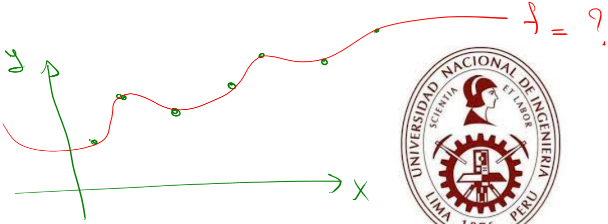
Interpolación trigonométrica

CM4F1

Ángel Enrique Ramírez Gutiérrez
aramirezg@uni.edu.pe

Escuela Profesional de Matemática
Universidad Nacional de Ingeniería

26 de julio de 2022



Interpolación polinomial $p \in P_n$

Contenidos

1. Interpolación trigonométrica
   1.1. Interpolación trigonométrica con nodos igualmente espaciados

2. Algoritmo de Cooley-Tukey

Contenido

1. Interpolación trigonométrica
1.1. Interpolación trigonométrica con nodos igualmente espaciados

2. Algoritmo de Cooley-Tukey

Interpolación trigonométrica

Las funciones periódicas son una clase de funciones muy importantes en diversas aplicaciones.

Definición:

Una función $f(t)$ se dice **periódica** de periodo τ si:

$$f(t+\tau) = f(t), \quad -\infty < t < \infty$$

Las funciones periódicas más conocidas son las trigonómetricas, cuyo periodo habitual es $\tau = 2\pi$.

Así, que sin pérdida de generalidad podemos suponer que el periodo es 2π, pues en caso que no lo sea basta con hacer un reescalamiento de la variable independiente.

Formulación del problema

Es posible usar en lugar de polinomios usuales:

$$p(x) = c_0 + \sum_{k=1}^{n} c_k x^k$$

los denominados **polinomios trigonométricos** de la forma:

$$p(\theta) = a_0 + \sum_{j=1}^{n} a_j cos(j\theta) + b_j sen(j\theta).$$

Note que esta expresión tiene $2n+1$ coeficientes indeterminados $a_0, a_1, \ldots, a_n, b_1, \ldots, b_n$ y el objetivo es calcular dichos coeficientes tal que:

$$p(\theta_k) = y_k, \quad k = 1, 2, \ldots, 2n+1$$

Como las funciones trigonométricas son periódicas de periodo 2π, tiene sentido asumir que:

$$0 \leq \theta_0 < \theta_1 < \ldots < \theta_{2n} < 2\pi.$$

Definición:

Un **polinomio trigonométrico** de grado n es de la forma:

$$p_n(\theta) = a_0 + \sum_{j=1}^{n} a_j cos(j\theta) + b_j sen(j\theta). \tag{1}$$

Por tanto, el **problema de interpolación trigonométrico** consiste en buscar un polinomio trigonométrico de la forma (1) que interpola a una función $f(\theta)$ periódica de periodo 2π en $2n+1$ nodos distintos en $[0, 2\pi]$, es decir:

$$p_n(\theta_k) = f(\theta_k), \quad k = 0, 1, 2, \ldots, 2n; \quad 0 \leq \theta_0 < \theta_1 < \ldots < \theta_{2n} < 2\pi. \tag{2}$$

Interpolación trigonométrica

UNIVERSIDAD NACIONAL DE INGENIERÍA

En principio, se puede abordar directamente este problema de forma análoga a la interpolación de Lagrange, donde los coeficientes quedan determinados por las ecuaciones:

$$p(\theta_k) = y_k \quad k = 1, \ldots, N$$

que definen un sistema de $2n+1$ ecuaciones lineales y que tiene solución única cuando los puntos θ_k son distintos. Esta solución viene dada por:

$$p(\theta) = \sum_{k=1}^{2n+1} y_k \prod_{m=1, m\neq k}^{2n+1} \frac{sen\left(\frac{1}{2}(\theta - \theta_m)\right)}{sen\left(\frac{1}{2}(\theta_k - \theta_m)\right)}.$$

Sin embargo, observe que el cálculo se vuelve un poco engorroso, así que se suele utilizar una solución alternativa.

