75.12/95.04 ANÁLISIS NUMÉRICO I
95.10 MODELACIÓN NUMÉRICA
95.13 MÉTODOS MATEMÁTICOS Y NUMÉRICOS
INTERPOLACIÓN - PARTE 2

Ing. Rodolfo A. Schwarz

Facultad de Ingeniería – Universidad de Buenos Aires

Año 2022

Índice

Método de Hermite

- Supongamos ahora que tenemos la siguiente serie de datos:

i	x_i	y_i	y'_i
0	x_0	y_0	y'_0
1	x_1	y_1	y'_1
2	x_2	y_2	y'_2
3	x_3	y_3	y'_3

- Nuevamente, esta tabla puede representar los resultados de mediciones o de cálculos.
- Al igual que en los casos anteriores, los datos se pueden usar para representar en forma discreta una función.

Método de Hermite

- Como antes, podemos utilizar un polinomio para interpolar y generar un sistema de ecuaciones lineales:

$$
\begin{aligned}
a_0 + a_1\,x_0 + a_2\,{x_0}^2 + \ldots + a_m\,{x_0}^m &= y_0, \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
a_0 + a_1\,x_n + a_2\,{x_n}^2 + \ldots + a_m\,{x_n}^m &= y_n, \\
a_1 + a_2\,2\,x_0 + \ldots + a_m\,m\,{x_0}^{m-1} &= y'_0, \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
a_1 + a_2\,2\,x_n + \ldots + a_m\,m\,{x_n}^{m-1} &= y'_n.
\end{aligned}
$$

- Generamos $2\,(n+1)$ ecuaciones, de modo tal que el polinomio interpolante será de grado $m = 2\,(n+1) - 1 = 2\,n + 1$.

Método de Hermite

- Cmo hicimos antes, lo podemos expresar en forma matricial:

$$\begin{bmatrix} 1 & x_0 & {x_0}^2 & \dots & \dots & {x_0}^{2n+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & {x_n}^2 & \dots & \dots & {x_n}^{2n+1} \\ 0 & 1 & 2\,x_0 & 3\,{x_0}^2 & \dots & (2n+1)\,{x_0}^{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 2\,x_n & 3\,{x_n}^2 & \dots & (2n+1)\,{x_n}^{2n} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ \vdots \\ a_{2n} \\ a_{2n+1} \end{bmatrix} = \begin{bmatrix} y_0 \\ \vdots \\ y_n \\ y'_0 \\ \vdots \\ y'_n \end{bmatrix}$$

- Tenemos otra vez un *Sistema de Ecuaciones Lineales*.
- Esto nos facilita resolver el problema.
- Como hemos visto que para interpolar los datos resulta más fácil utilizar algún algoritmo que resolver el sistema en forma explícita, generemos un nuevo agoritmo para este caso.

Método de Hermite

- Vamos a definir nuestro polinomio interpolante de esta forma:

$$H_{2n+1}(x) = \sum_{i=0}^{n} y_i \, H_{n,i}(x) + \sum_{i=0}^{n} y_i' \, \hat{H}_{n,i}(x),$$

  que debe pasar por y_i y por y_i'.

- Esta forma es similar al *Método de Lagrange*.
- Entonces, ¿por qué no usar un algoritmo que incluya los polinomios de Lagrange ($L_{n,i}(x)$)?
- Así, definamos lo siguiente:

$$H_{n,i}(x) = \left[1 - 2\,(x - x_i)\, L_{n,i}'(x_i)\right] \cdot (L_{n,i}(x))^2,$$

$$\hat{H}_{n,i}(x) = (x - x_i) \cdot (L_{n,i}(x))^2.$$

Método de Hermite

- El polinomio $H_{n,i}(x)$ debe cumplir con las siguientes condiciones para que $H_{2n+1}(x)$ pase por los y_i:
  1. Para $x = x_i$:

$$H_{n,i}(x_i) = 1 \ \rightarrow \ [1 - 2\,(x_i - x_i)\,L'_{n,i}(x_i)] \cdot (L_{n,i}(x_i))^2 = 1 \cdot 1^2 = 1.$$

