06

INTERPOLACIÓN Y APROXIMACIÓN DE UNA PISTA DE FÓRMULA 1

Jesús Ait Idir Lahuerta *UO281742@uniovi.es*
Universidad de Oviedo

PALABRAS CLAVE
Interpolación, numérico, aproximación, circuito, modelos.

REFERENCIA
AIT IDIR LAHUERTA, Jesús. «Interpolación y aproximación de una pista de Fórmula 1». En: TEMat, 9 (2025), págs. 79-90.
ISSN: 2530-9633.
URL: https://temat.es/articulo/2025-p79.

Este trabajo se distribuye bajo una licencia Creative Commons Reconocimiento 4.0 Internacional
https://creativecommons.org/licenses/by/4.0/

MSC2020
65-06.

Recibido:
25 de febrero de 2022.
Aceptado:
2 de enero de 2025.

01 RESUMEN

Con este artículo buscamos posibles acercamientos a un problema de interpolación y/o aproximación. Tomamos un circuito de Fórmula 1 y, a partir de su posición geográfica, logramos recuperar de diferentes formas la trayectoria que dibuja dicho circuito.
Utilizamos diferentes métodos conocidos de interpolación y aproximación, como el método de Lagrange, de Hermite o de series de Fourier, entre otros. Además, analizamos el circuito dividiéndolo en secciones y también lo estudiamos de forma global.
Tras probar estos diferentes métodos y perspectivas del problema obtenemos diferentes resultados. Tras compararlos exponemos ciertas conclusiones sobre qué método es el más adecuado para resolver el problema planteado.

02 ABSTRACT

In this article we look for an approach to an interpolation/approximation problem. We consider a Formula 1 race track and from its geographical position we recover in various ways the path of the track.
We use different known techniques of interpolation and approximation, such us Lagrange's, Hermite's or Fourier's methods, among others.
We also analyse the track by dividing it in parts or examining it as a whole.
After trying these different approaches and perspectives of the problem we obtain different results. After comparing them, we reach certain conclusions about which one is the best to solve the problem.

03 AGRADECIMIENTOS

Quiero dar las gracias a nuestros profesores de la Universidad de Oviedo, que me han motivado y han fomentado mi faceta divulgativa, a mis compañeros por ayudar con el proyecto inicial, a Susana y también a nuestras familias por su apoyo incondicional.

01. INTRODUCCIÓN

Buscamos con este artículo obtener una aproximación de un circuito cerrado en forma de función. El hecho de transformar un camino, carretera, etc., en una función es algo que puede facilitar enormemente muchas labores. Por ejemplo, cuando se construye una carretera en una ciudad, si queremos aproximar la cantidad de material que necesitaremos resulta casi imposible calcularlo a ojo, ya que habrá muchas desviaciones o curvas. También en cuanto a tareas de construcción o dimensionamiento el objetivo que nos proponemos puede ser muy útil. Al fin y al cabo, un mapa, por ejemplo, se crea a partir de una interpolación entre una cantidad muy elevada de puntos.

En nuestro artículo presentamos un ejemplo de este proceso. Hemos elegido un circuito de Fórmula 1, el circuito australiano de Albert Park. Nuestra labor ha consistido, básicamente, en probar diversas maneras de obtener una función similar a la del circuito. Hemos empleado tanto métodos que utilicen todos los puntos a nuestra disposición, como otros que interpolen por tramos. Podemos ver el circuito sobre el que trabajaremos en la figura 1.

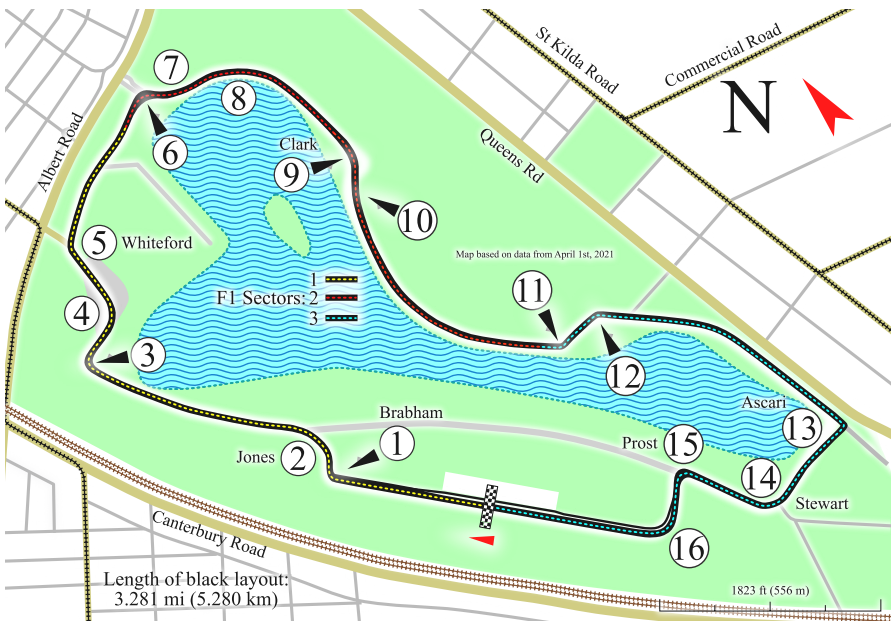


Figura 1: Circuito Albert Park. Imagen obtenida de WikiMedia Commons bajo una licencia Creative Commons Attribution-Share Alike 4.0 International.