$$P(\theta) = a_0 + \sum_{j=1}^{n} a_j \cos(j\theta) + b_j \operatorname{Sen}(j\theta)$$

$$= a_0 + \sum_{j=1}^{n} a_j \left[\frac{e^{i(j\theta)} + e^{-i(j\theta)}}{2}\right] + b_j \left[\frac{e^{i(j\theta)} - e^{-i(j\theta)}}{2i}\right]\frac{i}{i}$$

$$P(\theta) = a_0 + \sum_{j=1}^{n} \left(\frac{a_j - i b_j}{2}\right) e^{ij\theta} + \left(\frac{a_j + i b_j}{2}\right) e^{-ij\theta}$$

$$= a_0 + \sum_{j=1}^{n} \left(\frac{a_j - i b_j}{2}\right) e^{ij\theta} + \underbrace{\sum_{j=1}^{n} \left(\frac{a_j + i b_j}{2}\right) e^{-ij\theta}}_{\parallel}$$

$$\sum_{j=-n}^{-1} \left(\frac{a_{-j} + i b_{-j}}{2}\right) e^{ij\theta}$$

$$\Rightarrow \boxed{P(\theta) = \sum_{j=-n}^{n} c_j e^{ij\theta}}$$ donde: $c_j = \dfrac{a_j - i b_j}{2}$, $j = 1, \dots, n$

$c_j = \dfrac{a_{-j} + i b_{-j}}{2}$, $j = -n, \dots, -1$

$c_0 = a_0$

Aplicando la fórmula de Euler para números complejos:

$$e^{i\theta} = cos(\theta) + isen(\theta), \quad i = \sqrt{-1}, \tag{3}$$

se tiene:

$$cos(\theta) = \frac{e^{i\theta} + e^{-i\theta}}{2}, \quad sen(\theta) = \frac{e^{i\theta} - e^{-i\theta}}{2i}.$$

LLevando esto a (1) resulta:

$$p_n(\theta) = \sum_{j=-n}^{n} c_j e^{ij\theta}, \tag{4}$$

donde:

$$c_0 = a_0, \quad c_{-j} = \frac{a_j + ib_j}{2}, \quad c_j = \frac{a_j - ib_j}{2}, \quad 1 \leq j \leq n.$$

Por tanto, si calculamos los $\{c_j\}_{j=-n}^{n}$ se obtienen directamente los coeficientes del polinomio trigonométrico buscado.

Interpolación trigonométrica

UNIVERSIDAD NACIONAL DE INGENIERÍA

Para el cálculo de los coeficientes cuando el índice es negativo se multiplica $p_n(\theta)$ por $e^{in\theta}$, es decir:

$$q_{2n}(\theta) = e^{in\theta} p_n(\theta) = \sum_{j=0}^{2n} d_j e^{ij\theta}, \quad d_j = c_{j-n}, \quad 0 \leq j \leq 2n.$$

Para calcular los coeficientes d_j se hace un pequeño *viaje* al campo complejo. Así generalizamos (4) mediante el polinomio complejo:

$$Q_{2n}(z) = \sum_{j=0}^{2n} d_j z^j \tag{5}$$

de tal forma que cuando $\boxed{z = e^{i\theta}}$ resulta:

$$Q_{2n}(e^{i\theta}) = q_{2n}(\theta).$$

Interpolación trigonométrica

UNIVERSIDAD NACIONAL DE INGENIERÍA

Denotando:

$$z_k = e^{i\theta_k}, \quad k = 0, 1, \ldots, 2n,$$

resulta que el problema de interpolación trigonométrica (2) se transforma en encontrar los d_j que hagan que se verifique:

$$Q_{2n(z_k)} = z_k^n f(\theta_k), \quad k = 0, 1, \ldots, 2n \tag{6}$$

es decir, en buscar un polinomio (en el campo complejo) de grado $\leq 2n$ que en $2n+1$ nodos distintos $\{z_k\}_{k=0}^{2n}$ pase por las $2n+1$ ordenadas

$$\{z_0^n f(\theta_0), z_1^n f(\theta_1), \ldots, z_{2n}^n f(\theta_{2n})\}.$$

Interpolación trigonométrica

Del teorema de existencia y unicidad del polinomio interpolador queda garantizado que existe un único polinomio trigonométrico $p_n(\theta)$ de la forma (1) que interpola a la función periódica $f(\theta)$ en los $2n+1$ nodos distintos (2).

