener una cantidad de puntos de* $(x, S(x))$ *proporcional a la distancia entre dos puntos consecutivos conocidos*)
tomar $t = 1$;
Mientras $\frac{1}{t}\sqrt{(x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2} > 0.01$:
$\{t = t + 1\}$. (*0.01 es 1/10 de la mínima escala empleada y es la distancia deseada entre puntos consecutivos a interpolar*)
intervalo $= (x_{j+1} - x_j) / t$; (*"intervalo" es la distancia entre cada valor de x a interpolar dentro del intervalo* $[x_j, x_{j+1}]$)
$x = x_j$;
Mientras $x \le x_{j+1}$:
$\{ S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$; (*Inicio de la interpolación*)
$x = x +$ *intervalo*$\}$.

Paso 9 SALIDA (a_j, b_j, c_j, d_j) para $j = 0, 1, ..., n - 1$ {En pantalla};
$(x, S(x))$ {En archivo}. (*Nota:* $S(x) = S_j(x)$ *para* $x_j \le x \le x_{j+1}$)
PARAR.

Código fuente en C

```
/*  Trazo de un dedo del brazo derecho de Snoopy por medio de Interpolación de Trazado-
res Cúbicos Sujetos.

Para compilar hay que incluir al final la instrucciOn -lm
*/

#include<stdio.h>
#include<math.h>

double distancia(float a, float b); /* FunciOn para calcular la distancia
                                       entre dos puntos */

main()
{

  float h[100], FPO=-0.67, FPN=0.118, alfa[100],l[100],m[100];
  float z[100],a[100],c[100],b[100],d[100],S[100],x, t, intervalo;
  int n=3, j, i;

  /* ENTRADA VALORES: Coordenadas (X,Y) */
  float X[100]={-1.62,-1.52,-1.49,-1.46};
  float Y[100]={2.08,2.01,2.00,2.01};


  /* Crear y abrir archivo */
  FILE *Resultados;
  Resultados = fopen("snoopy[3.5].dat", "w");


  for(i=0;i<=n;i++)
  {
   a[i] = Y[i];
  }

  /* Paso 1 */
  for(i=0;i<=n-1;i++)
  {
   h[i]=X[i+1]-X[i];

   /* Paso 2 */
   alfa[0]=3*(Y[1]-Y[0])/h[0]-3*FPO;
   alfa[n]=3*FPN-3*(Y[n]-Y[n-1])/h[n-1];
  }

  /* Paso 4 */
  l[0]=2*h[0];
  m[0]=0.5;
  z[0]=alfa[0]/l[0];

  /* Paso 3 */
  for(i=1;i<=n-1;i++)
  {
   alfa[i]=(Y[i+1]-Y[i])*3/h[i] - (Y[i]-Y[i-1])*3/h[i-1];

   /* Paso 5 */
   l[i]=2*(X[i+1]-X[i-1])-h[i-1]*m[i-1];
   m[i]=h[i]/l[i];
   z[i]=(alfa[i]-h[i-1]*z[i-1])/l[i];
  }

  /* Paso 6 */
  l[n]=h[n-1]*(2-m[n-1]);
```

```
  z[n]=(alfa[n]-h[n-1]*z[n-1])/l[n];
  c[n]=z[n];

  /* Paso 7 */
  for(j=n-1;j>=0;j--)
  {
   c[j]=z[j]-m[j]*c[j+1];
   b[j]=(Y[j+1]-Y[j])/h[j]-h[j]*(c[j+1]+2*c[j])/3;
   d[j]=(c[j+1]-c[j])/(3*h[j]);
  }

  fprintf(Resultados,"\n# Datos nUm. 1\n#     x  \t S(x)\n"); /* TItulo del 1er bloque
                                          de datos en archivo, corresponde a (x,S(x)) */