02. PLANTEAMIENTO DEL MODELO MATEMÁTICO Y SUS PARÁMETROS

A continuación plantearemos el modelo matemático de nuestro problema. Para los diferentes códigos necesarios vamos a utilizar el programa MATLAB. Para estos programas hemos escrito algoritmos en forma de pseudocódigos que se encuentran en los anexos. Los códigos empleados para obtener los valores en este artículo pueden encontrarse adjuntos a este artículo.

Para realizar nuestro objetivo primero tenemos que obtener ciertos puntos del circuito y sus coordenadas geográficas. Vamos a geolocalizar el circuito y a marcarlo en rojo (véase la figura 2). Ahora, tomamos un grupo de puntos y buscamos sus coordenadas. Marcamos los puntos escogidos en la figura 3.

Una vez tenemos los puntos en coordenadas de latitud y longitud, tenemos que modificarlos para que trabajar con ellos. Vamos a crear un sistema de coordenadas más sencillo a partir del cual haremos nuestras operaciones y lo dibujaremos añadiendo abajo a la derecha la rosa de los vientos correspondiente (véase la figura 4). Las coordenadas de nuestro origen son (37°50′21″S, 144°57′38″E); en decimales, (−37,8392, 144,9606).

(a) Visión geográfica del circuito.

(b) Contorno del circuito.

Figura 2: Circuito.

Figura 3: Puntos geolocalizados.

Figura 4: Nuevo sistema de coordenadas.

Ahora, midiendo ambos ejes en kilómetros, podemos hallar una expresión (x, y) de cada punto mucho más sencilla. Para hacer esto, primero tendremos que poder calcular ciertas distancias que nos den una escala para todos los puntos. A partir de dos coordenadas, por ejemplo las de nuestro origen y el octavo punto, debemos obtener la distancia que separa estos puntos. Para ello utilizaremos lo que se conoce como la fórmula del *haversine* o del semiverseno [3]. Esta fórmula permite calcular la distancia entre dos puntos geográficos, dadas la longitud y latitud de ambos. Se basa en el uso de la función semiverseno, cuya definición es

$$\operatorname{semiversen}(\theta) = \operatorname{sen}^2\left(\frac{\theta}{2}\right).$$

Con esto, la fórmula del semiverseno es:

$$\operatorname{semiversen}\left(\frac{d}{R}\right) = \operatorname{semiversen}(\theta_1 - \theta_2) + \cos(\theta_1)\cdot\cos(\theta_2)\cdot\operatorname{semiversen}(\Delta\lambda),$$

donde d es la distancia buscada, R el radio de la Tierra,[1] θ_1 y θ_2 son las latitudes en radianes de los puntos y $\Delta\lambda$ la diferencia de longitudes. Así, haciendo uso de la función semiverseno inversa, que se obtiene a partir de la inversa del seno, podemos despejar d y hallar la distancia buscada:

$$d = 2\cdot R\cdot \arcsen\left(\sqrt{\operatorname{semiversen}(\theta_1 - \theta_2) + \cos(\theta_1)\cdot\cos(\theta_2)\cdot\operatorname{semiversen}(\Delta\lambda)}\right).$$

Apoyándonos en dicha fórmula y simplificando, podemos escribir el algoritmo 1 (véase el anexo A), que a partir de las coordenadas de dos puntos obtiene la distancia entre ellos. La distancia entre el origen y el vigésimo punto es 0,3418 km. Por tanto, dicho punto tendrá como coordenadas en el nuevo sistema el par (0,3418, 0). Podemos escalar para los demás puntos,[2] obteniendo el cuadro 1.

[1] Utilizamos el radio medio terrestre $R_T = 6378$ km.

[2] Esto es tan sencillo como imprimir la imagen a un tamaño cualquiera, establecer una escala y medir. También es posible hacerlo de forma informática, programas como MATLAB permiten, a partir de un origen y ejes, obtener las coordenadas de los píxeles. Con imágenes de mayor resolución, esta manera de proceder da muy buenos resultados.

Cuadro 1: Lista de puntos localizados junto a sus respectivas coordenadas en nuestro sistema de referencia.

Punto	Longitud	Latitud	Punto	Longitud	Latitud
1	1,3672	0,7196	17	0,1529	0,2069
2	1,3312	0,6476	18	0,2249	0,1934
3	1,2593	0,5667	19	0,3508	0,1349
4	1,1423	0,5127	20	0,3418	0
5	1,0344	0,4947	21	0,4947	0
6	0,8905	0,5217	22	0,6656	0,0360
7	0,7646	0,6116	23	0,8995	0,0989
8	0,6836	0,6746	24	1,0074	0,0540
9	0,6296	0,7646	25	1,2593	0,1349
10	0,4947	0,8095	26	1,5291	0,2249
11	0,3688	0,8185	27	1,7720	0,3598
12	0,3148	0,8005	28	1,7495	0,5217
13	0,2249	0,6836	28	1,9609	0,5307
14	0,1619	0,6386	30	1,9698	0,8095
15	0,1079	0,6296	31	1,7989	0,8257
16	0,1124	0,4677	32	1,6370	0,8365

2.1 INTERPOLACIÓN Y APROXIMACIÓN

Hasta ahora, hemos establecido como objetivo el aproximar el circuito elegido mediante una función. Sin embargo, para poder llevarlo a cabo, necesitamos definir matemáticamente qué se entiende por esta aproximación.