UNIVERSIDAD NACIONAL DE INGENIERÍA

En general, el cálculo de los coeficientes del polinomio interpolador trigonométrico es díficil de realizar. Una forma práctica de calcularlos es el caso cuando los nodos están **igualmente espaciados** en el intervalo $[0, 2\pi]$. Veamos:

$$\boxed{\theta_k = \frac{2k\pi}{2n+1}} \tag{7}$$

y teniendo en cuenta el siguientes resultado:

Teorema:

Para cualquier entero l se cumple:

$$\sum_{m=0}^{2n} e^{il\theta_m} = \begin{cases} 2n+1, & \text{si} \quad e^{i\theta_l} = 1 \\ 0, & \text{si} \quad e^{i\theta_l} \neq 1 \end{cases} \tag{8}$$

Para el cálculo de los coeficientes del polinomio trigonométrico interpolador (1)–(2) se tiene el siguiente resultado:

Teorema: [Transformada Discreta de Fourier]

Los coeficientes c_j de (4) vienen dados por:

$$c_j = \frac{1}{2n+1}\sum_{k=0}^{2n} e^{-ij\theta_k} f(\theta_k), \quad j = -n, -n+1, \ldots, n-1, n. \tag{9}$$

Los coeficientes a_j, b_j del polinomio trigonométrico interpolador (1)–(2) se calculan directamente del modo siguiente:

$$a_0 = c_0, \quad a_j = 2Re(c_j) = c_j + c_{-j}, \quad b_j = 2Im(c_j) = -i(c_j - c_{-j}), \quad j = 1, \ldots, n.$$

Contenido

1. Interpolación trigonométrica
1.1. Interpolación trigonométrica con nodos igualmente espaciados

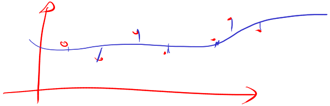
2. Algoritmo de Cooley-Tukey

Transformada rápida de Fourier

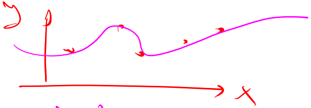
Sea E_k la función tal que $E_k(x) = e^{ikx}$ sabemos que el polinomio exponencial que interpola f en nodos equiespaciados $x_j = 2\pi j/N$ esta dado por

$$P = \sum_{k=0}^{N-1} c_k E_k, \quad c_k = \langle f, E_k \rangle_N$$

donde $\langle f, g \rangle_N = \dfrac{1}{N} \sum_{j=0}^{N-1} f(2\pi j/N) g(2\pi j/N)$.
La manera eficiente de calcular c_k es el algoritmo de transformada rápida de Fourier.



Burden



Aproximación por mínimos cuadrados

Interpolación

Transformada rápida de Fourier

UNIVERSIDAD NACIONAL DE INGENIERÍA

Teorema:

Sean p y q polinomios exponenciales de grado $\leq n-1$, tales que para $x_j = \pi j/n$ tenemos

$$p(x_{2j}) = f(x_{2j}), \quad q(x_{2j}) = f(x_{2j+1}), \quad 0 \leq j \leq n-1$$

entonces el polinomio exponencial de grado $\leq 2n-1$ que interpola f en los puntos $x_0, x_1, \ldots, x_{2n-1}$ esta dado por

$$P(x) = \frac{1}{2}(1+e^{i\pi x})p(x) + \frac{1}{2}(1-e^{i\pi x})q(x-\pi/n)$$

Demostración (cont.)