  2. Para $x = x_j$ y $j \neq i$:

$$H_{n,i}(x_j) = 0 \ \rightarrow \ [1 - 2\,(x_j - x_i)\,L'_{n,i}(x_i)] \cdot (L_{n,i}(x_j))^2 = (1 - 2\,h_j\,L'_{n,i}(x_i)) \cdot 0 = 0.$$

- Análogamente, el polinomio $\hat{H}_{n,i}(x)$ debe cumplir con las siguientes condiciones:
  1. Para $x = x_i$:

$$\hat{H}_{n,i}(x_i) = 0 \ \rightarrow \ (x_i - x_i) \cdot (L_{n,i}(x_i))^2 = 0 \cdot 1^2 = 0.$$

  2. Para $x = x_j$ y $j \neq i$:

$$\hat{H}_{n,i}(x_j) = 0 \ \rightarrow \ (x_j - x_i) \cdot (L_{n,i}(x_j))^2 = h_j \cdot 0 = 0.$$

Método de Hermite

- El polinomio $H'_{n,i}(x)$ debe cumplir con las siguientes condiciones para que $H'_{2n+1}(x)$ pase por los y'_i:
  1. Para $x = x_i$:

$$H'_{n,i}(x_i) = 0 \;\rightarrow$$
$$-\,2\,L'_{n,i}(x_i)\,(L_{n,i}(x_i))^2 + [1 - 2\,(x_i - x_i)\,L'_{n,i}(x_i)]\,2\,L_{n,i}(x_i)\,L'_{n,i}(x_i) = 0.$$

  2. Para $x = x_j$ y $j \neq i$:

$$H'_{n,i}(x_j) = 0 \;\rightarrow$$
$$-\,2\,L'_{n,i}(x_i)\,(L_{n,i}(x_j))^2 + [1 - 2\,(x_j - x_i)\,L'_{n,i}(x_i)]\,2\,L_{n,i}(x_j)\,L'_{n,i}(x_j) = 0.$$

- Análogamente, el polinomio $\hat{H}'_{n,i}(x)$ debe cumplir con las siguientes condiciones:
  1. Para $x = x_i$:

$$\hat{H}'_{n,i}(x_i) = 1 \;\rightarrow\; (L_{n,i}(x_i))^2 + (x_i - x_i)\,L_{n,i}(x_i)\,L'_{n,i}(x_i) = 1.$$

  2. Para $x = x_j$ y $j \neq i$:

$$\hat{H}'_{n,i}(x_j) = 0 \;\rightarrow\; (L_{n,i}(x_j))^2 + (x_j - x_i)\,L_{n,i}(x_j)\,L'_{n,i}(x_j) = 0.$$

Método de Hermite

- Con estas condiciones, para un x_j con j entre 0 y *n*:

$$H_{2n+1}(x_j) = \sum_{i=0}^{n} y_i \, H_{n,i}(x_j) + \sum_{i=0}^{n} y_i' \, \hat{H}_{n,i}(x_j) = y_j,$$
$$H_{2n+1}'(x_j) = \sum_{i=0}^{n} y_i \, H_{n,i}'(x_j) + \sum_{i=0}^{n} y_i' \, \hat{H}_{n,i}'(x_j) = y_j'.$$

- En consecuencia, este algoritmo cumple con la condición de pasar por los puntos que son datos. Este algoritmo se conoce como **Método de Hermite**.
- Al igual que el **Método de Lagrange**, el orden de los puntos no incide en la interpolación, y es posible que en el caso de puntos distribuidos uniformemente ($x_{i+1} - x_i = \mathrm{cte}$) se produzca el *Fenómeno de Runge*.