En general, ante este tipo de situaciones, se utiliza la interpolación. La interpolación de una función f se basa en encontrar otra función que coincida con la inicial en un número finito de puntos fijados previamente, bien porque la expresión de f es demasiado compleja y deseamos representarla mediante funciones más manejables, bien porque solo conocemos el valor que toma en los puntos que hemos fijado, como es el caso que nos ocupa una vez que hemos obtenido las coordenadas vistas anteriormente.

La idea de la aproximación de una función es en esencia la misma. Sin embargo, para estos métodos no es necesario que la función que aproxima a la inicial coincida con ella en los puntos fijados, sino que se busca que el error entre f en dichos puntos y el valor de la nueva función en ellos sea el mínimo posible.

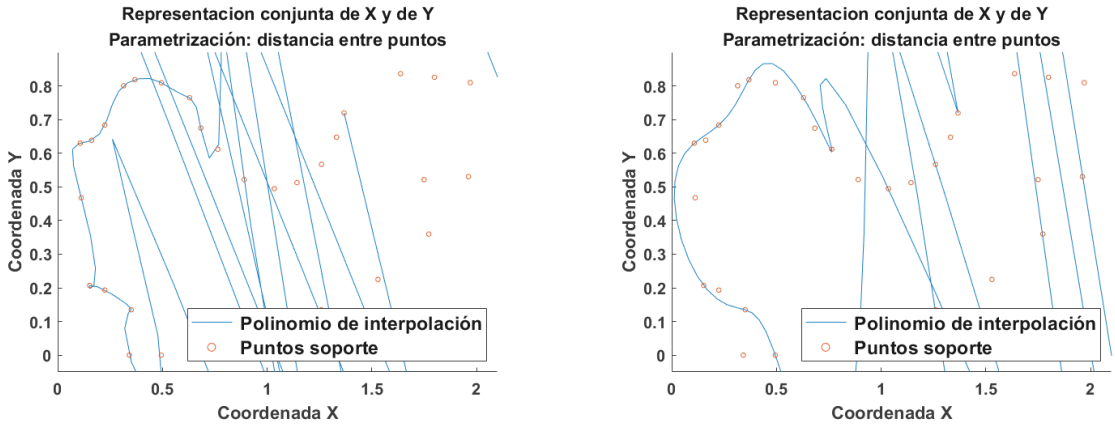
Además, para nuestro problema vemos que, para el mismo valor de x, habría en principio varias imágenes, por lo que no sería posible describir el comportamiento de todo el circuito mediante una función. Es por ello que utilizaremos la aproximación e interpolación paramétricas, que consisten en representar las coordenadas (x, y) en función de un parámetro t de la forma $(x(t), y(t))$, evitando así el problema. De esta forma, se aproximarán o interpolarán x respecto a t e y respecto a t y posteriormente se representará x frente a y.

03. RESULTADOS

3.1 APROXIMACIÓN DEL CIRCUITO GLOBALMENTE

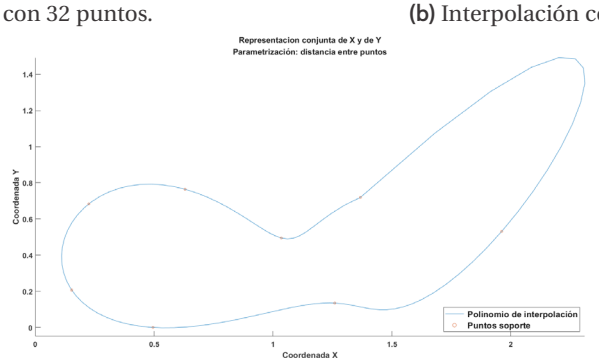
En general, el método de interpolación más usado es el de diferencias divididas [1]. Este se puede aplicar generalizando las diferencias divididas a un método paramétrico [6]. En este artículo probaremos con este método y, después, con otros métodos que darán mejores resultados y funcionarán con mayor precisión.

La elección de parámetro es algo bastante complejo y que afecta mucho a los resultados [7]. Vamos a probar como parámetro la distancia entre dos puntos consecutivos.



(a) Interpolación con 32 puntos.

(b) Interpolación con dieciséis puntos.



(c) Interpolación con ocho puntos.

Figura 5: Interpolación parámetrica.