UNIVERSIDAD NACIONAL DE INGENIERÍA

Como p y q son de grado $\leq n-1$ y e^{inx} es de grado n entonces el grado de P es menor o igual que $2n-1$. Ademas P interpola f pues

$x_j = \frac{j\pi}{n}$

$$P(x_j) = \frac{1}{2}(1 + e^{i\pi x_j})p(x_j) + \frac{1}{2}(1 - e^{i\pi x_j})q(x_j - \pi/n)$$

como $e^{i\pi x_j} = e^{\pi j i} = \cos(\pi j) = (-1)^j$, entonces para j par:

e^{inx_j}

$$P(x_j) = \frac{1}{2}(1+1)p(x_j) + \frac{1}{2}(1-1)q(x_j - \pi/n) = p(x_j) = f(x_j)$$

como $e^{i\pi x_j} = e^{\pi j i} = \cos(\pi j) = (-1)^j$, entonces para j impar:

$$P(x_j) = \frac{1}{2}(1-1)p(x_j) + \frac{1}{2}(1+1)q(x_j - \pi/n) = q(x_j - \pi/n) = q(x_{j-1}) = f(x_j)$$

Transformada rápida de Fourier

UNIVERSIDAD NACIONAL DE INGENIERÍA

Teorema:

Sean los coeficientes de los polinomios descritos en el teorema anterior:

$$p = \sum_{j=0}^{n-1} \alpha_j E_j, \quad q = \sum_{j=0}^{n-1} \beta_j E_j, \quad P = \sum_{j=0}^{2n-1} \gamma_j E_j,$$

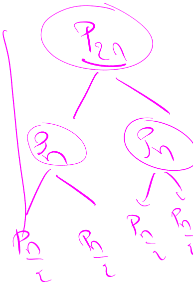
entonces para $0 \leq j \leq n-1$

$$\gamma_j = \frac{1}{2}\alpha_j + \frac{1}{2}e^{-ij\pi/n}\beta_j$$

$$\gamma_{j+n} = \frac{1}{2}\alpha_j - \frac{1}{2}e^{-ij\pi/n}\beta_j$$

Demostración:

UNIVERSIDAD NACIONAL DE INGENIERÍA



$$P(x) = \frac{1}{2}(1+e^{i\pi x})\sum_{j=0}^{n-1}\alpha_j E_j(x) + \frac{1}{2}(1-e^{i\pi x})\sum_{j=0}^{n-1}\beta_j E_j(x-\pi/n)$$

$$P(x) = \frac{1}{2}\sum_{j=0}^{n-1}(\alpha_j+\beta_j e^{-i\pi j/n})E_j(x) + (\alpha_j-\beta_j e^{-i\pi j/n})E_{n+j}(x)$$

$\square$

Lo que muestra el resultado anterior es que podemos interpolar f en $2n$ nodos si primero conocemos los polinomios de interpolación en dos conjuntos disjuntos de n nodos.

Transformada rápida de Fourier

Sea $R(n)$ el mínimo número de multiplicaciones necesarias para calcular lo coeficientes de un polinomio de interpolación exponencial en los nodos $\{2\pi j/n; 0 \leq j \leq n-1\}$

n nodos

Teorema:

$R(2^m) \leq m2^m$

Demostración:

Por el teorema anterior para calcular γ_j se requieren $2n$ multiplicaciones: n para $\frac{1}{2}\alpha_j$ y n para $(\frac{1}{2}e^{-ij\pi/n})\beta_j$ suponiendo que los $(\frac{1}{2}e^{-ij\pi/n})$ ya están almacenados en memoria. Como los α_j se calcularon en un conjunto de n nodos entonces el numero de multiplicaciones sera a los mas $R(n)$, lo mismo es cierto para los β_j, en total

$$R(2n) \leq R(n) + R(n) + 2n = 2R(n) + 2n \quad (1)$$

Demostración (cont.)

Procedemos por inducción, si $m = 1$ se requieres 2 multiplicaciones por lo tanto el enunciado del teorema se cumple. Suponemos que el teorema es cierto para $m = k$, $R(2^k) \leq k2^k$, entonces por (1)

$$R(2(2^k)) \leq 2R(2^k) + 2(2^k) \leq 2k2^k + 2(2^k) = (k+1)2^{k+1}$$

□

Como consecuencia si $N = 2^m$, el costo de calcular el polinomio de interpolación es $O(N \log_2 N)$.