  /* Paso 8 */ /* CreaciOn de Sj(x) */
  for(j=0;j<=n-1;j++)
  {          /* Procedimiento para calcular un nUmero de intervalos proporcional
              a la distancia entre X[j] y X[j+1]*/
    t = 1;

    while( ( distancia( X[j+1]-X[j], Y[j+1]-Y[j] )/t ) > 0.01)
                                    /* 0.01 es la distancia deseada entre cada punto a
                                    interpolar */
      {
       t = t + 1;    /* t es el nUmero de trozos en que se divide la distancia entre
                     X[j] y X[j+1] */
      }

    intervalo = (X[j+1]-X[j]) / t; /* "intervalo" es la distancia entre cada valor
                                de x a interpolar dentro del intervalo [X[j],X[j+1]] */
    x = X[j];

    while(x <= X[j+1])  /* Inicio de la interpolaciOn de X[j] a X[j+1] */
      {
       S[j] = a[j] + b[j]*(x - X[j]) + c[j]*pow((x - X[j]),2) + d[j]*pow((x - X[j]),3);
       x = x + intervalo;  /* S[j] va tomando valores distintos mientras x aumenta */

  /* Paso 9 SALIDA */
       fprintf(Resultados,"  %.5f \t%.5f\n", x, S[j]);   /* Imprime en archivo:
                                                          (x, S[j](x)) */
      }
  }

  fprintf(Resultados,"\n");
  fprintf(Resultados,"\n# Datos nUm. 2\n#    x \t\t       y\n");   /* TItulo del 2o
                                  bloque de datos en archivo (separado del primero por
                                  dos renglones), corresponde a (x,y) */
  printf("\n j\t    x \t       a \t   b \t        c \t     d \n");
  for(j=0;j<=n-1;j++)
  {
   printf(" %d\t %.3f\t  %.3f\t%.3f\t  %.3f\t%.3f\n",j,X[j],a[j],b[j],c[j],d[j]);
                                  /* Imprime en pantalla x, a, b, c y d, para cada j*/
   fprintf(Resultados,"  %.3f\t     %.3f\n",X[j],Y[j]);    /* Imprime en archivo (x,y)
                                para cada j, que son los puntos conocidos al inicio */
  }
  printf(" %d\t %.3f\t     %.3f\n",j,X[n],a[n]);
  fprintf(Resultados,"  %.3f\t     %.3f\n",X[n],Y[n]);    /* Imprime el Ultimo punto
                                                          (extremo derecho)*/
  printf("\n");  /* fin del paso 9 */


  /* Cerrar Archivo */
  fclose(Resultados);

} /* fin del main */
```

```
double distancia(float a, float b)/* FunciOn que calcula la distancia entre dos puntos:
                                    (x1,y1) y (x2,y2), donde a = x2 -x1, y = y2 - y1 */
{
 return( sqrt(pow(a,2) + pow(b,2)) );
}

/* fin del programa */
```