Método de Hermite

- Otra forma de obtener el **Método de Hermite** es aplicando una adaptación del **Método de Newton**.
- Para ello, armemos una nueva tabla con los datos del problema, creando una nueva variable z_j , cuyos valores cumplirán la siguiente condición:

$$\underbrace{z_{2i}}_{z_j} = \underbrace{z_{2i+1}}_{z_{j+1}} = x_i,$$

con

$$i = 0, 1, \ldots, n \quad \wedge \quad j = 0, 1, \ldots, 2n + 1.$$

- En consecuencia, también tenemos que:

$$\underbrace{f(z_{2i})}_{f(z_j)} = \underbrace{f(z_{2i+1})}_{f(z_{j+1})} = f(x_i).$$

Método de Hermite

- Para esta forma debemos considerar que:

$$F(z_{2i}, z_{2i+1}) = \frac{f(z_{2i+1}) - f(z_{2i})}{z_{2i+1} - z_{2i}} = \frac{f(x_i) - f(x_i)}{x_i - x_i} = f'(x_i),$$

que

$$F(z_{2i-1}, z_{2i}) = \frac{f(z_{2i}) - f(z_{2i-1})}{z_{2i} - z_{2i-1}} = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} = F(x_{i-1}, x_i).$$

- También que:

$$F(z_{2i}, z_{2i+1}, z_{2i+2}) = \frac{F(z_{2i+1}, z_{2i+2}) - F(z_{2i}, z_{2i+1})}{z_{2i+2} - z_{2i}} = \frac{F(x_i, x_{i+1}) - f'(x_i)}{x_{i+1} - x_i},$$

y que

$$F(z_{2i-1}, z_{2i}, z_{2i+1}) = \frac{F(z_{2i}, z_{2i+1}) - F(z_{2i-1}, z_2)}{z_{2i+1} - z_{2i-1}} = \frac{f'(x_i) - F(x_{i-1}, x_i)}{x_i - x_{i-1}}.$$

Método de Hermite

- Con estas consideraciones podemos armar un nuevo polinomio interpolante:

$$P_{2n+1}(z) = F(z_0) + F(z_0, z_1)(z - z_0) + F(z_0, z_1, z_2)(z - z_0)(z - z_1) + \dots .$$

- Si reemplazamos z por x nos queda:

$$P_{2n+1}(x) = F(x_0) + F(x_0, x_0)(x - x_0) + F(x_0, x_0, x_1)(x - x_0)(x - x_0) + \dots ,$$

que resulta ser

$$P_{2n+1}(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{F(x_0, x_1) - f'(x_0)}{x_1 - x_0}(x - x_0)^2 + \dots .$$

- Lo mismo podemos hacer si ordenamos en forma descendente los datos.

Método de Hermite Segmentado

- Al igual que con los **Métodos de Lagrange** y de **Newton**, polinomios de alto grado pueden generar distorsiones en los extremos.
- Una forma de evitar eso es aplicar el **Método de Hermite Segmentado**.
- Entre dos puntos sucesivos contamos con cuatro datos: y_i, y_i', y_{i+1}, y_{i+1}'. Con eso podemos generar una interpolación entre esos puntos sucesivos con una parábola cúbica.
- Para ello planteamos:

$$
\begin{aligned}
H_{3,i}(x) &= \sum_{j=0}^{1} y_{i+j}\, H_{1,j}(x) + \sum_{j=0}^{1} y'_{i+j}\, \hat{H}_{1,j}(x), \\
H_{1,j}(x) &= [1 - 2(x - x_{i+j})L'_{1,j}(x_{i+j})](L_{1,j}(x))^2, \\
\hat{H}_{1,j}(x) &= (x - x_{i+j})(L_{1,j}(x))^2.
\end{aligned}
$$

Método de Hermite Segmentado

- Esos polinomios son:

$$
\begin{aligned}
H_{1,0}(x) &= [1 - 2(x - x_i)L'_{1,0}(x_i)](L_{1,0}(x))^2, \\
H_{1,1}(x) &= [1 - 2(x - x_{i+1})L'_{1,1}(x_{i+1})](L_{1,1}(x))^2, \\
\hat{H}_{1,0}(x) &= (x - x_i)(L_{1,0}(x))^2, \\
\hat{H}_{1,1}(x) &= (x - x_{i+1})(L_{1,1}(x))^2.
\end{aligned}
$$

- Generamos *n* segmentos con parábolas cúbicas, que pasan por y_i, y'_i, y_{i+1} y y'_{i+1}.
- Al igual que para los **Trazadores Cúbicos**, obtenemos un conjunto de curvas entre x_0 y x_n.