3.1.1 INTERPOLACIÓN PARAMÉTRICA

Una primera aproximación es utilizar el método de interpolación de Lagrange. Este método se basa en la construcción de un polinomio que pase por todos nuestros puntos (si el grado del polinomio es igual al número de puntos menos, siempre podemos encontrar uno) a partir de los polinomios de interpolación de Lagrange. Dados $n+1$ puntos $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$, este polinomio $P(x)$ de grado n es el siguiente:

$$P(x) = \sum_{i=0}^{n} y_i L_i(x),$$

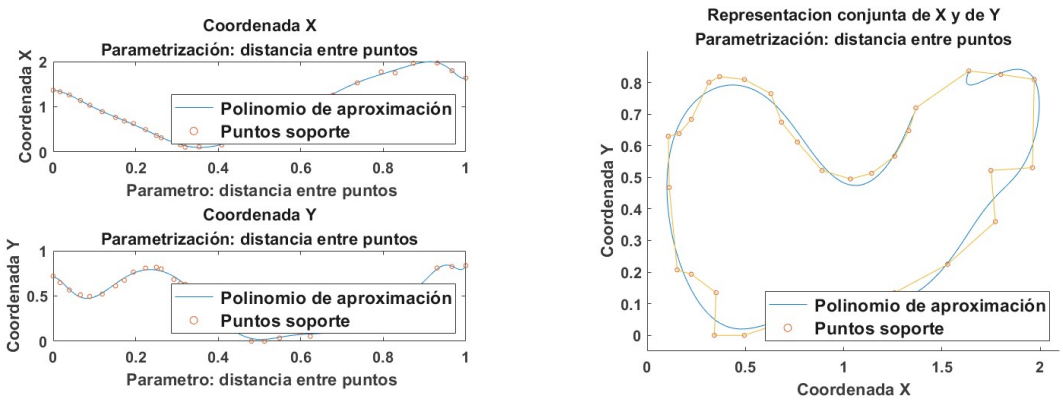
donde $L_i(x)$ son los conocidos polinomios de Lagrange,

$$L_i(x) = \prod_{\substack{0 \le j \le n, \\ j \neq i}} \frac{x - x_j}{x_i - x_j}.$$

Estos polinomios cumplen que $L_i(x_j) = 0$ para $i \neq j$ y $L_i(x_i) = 1$. Por tanto, si $x = x_i$ entonces $P(x_i) = y_i$ y aseguramos que $P(x)$ efectivamente pasa por todos nuestros nodos.

El problema con el método de Lagrange es que, cuando hay muchos puntos que interpolar, deja de ser eficiente, al obligar que un mismo polinomio pase por todos los nodos. Para llevar a cabo la interpolación hemos utilizado el algoritmo 2 (véase el anexo A). Veamos en la figura 5 lo que ocurre.

Observamos el conocido fenómeno de Runge. Este fenómeno explica que, en ciertas interpolaciones, al aumentar el número de nodos el error empeora en lugar de mejorar, tendiendo a infinito cuando el número de nodos tiende a infinito. En nuestra figura, con ocho nodos tenemos una mejor aproximación del circuito que usándolos todos.



(a) Valores de x e y por separado.

(b) Valores de (x, y) parametrizados.

Figura 6: Aproximación paramétrica.

En general, por esta razón vamos a desechar directamente la interpolación parámetrica y a explorar otros métodos, como la aproximación o incluso la interpolación por partes y no global.

3.1.2 APROXIMACIÓN PARAMÉTRICA

Ante el mal rendimiento de la interpolación paramétrica, tenemos que recurrir a otras opciones, como la aproximación [9].

Hemos elaborado el algoritmo 3 (véase el anexo A), que aproxima una función mediante el método de mínimos cuadrados para una base polinómica. En general, el método de mínimos cuadrados consiste en expresar una función como combinación lineal de elementos de una base de funciones y minimizar el cuadrado del error entre la función real y la buscada. En este caso, la base está formada por los polinomios $\{1, x, x^2, \ldots\}$ hasta el grado n deseado [4]. Así, aproximamos una función f por la combinación lineal

$$\Phi := \sum_{i=0}^{n} c_i x^i.$$

La función a minimizar es

$$d(c_0, \ldots, c_n) = \|f - \Phi\|_2^2 = \langle f - \Phi, f - \Phi \rangle = \langle f, f \rangle - 2\langle f, \Phi \rangle + \langle \Phi, \Phi \rangle,$$

con $\langle \cdot, \cdot \rangle$ el producto escalar

$$\langle f, \Phi \rangle = \sum_{k=1}^{N} f(x_k)\Phi(x_k),$$

y siendo $\{x_k\}$ los puntos donde conocemos la función f.

Para encontrar los mínimos de la función d igualamos cada una de sus derivadas parciales a cero, obteniendo el siguiente sistema lineal de ecuaciones, que es el que nos proporciona los coeficientes de la función de aproximación:

$$\begin{pmatrix} \langle 1,1\rangle & \langle 1,x\rangle & \cdots & \langle 1,x^n\rangle \\ \langle x,1\rangle & \langle x,x\rangle & \cdots & \langle x,x^n\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle x^n,1\rangle & \langle x^n,x\rangle & \cdots & \langle x^n,x^n\rangle \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} \langle 1,f\rangle \\ \langle x,f\rangle \\ \vdots \\ \langle x^n,f\rangle \end{pmatrix}.$$

En la práctica, se conocen descomposiciones de este sistema que permiten resolverlo de forma más eficiente [1], como la utilizada para el algoritmo 3.