Tratamiento posterior a la ejecución del programa en C

Una vez obtenidos los archivos de datos de todas las secciones de la figura 5 es posible graficar las funciones interpolantes Sj(x). Para esto se creó un script para Gnuplot.

Básicamente lo que hace el siguiente script es mostrar en pantalla la gráfica y a la vez guardarla en formato png, además de agregar otras opciones como: color y grosor de línea, tamaño de la imagen png, entre otras.

Para ejecutarlo en Ubuntu se escribe en la terminal: `$gnuplot nombredelscript.p`

Script para Gnuplot

```
########################################
# SCRIPT para GNUPLOT, Archivo llamado "snoopy[completo].p" para unir todas las partes
de la imagen.

# Para activar este postscript escribir en la terminal: gnuplot snoopy[completo].p
(despues de tener todos los archivos .dat )
########################################

reset # poniendo en ceros los sets

set grid
set xlabel "X" # poner etiqueta al eje x
set ylabel "Y" # poner etiqueta al eje y
set title "Imagen realizada por medio de InterpolaciOn de Trazadores CUbicos"
                                                    # poner titulo a la grafica

set xrange [-6.5:0] # poner rango eje x [a:b]
set yrange [0:5] # poner rango eje y [a:b]
set xtics 1 # poner graduacion eje x aumentando 1 unidad
set mxtics 10 # poner 10 pequeñas graduaciones entre cada stic
set ytics 1 # poner graduacion eje y aumentando en 1 unidad
set mytics 10 # poner 10 graduaciones entre cada stic

unset key #-0.5,0.5 # posicion del identificador gráfico (deshabilitado)

set style line 1 lt -1 lw 4       #Se define el estilo de línea a utilizar

# DIBUJANDO LA GRAFICA (la figura 5 se dividiO en 25 secciones)
plot "snoopy[1.1].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[1.2].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[1.3].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[1.4].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[1.5].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[1.6].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[1.7].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[1.8].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[1.9].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[2.0].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
```

```
     "snoopy[2.1].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[2.2].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[2.3].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[2.4].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[2.5].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[2.6].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[2.7].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[2.8].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[2.9].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[3.0].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[3.1].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[3.2].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[3.3].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[3.4].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1, \
     "snoopy[3.5].dat" index 0:0 using 1:2 title "S(x)" with lines linestyle 1


# CREANDO UN GRAFICO EN POSTSCRIPT
set terminal png large size 840,697 # gnuplot realiza los ajustes de la salida al tipo
                                    #       png antes de salir
set output "snoopy[completo].png" # La salida al fichero.png
replot # volver a graficar

set output # Marca la salida al fichero actual
set terminal pop # pop: es equivalente a “salvar terminal” y “cargar terminal” pero sin
                 #     tener acceso al sistema de ficheros
pause -1 # mantiene la grafica en pantalla hasta que se presione ‘enter’ en la terminal

reset # Regresando los sets a default

quit

# FIN DEL SCRIPT PARA GNUPLOT
```

La figura 6 muestra las secciones interpoladas a partir de la figura 5. Las secciones están separadas por diferentes colores. Los centros de los pequeños cuadrados negros representan los puntos que fueron utilizados para la interpolación.

La figura 7 es semejante a la figura 6, sólo que en ésta todas las secciones interpoladas se muestran con color negro y los puntos utilizados para la interpolación se indican con puntos rojos.

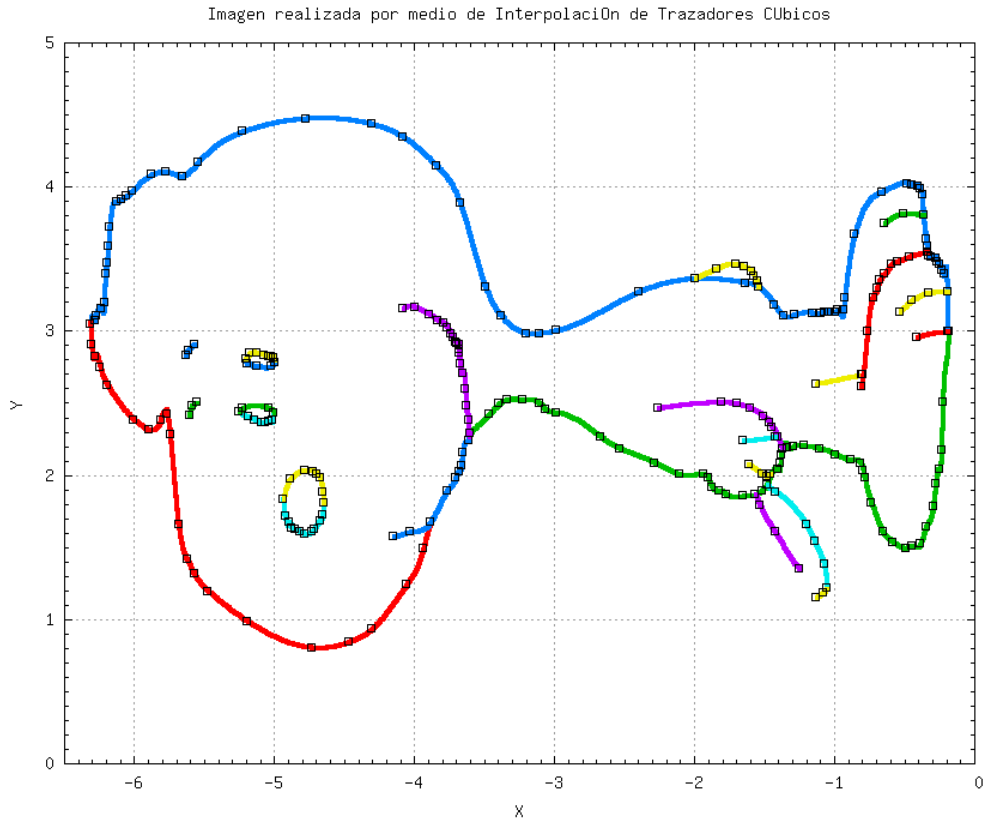


Figura 6. Secciones interpoladas de la figura 5 mostradas en diferentes colores. El centro de los cuadrados pequeños son los puntos utilizados para la interpolación.