Error y convergencia

- Contamos con varias formas de interpolar con polinomios un conjunto de datos.
- ¿Qué error se comete al aplicarlos?
- Lo más probable: no conocemos la función que vincula a los x_i e y_i (y tampoco a y_i').
- El error que estimemos será teórico.
- Para determinar este error, supongamos por un momento que conocemos la función f(x).
- Estimaremos el error cuando interpolamos por **Lagrange** (**Newton**) y por **Hermite**.
- Y a partir de estos errores, estimaremos el de **Trazadores Cúbicos**.

Error y convergencia

- Para el caso del **Método de Lagrange** tenemos:

$$E(x) = f(x) - P_n(x) = \frac{f^{\langle n+1 \rangle}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i),$$

con $x = x_0, x_1, \ldots, x_n \in (a,b),\ \xi \in (a,b)$ y $f(x) \in C^{n+1}(a,b)$.

- Para el caso del **Método de Hermite** tenemos:

$$E(x) = f(x) - H_{2n+1}(x) = \frac{f^{\langle 2(n+1) \rangle}(\xi)}{[2(n+1)]!} \prod_{i=0}^{n} (x - x_i)^2,$$

con $x = x_0, x_1, \ldots, x_n \in (a,b),\ \xi \in (a,b)$ y $f(x) \in C^{2(n+1)}(a,b)$.

Error y convergencia

- Para **Trazadores Cúbicos** («**Splines**») podemos estimar el error con la siguiente expresión:

$$E(x) = f(x) - S(x) = \frac{5\, f^{\langle iv \rangle}(\xi)}{384} \max(h_i)^4,$$

con $x = x_0, x_1, \ldots, x_n \in (a, b),\ \xi \in (a, b)$, $h_i = x_{i+1} - x_i$ y $f(x) \in C^4(a, b)$. En este caso, $S(x)$ representa al conjunto de segmentos $S_i(x)$.

- Para el **Método de Hermite Segmentado** tenemos:

$$E(x) = f(x) - H_3(x) = \frac{f^{\langle iv \rangle}(\xi)}{384} \max(h_i)^4,$$

con $x = x_0, x_1, \ldots, x_n \in (a, b),\ \xi \in (a, b)$, $h_i = x_{i+1} - x_i$ y $f(x) \in C^4(a, b)$. En este caso, $H_(x)$ representa al conjunto de segmentos $H_{3,i}(x)$.

Interpolación trigonométrica

- Hemos visto varias formas de obtener funciones mediante polinomios a partir de disponer de datos discretos.
- Analizamos los casos con conjuntos del tipo (x_j, y_j), pares de puntos unívocamente relacionados. También cuando el conjunto era (x_j, y_j, y'_j).
- En todos los casos, la función debe pasar por los puntos, es decir, se debe cumplir que $P(x_j) = y_j$, entonces el modelo para obtener la función es la **Interpolación**.
- En ambos casos, es común el uso de polinomios tradicionales:

$$P(x) = a_0 + a_1\, x + a_2\, x^2 + \cdots + a_n\, x^n = \sum_{j=0}^{n} a_j\, x^j. \tag{1}$$

Interpolación trigonométrica

- No siempre la interpolación polinómica tradicional es el mejor modelo para aproximar una función.
- Esto queda evidente cuando los datos discretos muestran cierta periodicidad.
- Para este tipo de casos, un modelo muy usado es el de los *polinomios trigonométricos* como el siguiente:

$$H(x) = a_0 + a_1 \operatorname{sen}(x) + a_2 sen(2x) + \cdots + a_n \operatorname{sen}(nx) = a_0 + \sum_{j=1}^{n} a_j \ \operatorname{sen}(j\,x), \quad (2)$$

donde las incógnitas son los coeficientes a_j (incluye a a_0, que puede asumirse que afecta a $\operatorname{sen}(j\,x)$ cuando $j = 0$).