Definición:

Definimos el operador lineal L_n tal que $L_n f$ es el polinomio de grado $n-1$ que interpola a f en los nodos $2\pi j/n$, $0 \leq j \leq n-1$. T_h es el operador de traslación

$$(T_h f)(x) = f(x+h)$$

con esta notación tenemos que

$$L_n f = \sum_{k=0}^{n-1} \langle f, E_k \rangle_n E_k$$

$$P = L_{2n}f, \quad p = L_n f, \quad q = L_n T_{\pi/n} f$$

Definimos ademas para $N = 2^m$

$$P_k^{(n)} = L_{2^n} T_{2k\pi/N} f \quad 0 \leq n \leq m, 0 \leq k \leq 2^{m-n} - 1$$

podemos entender $P_k^{(n)}$ como el polinomio de grado $2^n - 1$ que interpola a f de la manera siguiente

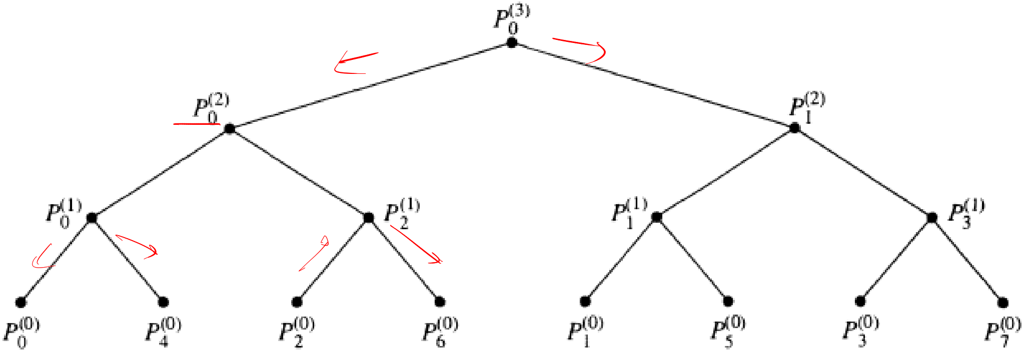
$$P_k^{(n)}\left(\frac{2\pi j}{2^n}\right) = f\left(\frac{2\pi k}{N} + \frac{2\pi j}{2^n}\right), 0 \leq j \leq 2^n - 1$$

Los conjunto de nodos $(2\pi k/N) + (2\pi j/2^n)$ son disjuntos para valores diferentes de k , entonces

$$P_k^{(n+1)}(x) = \frac{1}{2}(1 + e^{i2^n x})P_k^{(n)}(x) + \frac{1}{2}(1 - e^{i2^n x})P_{k+2^{m-n-1}}^{(n)}(x - \frac{\pi}{2^n})$$

Transformada rápida de Fourier

Este proceso constructivo se puede ilustrar en el siguiente diagrama de árbol



Transformada rápida de Fourier

UNIVERSIDAD NACIONAL DE INGENIERÍA

Teorema:

Si

$$P_k^{(n)}(x) = \sum_{j=0}^{2^n-1} A_{kj}^{(n)} E_j(x) = \sum_{j=0}^{2^n-1} A_{kj}^{(n)} e^{ijx}$$

por el teorema 2 podemos calcular los coeficientes $A_{kj}^{(n+1)}$ en base a los $A_{kj}^{(n)}$

$$A_{kj}^{(n+1)} = \frac{1}{2}\left(A_{kj}^{(n)} + e^{-ij\pi/2^n} A_{k+2^{m-n-1},j}^{(n)}\right)$$

$$A_{k,j+2^n}^{(n+1)} = \frac{1}{2}\left(A_{kj}^{(n)} - e^{-ij\pi/2^n} A_{k+2^{m-n-1},j}^{(n)}\right)$$

Observación:

UNIVERSIDAD NACIONAL DE INGENIERÍA

Si C es un vector que almacena $A^{(n)}$ y D es un vector que almacena $A^{(n+1)}$ entonces

$$C(2^n k + j) \leftarrow A_{kj}^{(n)}$$

$$D(2^n k + j) \leftarrow A_{kj}^{(n+1)}$$

los factores $Z(j) = e^{-2\pi ij/N}$ se calculan al inicio, y aprovechamos que

$$Z(j2^{m-n-1}) = e^{-ij\pi/2^n}$$

Algoritmo de transformada rápida (FFT)