Basándonos en ese algoritmo podemos escribir otro, de aproximación paramétrica: el algoritmo 4 (véase el anexo A), en el que utilizamos el método antes expuesto para aproximar x e y, por separado, cada una

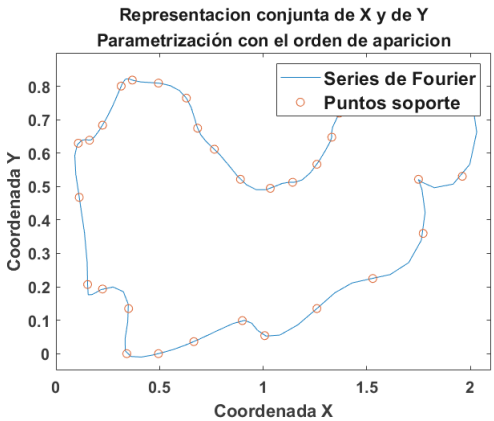


Figura 7: Interpolación de Fourier.

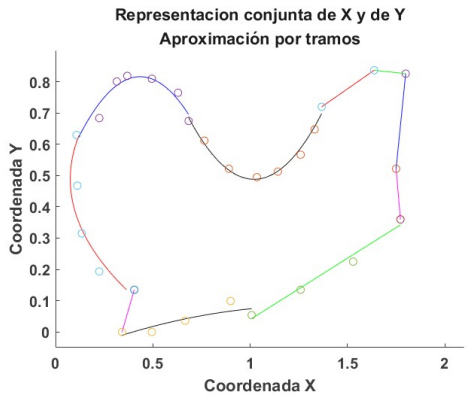


Figura 8: Aproximación por tramos.

en función del parámetro que habíamos elegido [9]. Tras ejecutarlo representamos la función, donde recordamos que habíamos tomado la distancia entre cada dos puntos como parámetro.[3] Obtenemos la figura 6.[4]

3.1.3 INTERPOLACIÓN POR SERIES DE FOURIER

Después de probar la aproximación paramétrica probaremos otro conocido método, el de interpolación por series de Fourier, implementado en el algoritmo 5 (véase el anexo A). Este método podría ser adecuado debido a que incluye las condiciones de periodicidad [5] y es preciso cuando las funciones a interpolar no presentan saltos.

El método es bastante simple. Primero se obtiene la transformada de Fourier para los valores de nuestro circuito. Se interpola la transformada de estos valores dando lugar al número que queramos de puntos equiespaciados, que después se devuelven a su formato original.[5] Esta técnica es muy similar a la utilizada en la sección anterior, con la diferencia que la base de funciones es $\{1, \text{sen}(x), \cos(x), \text{sen}(2x), \cos(2x), \ldots\}$.

Vemos los resultados obtenidos en la figura 7, de nuevo teniendo en cuenta que la aproximación que realizamos es paramétrica.

3.2 APROXIMACIÓN DEL CIRCUITO POR PARTES

Hasta ahora hemos obtenido resultados para diferentes métodos que nos dan una representación del circuito de manera general, con respecto a todos los puntos. Ahora probaremos a hacerlo por partes.

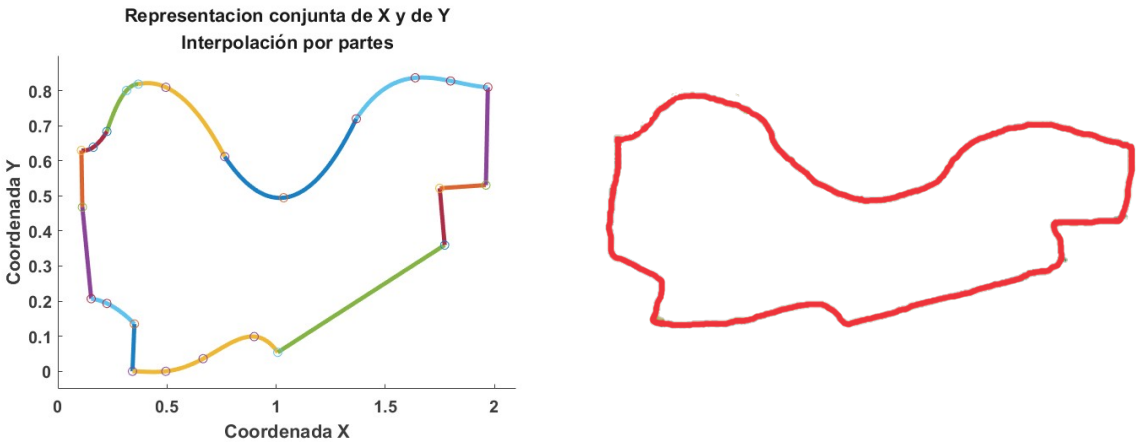
3.2.1 APROXIMACIÓN POR TRAMOS

Dividimos los puntos en pequeños grupos y, fijándonos en el dibujo del circuito, analizamos qué función polinómica le puede servir mejor. Utilizamos diferentes parábolas y rectas dependiendo del dibujo. Es por esto que usaremos de nuevo el algoritmo de aproximación con base polinómica, pero escogiendo la base $\{1, x\}$ o $\{1, x, x^2\}$ según consideremos que el tramo correspondiente del circuito sigue una trayectoria recta o parabólica (véase la figura 8).