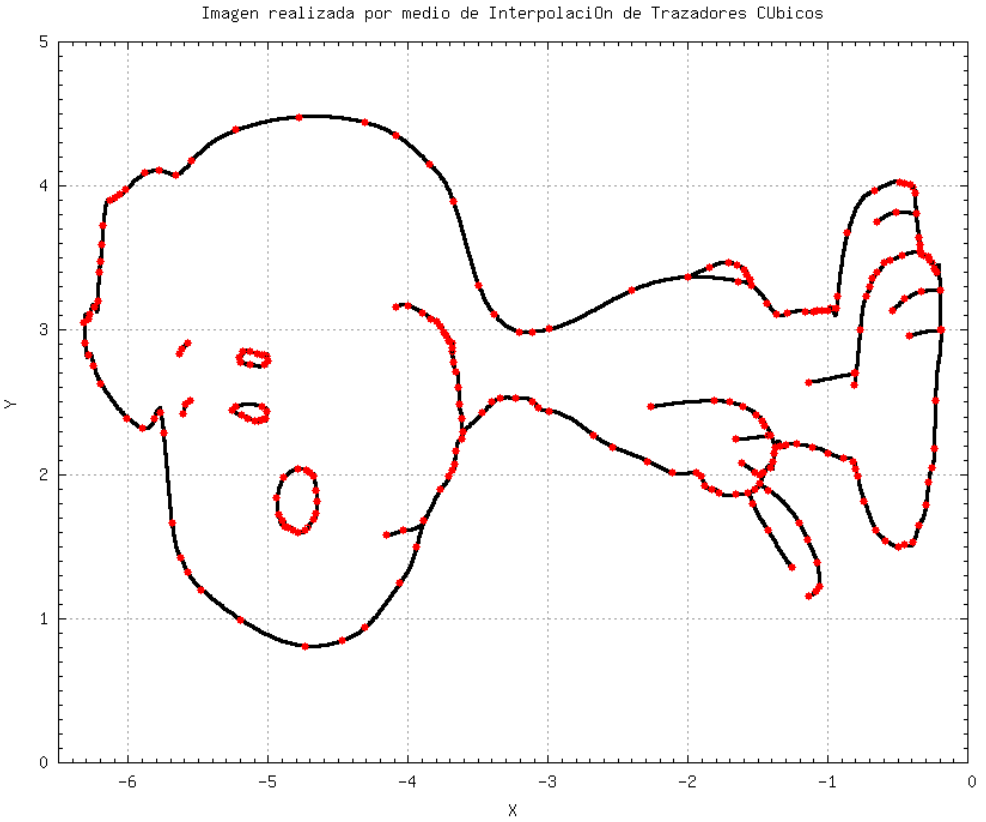


Figura 7. Todas las secciones interpoladas de la figura 5 se muestran en color negro. Los puntos rojos son los puntos utilizados para la interpolación

RESULTADOS

La figura 8 es la gráfica obtenida por el método de Interpolación de Trazadores Cúbicos Sujetos (MITCS) a partir de la figura 5. La figura 9 muestra sobrepuestas a la imagen interpolada con la original.

En cada sección o "corte" interpolado se utilizó un polinomio cúbico de la forma:

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$$

para $j = 0, 1, ..., n - 1$ ($n = k - 1$, donde k es el número total de puntos conocidos por sección). Las constantes a_j, b_j, c_j y d_j son específicas para cada j y para cada sección (ver figura 6).

En la figura 7 se aprecian los puntos entre los cuales se interpoló. Se puede ver que hubieron zonas en las que la distancia entre puntos podía ser mayor que en otras. Se requirieron puntos más cercanos en la mayor parte de las zonas donde hay cambios bruscos de curvatura.

El MITCS brindó buenos resultados al recrear la figura 1, aunque presentó el inconveniente de tener que cortar la imagen en secciones para que cada una fuera función $f(x)$, por lo que tuvo que realizarse un programa para cada seccion.