input m

$N \leftarrow 2^m$

$w \leftarrow e^{-2\pi i/N}$

for $k = 0$ **to** $N-1$ **do**

  $Z(k) \leftarrow w^k$

  $C(k) \leftarrow f(2\pi k/N)$

end do

for $n = 0$ **to** $m-1$ **do**

  for $k = 0$ **to** $2^{m-n-1}-1$ **do**

    for $j = 0$ **to** $2^n - 1$ **do**

      $u \leftarrow C(2^n k + j)$

      $v \leftarrow Z(j2^{m-n-1})C(2^n k + 2^{m-1} + j)$

      $D(2^{n+1}k + j) \leftarrow (u+v)/2$

      $D(2^{n+1}k + j + 2^n) \leftarrow (u-v)/2$

    end do

  end do

  for $j = 0$ **to** $N-1$ **do**

    $C(j) \leftarrow D(j)$

  end do

end do

output $C(0), C(1), \ldots, C(N)$

Ejemplo:

Considere la siguiente tabla de datos

x	0	$\frac{\pi}{2}$	π	$\frac{3\pi}{2}$
y	$f(0)$	$f(\frac{\pi}{2})$	$f(\pi)$	$f(\frac{3\pi}{2})$

y aplique el algoritmo de la transformada rápida de Fourier.

Solución:

Tenemos $n = 2$, $N = 2^n = 4$ entonces P tiene la forma $P(x) = \sum_{j=0}^{3} \gamma_j E_j(x)$, y los coeficientes γ_j se pueden calcular como

$$\gamma_0 = \frac{1}{2}(\alpha_0 + \beta_0),$$

$$\gamma_1 = \frac{1}{2}(\alpha_1 + \beta_1 e^{-i\pi/2}) = \frac{1}{2}(\alpha_1 - i\beta_1),$$

$$\gamma_2 = \frac{1}{2}(\alpha_0 - \beta_0),$$

$$\gamma_3 = \frac{1}{2}(\alpha_1 - \beta_1 e^{-i\pi/2}) = \frac{1}{2}(\alpha_1 + i\beta_1)$$

Solución (cont.)

UNIVERSIDAD NACIONAL DE INGENIERÍA

la etapa previa involucra a p y q, dos polinomios de interpolación que interpolan a f en la mitad de nodos,

$$p(x) = \sum_{j=0}^{1} \alpha_j E_j(x), \quad p(x_0) = f(x_0), \quad p(x_2) = f(x_2),$$

$$p(x) = \frac{1}{2}(f(x_0) + f(x_2)) + \frac{1}{2}(f(x_0) - f(x_2))e^{i\pi x}$$

$$q(x) = \sum_{j=0}^{1} \beta_j E_j(x), \quad q(x_0) = f(x_1), \quad q(x_2) = f(x_3)$$

$$q(x) = \frac{1}{2}(f(x_1) + f(x_3)) + \frac{1}{2}(f(x_1) - f(x_3))e^{i\pi x}$$

luego

$$\alpha_0 = \frac{1}{2}(f(x_0) + f(x_2)), \quad \alpha_1 = \frac{1}{2}(f(x_0) - f(x_2))$$

$$\beta_0 = \frac{1}{2}(f(x_1) + f(x_3)), \quad \beta_1 = \frac{1}{2}(f(x_1) - f(x_3))$$

Ejemplo

UNIVERSIDAD NACIONAL DE INGENIERÍA

```python
import numpy as np
from numpy import pi as PI

def fft1(f,N):
    w = np.exp(-2*PI*1j/N) # 1j es la variable compleja
    C =  np.zeros( (N,),dtype=complex)
    xx = np.linspace(0,2*PI*(N-1.0)/N,N)
    C[:]= f(xx)
    D = np.zeros_like(C)
    Z = np.array([w**k for k in range(N)])
    for n in range(m):
        for k in range(2**(m-n-1)):
            for j in range(2**(n)):
                u = C[2**n * k + j]
                v = Z[j*2**(m-n-1)]*C[2**n * k + 2**(m-1) + j]
                D[2**(n+1) * k + j] = (u+v)/2
                D[2**(n+1) * k + j + 2**n] = (u-v)/2
        C[:]=D[:]
    return C
```

Ejemplo:

UNIVERSIDAD NACIONAL DE INGENIERÍA

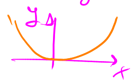
Determine el polinomio interpolante trigonométrico de grado 4 en $[0,2]$ que interpola a $f(x) = x^4 - 3x^3 + 2x^2 - \tan(x(x-2))$ en los nodos $\{j/4\}_{j=0}^{7}$.