Aproximación con polinomios trigonométricos

- Una forma más extendida es usar un esquema derivado de la serie de Fourier:

$$\begin{aligned}
\phi_0(x) &= \frac{1}{2}, && \\
\phi_k(x) &= \cos kx && \text{con } k = 1, 2, \ldots, n, \\
\phi_{n+k}(x) &= \operatorname{sen} kx && \text{con } k = 1, 2, \ldots, n-1.
\end{aligned}$$

- Si se aplican en el intervalo $[-\pi, \pi]$, las funciones antes vistas son ortogonales.
- Este esquema se usa mucho para la aproximación de funciones.
- Un modelo con un esquema de funciones ortogonales, pero con polinomios, es usado para obtener el modelo de *Cuadratura de Gauss-Legendre*, al usar los *polinomios de Legendre* que son ortogonales en el intervalo $[-1; 1]$.

Aproximación con polinomios trigonométricos

- En este caso, la aproximación de una función cualquiera $f(x)$ la haremos con este modelo:

$$S_n(x) = \frac{a_0}{2} + a_n \cos nx + \sum_{k=1}^{n-1} (a_k \cos kx + b_k \operatorname{sen} kx), \tag{3}$$

con n «grado» del polinomio trigonométrico.

- El hecho de que las funciones sean ortogonales en el intervalo $[-\pi, \pi]$ facilita obtener los coeficientes a_k y b_k con $k = 0; 1; 2 \ldots, n$.
- Los coeficientes a_k los obtenemos con:

$$a_k = \frac{\int_{-\pi}^{\pi} f(x) \cos kx \, \mathrm{d}x}{\int_{-\pi}^{\pi} \cos^2 kx \, \mathrm{d}x} = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos kx \, \mathrm{d}x, \tag{4}$$

con $k = 1, 2, \ldots, n$.

Aproximación con polinomios trigonométricos

- Los coeficientes b_k los obtenemos con:

$$b_k = \frac{\int_{-\pi}^{\pi} f(x) \operatorname{sen} kx \, \mathrm{d}x}{\int_{-\pi}^{\pi} \operatorname{sen}^2 kx \, \mathrm{d}x} = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \operatorname{sen} kx \, \mathrm{d}x, \tag{5}$$

  con $k = 1, 2, \ldots, n-1$.
- Esto mismo podemos adaptarlo para la aproximación discreta, también en el intervalo $[-\pi, \pi]$.
- Podemos efectuar este procedimiento con $2m$ pares de puntos:

$$\{(x_j, y_j)\}_{j=0}^{2m-1}.$$

- La discretización la haremos con esta distribución:

$$x_j = -\pi + \frac{j}{m}\pi \quad \text{con } j = 0; 1; 2 \ldots; 2m-1. \tag{6}$$

Aproximación con polinomios trigonométricos

- Como usamos una *Aproximación por Mínimos Cuadrados*, se debe cumplir que:

$$E(S_n) = \sum_{j=0}^{2m-1} [y_j - S_n(x_j)]^2, \tag{7}$$

sea mínimo.