[3] Este parámetro tiene más sentido que otros como, por ejemplo, la distancia de los puntos al origen, ya que el parámetro no trata a todos los puntos por igual, sino en función de la distancia entre cada dos; es decir, tiene en cuenta una mayor cantidad de factores [7].

[4] Hemos elegido para la aproximación una base de polinomios hasta grado 19. Probando una base mayor nos encontramos con problemas de *overfitting* (incluso oscilaciones), mientras que una base menor da resultados menos precisos.

[5] Hemos elegido cien puntos porque, a través de diferentes pruebas, vimos que obteníamos buenos resultados: escoger más puntos no mejoraba mucho la precisión.



(a) Interpolación por partes. **(b)** Contorno de la pista.

Figura 9: Interpolación por partes.

Vemos que en este caso, al aproximar con tantos puntos, la aproximación con base polinómica se hace algo menos precisa. Por eso tenemos algunos problemas que, en particular, se ven en la unión de cada aproximación. Al tratarse de una aproximación llevada a cabo a trozos vemos que no mantiene la continuidad del circuito ya que no se le exige esto en ningún momento: por tanto, no es una buena aproximación y tenemos que probar con otros métodos.

3.2.2 INTERPOLACIÓN POR PARTES

Debido a los problemas vistos anteriormente, vamos a probar a utilizar interpolación (en este caso sí imponemos que la función pase por los puntos conocidos) con las mismas bases anteriores, $\{1, x\}$ o $\{1, x, x^2\}$, y lo hacemos por partes. Para garantizar la continuidad, en cada subconjunto de puntos incluimos el último del anterior, por lo que la interpolación nos garantiza que ambas partes de la función coincidirán en dicho punto. Obtenemos la figura 9 que, comparada con el recorrido de la pista real, nos deja ver una precisión realmente buena. Representamos en colores las diferentes funciones que hemos obtenido con la interpolación.

3.2.3 INTERPOLACIÓN POR SPLINES CÚBICOS

Probamos ahora a aproximar el circuito utilizando splines cúbicos. El lector puede encontrar su implementación en el algoritmo 6 (véase el anexo A).

El spline cúbico es uno de los más utilizados y es realmente útil, ya que al usar polinomios de grado bajo obtenemos menos errores e imprecisiones. Este método interpola una función con curvas suaves [4]. Más concretamente, un spline es una función que se define por varios polinomios de igual grado, cada uno definido sobre un subintervalo del intervalo original y que se unen entre sí atendiendo a una serie de condiciones iniciales impuestas que garantizan también la continuidad. En este caso, el grado de los polinomios que utilizaremos será 3, por lo que el spline será

$$S(x) = P_3^k(x) = a_k(x - x_k)^3 + b_k(x - x_k)^2 + c_k(x - x_k) + d_k \quad \text{si } x \in [x_k, x_{k+1}],$$

donde denotamos $x_{N+1} = x_1$, pues nuestro circuito es cerrado.

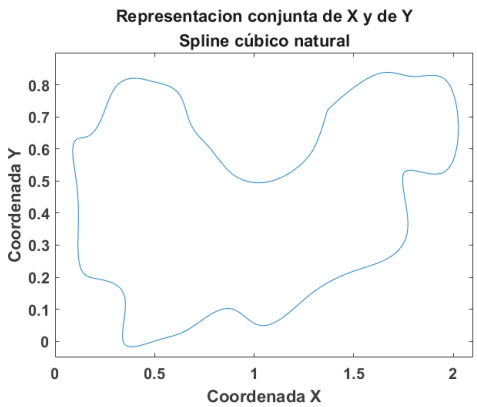


Figura 10: Spline natural con distancia entre puntos como parámetro.

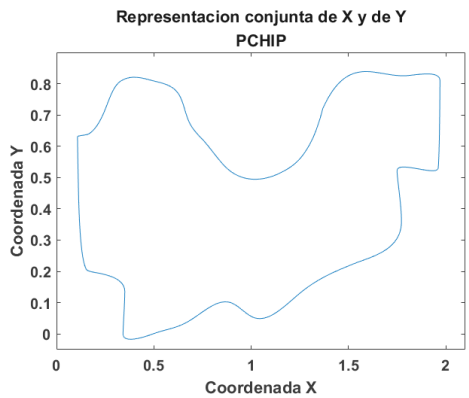


Figura 11: PCHIP con distancia entre puntos como parámetro.

Por otra parte, las condiciones iniciales son:

$$\begin{aligned} P_3^k(x_k) &= f(x_k), && k = 1, \dots, N \\ P_3^k(x_{k+1}) &= f(x_{k+1}), && k = 1, \dots, N \\ \frac{\mathrm{d}}{\mathrm{d}x} P_3^k(x_{k+1}) &= \frac{\mathrm{d}}{\mathrm{d}x} P_3^{k+1}(x_{k+1}) && k = 1, \dots, N \\ \frac{\mathrm{d}^2}{\mathrm{d}x^2} P_3^k(x_{k+1}) &= \frac{\mathrm{d}^2}{\mathrm{d}x^2} P_3^{k+1}(x_{k+1}) && k = 1, \dots, N. \end{aligned}$$

Con esta información aún hay dos incógnitas más que ecuaciones, por lo que necesitamos imponer dos condiciones más. Dependiendo de las elegidas tendremos los splines naturales, tomando $S''(a) = S''(b) = 0$, y los sujetos, fijando $S'(a) = m_a$, $S'(b) = m_b$ con m_a y m_b las pendientes deseadas en a y b. Así, resolviendo este sistema, que se puede representar de manera matricial, obtenemos de forma única los coeficientes deseados [8]. Es importante notar que, aunque tengamos una única función spline, este método no deja de ser una interpolación a trozos donde en cada intervalo se aproxima por un polinomio cúbico.