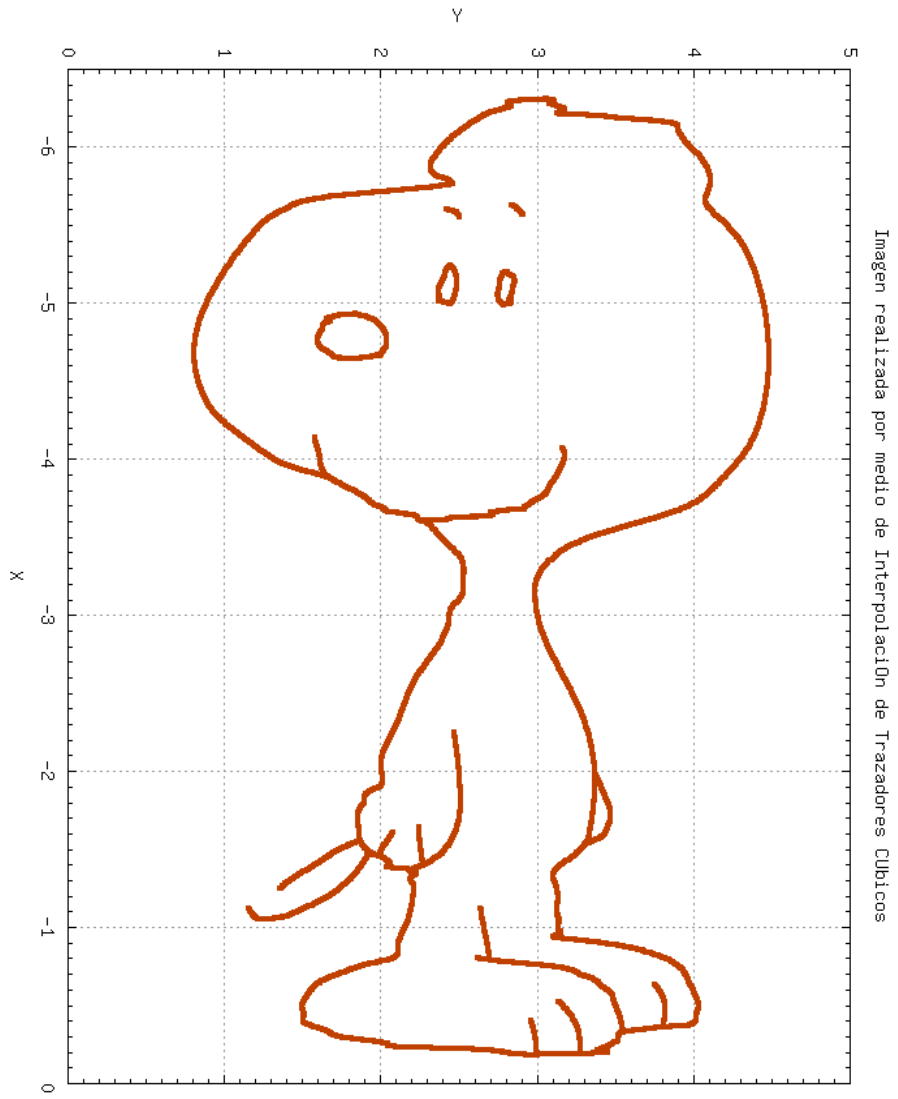


Figura 8. Gráfica obtenida por medio del MITCS.