- Al usar funciones ortogonales en $[-\pi, \pi]$ se cumple que:

$$\sum_{j=0}^{2m-1} \phi_k(x_j)\, \phi_l(x_j) = 0 \quad \text{con } k \neq l, \tag{8}$$

y que

$$\sum_{j=0}^{2m-1} \phi_k(x_j)\, \phi_k(x_j) = m. \tag{9}$$

Aproximación con polinomios trigonométricos

- Con estas propiedades de las funciones trigonométricas en el intervalo visto, podemos calcular los coeficientes a_k y b_k:

$$a_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \cos kx_j \qquad \text{con } k = 0; 1; 2; \ldots; n, \tag{10}$$

y

$$b_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \operatorname{sen} kx_j \qquad \text{con } k = 1; 2; \ldots; n-1. \tag{11}$$

- Si el intervalo es $[a, b]$ en vez de $[-\pi, \pi]$, debemos crear un nuevo conjunto con esta transformación lineal para aplicar el método:

$$z_j = \frac{\pi}{b-a}[2x_j - (a+b)].$$

Transformada rápida de Fourier

- Si en vez de aproximar un conjunto de puntos discretos mediante un esquema de mínimos cuadrados buscamos interpolar ese mismo conjunto, el procedimiento debe modificarse.
- Para ese fin tomaremos la siguiente función:

$$S_m(x) = \frac{a_0 + a_m \cos mx}{2} + \sum_{k=1}^{m-1} (a_k \cos kx + b_k \operatorname{sen} kx). \tag{12}$$

- Como ahora el modelo es de interpolación, se debe cumplir que:

$$S_m(x_j) = y_j. \tag{13}$$

Transformada rápida de Fourier

- Así, los coeficientes pueden obtenerse con:

$$a_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \cos kx_j \qquad \text{con } k = 0; 1; 2; \ldots; n, \tag{14}$$

y

$$b_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \operatorname{sen} kx_j \qquad \text{con } k = 1; 2; \ldots; n-1. \tag{15}$$

- Con estas expresiones se necesitan muchos cálculos para obtener todos los coeficientes.

Transformada rápida de Fourier

- En 1965, James Cooley y John Tukey ([Cooley & Tukey, 1965]) publicaron un artículo con un algoritmo que reduce la cantidad de cálculos para obtener la función interpolante.
- El algoritmo parte de calcular los coeficientes c_k complejos en

$$S_m(x) = \frac{1}{m} \sum_{k=0}^{2m-1} c_k \, e^{\mathrm{i}kx}, \tag{16}$$

donde

$$c_k = \sum_{j=0}^{2m-1} y_j e^{\mathrm{i}k\pi \frac{j}{m}}, \tag{17}$$

para cada $k = 0, 1, 2, \ldots, 2m-1$.

Transformada rápida de Fourier

- El algoritmo se basa en el siguiente desarrollo:

$$\begin{aligned} \frac{1}{m} c_k (-1)^k = \frac{1}{m} c_k \, e^{-\mathrm{i}\pi k} &= \frac{1}{m} \sum_{j=0}^{2m-1} y_j \, e^{\mathrm{i}k\pi \frac{j}{m}} \, e^{-\mathrm{i}k\pi} \\ &= \frac{1}{m} \sum_{j=0}^{2m-1} y_j \, e^{\mathrm{i}k\pi(-1+\frac{j}{m})} \\ &= \frac{1}{m} \sum_{j=0}^{2m-1} y_j (\cos kx_j + \mathrm{i} \operatorname{sen} kx_j). \end{aligned} \tag{18}$$

- Una vez obtenidos los coeficientes c_k, los coeficientes a_k y b_k se obtienen con ayuda de la *Fórmula de Euler*: $e^{\mathrm{i}z} = \cos z + \mathrm{i} \operatorname{sen} z$. Así, tenemos que:

$$a_k + \mathrm{i} b_k = \frac{(-1)^k}{m} c_k. \tag{19}$$

Bibliografía

- Burden, R. L., Faires, J. D. & Burden, A. M.
  Análisis Numérico.
  Décima Edición. CENGAGE Learning, 2016.
- Cooley, J. W. & Tukey. J. W.
  An algorithm for the machine calculation of complex Fourier series.
  Mathematics of Computation. American Mathematical Society, 1965.
- Trefethen, L. N. & Berrut, J. P.
  Barycentric Lagrange Interpolation.
  2004.
- Higham, N. J.
  The numerical stability of Barycentric Lagrange Interpolation.
  IMA. 2004.