Estudiamos el spline cúbico natural para el parámetro de distancia entre los puntos (véase la figura 10).[6] Hemos elegido este método debido a que, en general, el spline cúbico natural aporta mejores resultados cuando las curvas a interpolar son suaves, como lo son bastantes en este caso. El spline sujeto requiere conocer ciertas condiciones de contorno en los extremos para obtener mayor precisión, pero, en este caso, esta mayor precisión podría derivar en cierto *overfitting* debido a que tratamos con curvas suaves.

3.2.4 INTERPOLACIÓN CÚBICA DE HERMITE

Para finalizar este apartado vamos a utilizar la interpolación cúbica de Hermite, que también recibe el nombre de PCHIP (*Piecewise Cubic Hermite Interpolation Polynomial*). Modificamos entonces ligeramente el algoritmo anterior, usando la función `pchip` en vez de la función `spline` [2]. Este método es bastante similar al spline cúbico. Este método de interpolación por partes utiliza polinomios de Hermite en lugar de polinomios de Lagrange. Por eso, la diferencia principal con los splines es que el PCHIP utiliza también el valor de la derivada de la función en cada punto, consiguiendo una mejor y más suave transición entre segmentos. Esta precisión añadida es además especialmente útil a la hora de evitar oscilaciones indeseadas, las cuales a veces sí pueden aparecer para splines. Obtenemos la figura 11.

[6] Se ha hecho la aproximación para mil puntos equidistantes ya que con este número obtuvimos buenos resultados, evitando hacer *overfitting* y alcanzando una buena precisión.

3.3 ERRORES

Para poder comparar todos los métodos hemos buscado establecer una medida del error para la bondad del ajuste de cada uno. Hemos creado una malla de cien puntos tomados del circuito real y en cada método hemos calculado la norma euclídea de la diferencia entre los valores experimentales correspondientes y los cien puntos del circuito. Hemos obtenido la siguiente tabla de errores.

Método	Error
Interpolación paramétrica (con ocho puntos)	25,6960
Aproximación paramétrica	0,1103
Interpolación Fourier	3,3169
Aproximación por tramos	7,4900
Interpolación por partes	0,0316
Interpolación por splines cúbicos	2,3004
Interpolación cúbica Hermite	2,1701

04. CONCLUSIONES

Después de haber probado con varios métodos diferentes, hemos obtenido las siguientes conclusiones.

Para empezar, concluimos algo que ya suponíamos desde un principio: las aproximaciones globales, en general, resultan más inexactas que las que realizamos por partes. Además, vimos al inicio que la interpolación para todos los puntos daba pésimos resultados. Como ya explicamos, esto se debe a que, al obligar a la función de interpolación a pasar por todos los puntos, esta presenta muchas oscilaciones y empeora el resultado: este hecho se conoce como fenómeno de Runge. Esto se podría solucionar eligiendo nodos con el método de Chebyshev, el cual nos aseguraría un error menor. Debido a este fenómeno, entonces, resulta lógico que tengamos peores resultados en este método que en los métodos posteriores que interpolan por tramos o aproximan en general.

A pesar de que la aproximación con más error sea la aproximación por tramos, conjeturamos que esto se debe a los errores de continuidad en nuestro circuito, que son bastante relevantes. De todas formas, la interpolación por partes nos da resultados óptimos, confirmando nuestra hipótesis. Las interpolaciones por series de Fourier, de Hermite y por splines dan buenos resultados pero peores que la interpolación por partes. Esto es coherente: al fin y al cabo, al interpolar por partes hemos elegido con criterio la mejor opción para cada tramo y estamos utilizando más información que los otros métodos.

Sorprendentemente, la aproximación paramétrica global sí nos da un circuito que se asemeja a la forma de la pista sobre la que estamos trabajando, con un error realmente bajo. Se trata de un método de aproximación, evitando el fenómeno de Runge, y que además no sufre los problemas de continuidad que sí afectan a la aproximación por tramos.

Como conclusión final, hemos observado que los mejores métodos son la aproximación paramétrica y la interpolación por partes. Que el segundo lo fuera era algo que cabía esperar, ya que se toman los puntos en grupos más pequeños decidiendo el tipo de interpolación en cada caso. La aproximación paramétrica no consigue ser tan exacta pero se acerca mucho, como mínimo brindando una imagen bastante buena de la forma del circuito sin quizás acabar de ser excesivamente detallista.

REFERENCIAS

[1] Boor, Carl de. «Divided differences». En: *Surveys in Approximation Theory* 1 (2005), págs. 46-69. ISSN: 1555-578X.