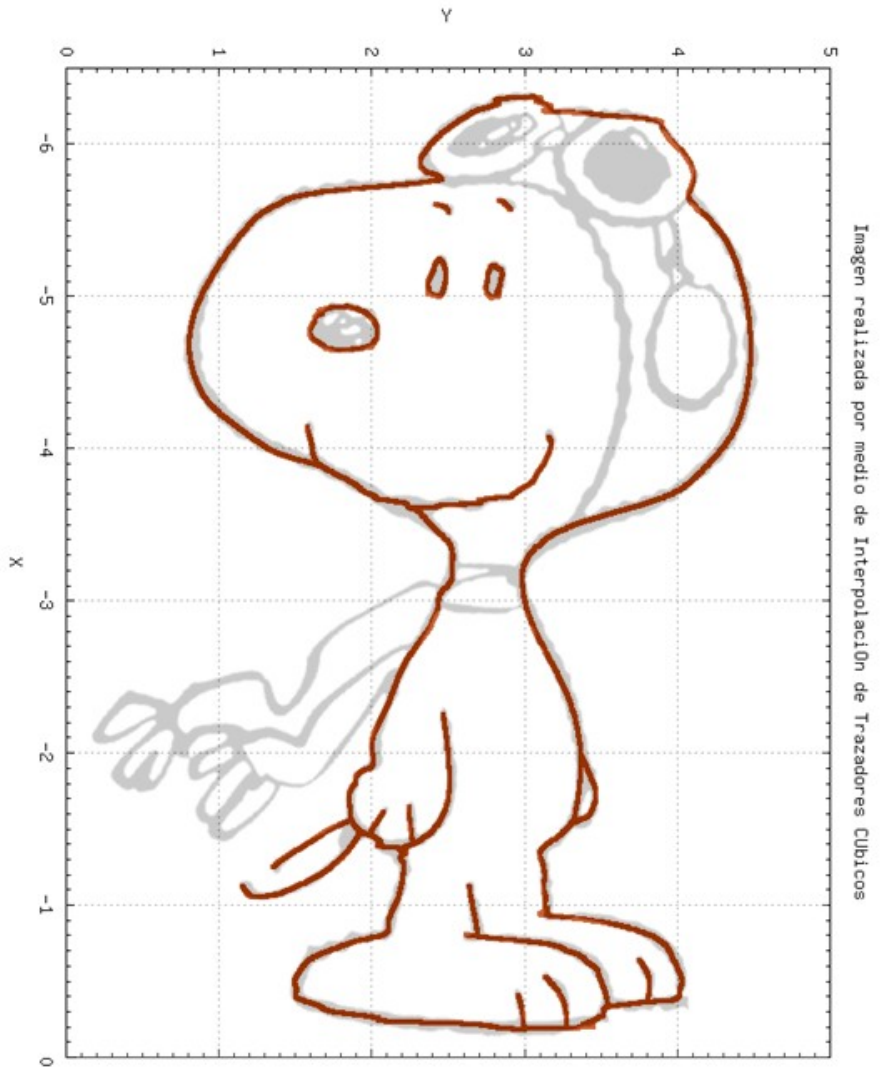


Figura 9. Sobreposición de la imagen interpolada (figura 8) y la original (figura 3).

BIBLIOGRAFÍA

[1] Villar, J. *Métodos numéricos con MATLAB. Aplicación a las telecomunicaciones.* España: Edicions UPC, 2003, p.67

[2] Louden, K. *Lenguajes de programación: Principios y práctica*. 2a edición. México: Cengage Learning Editores, 2004, p. 38.

[3] Alonso, P. *Diseño e implementación de programas en lenguaje C*. Universidad Politécnica de Valencia, 1998, p. 11.

[4] Zhang, T. *Aprendiendo C en 24 Horas.* México: Editorial Pearson Educación, 2001, p. 12 -17.

[5] Martin, O. *Uso de gnuplot* [en línea]. 15 de junio de 2007. Disponible en: http://termodinamica.us.es/tecnicas/como/node62.html (Palabras clave: gnuplot, gráficas, LINUX) [Consulta: 10 de junio de 2008]

[6] Burden, R. *Análisis Numérico.* 7ª edición. México: International Thomson Editores, 2002, p. 141-145, 147-149.

Programas empleados:

- *Ubuntu 6.06 (Dapper Drake).* Web: http://www.ubuntu.com/
- *Vi.* Web: http://ex-vi.sourceforge.net/
- *GCC*. Web: http://gcc.gnu.org/
- *Gnuplot.* Web: http://www.gnuplot.info/
- *Microsoft Windows XP.* Web: http://www.microsoft.com/windows/windows-xp/default.aspx
- *LogPaper.* Web: http://logicnet.dk/LogPaper/
- *PDFCreator.* Web: http://www.pdfforge.org/products/pdfcreator
- *Adobe PhotoShop CS1*. Web: http://www.adobe.com/es/products/photoshop/photoshop/