[2] Boor, Carl de; Höllig, Klaus, y Sabin, Malcolm. «High accuracy geometric Hermite interpolation». En: *Computer Aided Geometric Design* 4.4 (1987), págs. 269-278. https://doi.org/10.1016/0167-8396(87)90002-1.

[3] Castillo, Génesis. *Calculo Por Haversine*. URL: https://es.scribd.com/document/399565628/Calculo-por-Haversine-docx (visitado 24-05-2025).

[4] Chapra, Steven C. y Canale, Raymond P. *Numerical Methods for Engineers*. 7.ª ed. Nueva York, US: McGraw-Hill Education, 2015. ISBN: 978-0-07-339792-4.

[5] González, Genaro. «Series de Fourier, Transformadas de Fourier y Aplicaciones». En: *Divulgaciones Matemáticas* 5 (1997). ISSN: 1315-2068.

[6] Jeong, Soon Yong; Choi, Yun Jong, y Park, PooGyeon. «Parametric interpolation using sampled data». En: *Computer-Aided Design* 38.1 (2006), págs. 39-47. https://doi.org/10.1016/j.cad.2005.06.002.

[7] Kuznetsov, Evgenii B. y Yakimovich, Anna Yu. «The best parameterization for parametric interpolation». En: *Journal of Computational and Applied Mathematics* 191.2 (2006), págs. 239-245. https://doi.org/10.1016/j.cam.2005.06.040.

[8] McKinley, Sky y Levine, Megan. *Cubic Spline Interpolation*. URL: https://es.scribd.com/document/24255441/Cubic-Spline-Interpolation (visitado 24-05-2025).

[9] Shalashilin, V. I. y Kuznetsov, Evgenii B. «The Parametric Approximation». En: *Parametric Continuation and Optimal Parametrization in Applied Mathematics and Mechanics*. Dordrecht, NL: Springer Netherlands, 2003, págs. 149-164. https://doi.org/10.1007/978-94-017-2537-8_6.

A. ANEXO: ALGORITMOS PARA LOS DIFERENTES MÉTODOS DE INTERPOLACIÓN

A continuación presentamos los distintos algoritmos empleados a lo largo de este texto, redactados en pseudocódigo. El lector interesado puede encontrar los archivos de código fuente escritos en el lenguaje MATLAB en el siguiente enlace:

Programas en MATLAB: https://temat.es/articulo/2025-p79/2025-p79-a-anexoiaF1-zip.

Algoritmo 1 (Cálculo de la distancia usando coordenadas).

función DISTANCIA((a,b),(c,d))
    convertir las coordenadas (a,b) y (c,d) a grados
    convertir las coordenadas (a,b) y (c,d) a radianes
    devolver $2\cdot 6380\cdot \arcsin\left(\sqrt{\operatorname{semiversen}(a-c)+\cos(a)\cdot\cos(c)\cdot\operatorname{semiversen}(b-d)}\right)$
fin función

Algoritmo 2 (Interpolación paramétrica).

subrutina INTERPOLACIÓNPARAMÉTRICA(X, Y, T)
    calcular el polinomio de interpolación de Lagrange $x(t)$ de X en T para una malla de puntos.
    calcular el polinomio de interpolación de Lagrange $y(t)$ de Y en T para una malla de puntos.
    representar los puntos $(x(t),y(t))$
fin subrutina

Algoritmo 3 (Aproximación en una base polinómica).

función APROXIMACIÓNPOLINÓMICA(X, Y, n)
    definir la matriz $M_{i,j}\leftarrow X_i^{j-1}$ para $0\le i<\log X$ y $0\le j<n$
    calcular $A\leftarrow M^\intercal\cdot M$
    calcular $b\leftarrow M^\intercal\cdot Y^\intercal$
    resolver el sistema $Ac=b$
    devolver c
fin función

Algoritmo 4 (Aproximación paramétrica).

subrutina APROXIMACIÓNPARAMÉTRICA(X, Y, T)
    calcular la aproximación $c_1\leftarrow$ APROXIMACIÓNPOLINÓMICA(T, X, 20)
    calcular la aproximación $c_2\leftarrow$ APROXIMACIÓNPOLINÓMICA(T, Y, 20)
    definir el polinomio $x(t)\leftarrow c_1(0)t^0+\cdots+c_1(19)t^{19}$
    definir el polinomio $y(t)\leftarrow c_2(0)t^0+\cdots+c_2(19)t^{19}$
    representar los puntos $(x(t),y(t))$
fin subrutina

Algoritmo 5 (Interpolación por el método de Fourier).

1: **subrutina** InterpoladónFourier(X, Y, n)
2: calcular la interpolación de Fourier $x(t)$ de X en n puntos equiespaciados
3: calcular la interpolación de Fourier $y(t)$ de Y en n puntos equiespaciados
4: representar los puntos $(x(t), y(t))$
5: **fin subrutina**

Algoritmo 6 (Interpolación por splines cúbicos).

Precondición $X_N = X_0 \vee Y_N = Y_0$

1: **subrutina** InterpoladónSpline(X, Y, n)
2: calcular la interpolación por spline cúbico $x(t)$ de X en n puntos equiespaciados
3: calcular la interpolación por spline cúbico $y(t)$ de Y en n puntos equiespaciados
4: representar los puntos $(x(t), y(t))$
5: **fin subrutina**