Solución: En primer lugar hacemos $z = \pi x$, y aplicamos el algoritmo de transformada rápida a $f(\frac{z}{\pi})$ y los nodos $\{j\pi/4\}_{j=0}^{7}$.

Extensión PAR e IMPAR de una función.

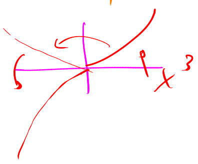
- Función par: $f(x) = f(-x)$ → Ej: $f(x) = x^2$

El eje y es un eje de simetría



- Función impar: $f(-x) = -f(x)$ → Ej: $f(x) = x^3$

Es simétrica respecto al origen.



Sea f una función definida en un intervalo $0 \le x \le L$

Se define:

$$h(x) = \begin{cases} f(x), & 0 \le x \le L \\ f(-x), & -L < x < 0 \end{cases}$$

se cumple: $h(x) = h(x + 2L)$

Observe que h es PAR y PERIÓDICA.

Análogamente, se define:

$$g(x) = \begin{cases} -f(-x), & -L \le x \le 0 \\ f(x), & 0 \le x \le L \end{cases}$$

Se cumple: $g(x) = g(x + 2L)$

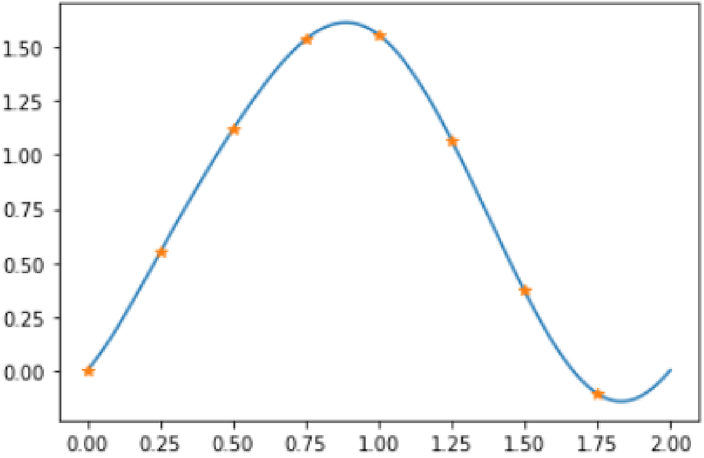
Observe que g es impar y periódica de periodo $2L$.

Implementación FFT

```
g = lambda x : x**4-3*x**3+2*x**2-np.tan(x*(x-2))
a = 0
b = 2
z = lambda x : a+x*(b-a)/(2*PI)
f = lambda x : g(z(x))
m=3
N=2**m
C = fft1(f,N)
print("C=",C)
A = 2*C[0:N//2].real
A[0]  = 0.5*A[0]
B = -2*C[0:N//2].imag
print("A=",A);print("B=",B)
import matplotlib.pyplot as plt
xx = np.linspace(0,2*PI,100)
yy = sum([ (A[k]*np.cos(k*xx)+B[k]*np.sin(k*xx))[:,None] for k in range(len(A))])
tt = np.linspace(0,2*PI*(N-1.0)/N,N)
plt.plot(z(xx),yy,z(tt),f(tt),'*')
```

```
C= [ 7.61978706e-01+0.j          -3.85920410e-01-0.19318689j
  8.65185060e-03-0.0234375j  -3.43152066e-03-0.00568689j
 -5.78544889e-04+0.j          -3.43152066e-03+0.00568689j
  8.65185060e-03+0.0234375j  -3.85920410e-01+0.19318689j]
A= [ 0.76197871 -0.77184082  0.0173037  -0.00686304]
B= [-0.          0.38637378  0.046875    0.01137378]
```

Resultados numéricos



Bibliografía

Referencias I

- J. L. DE LA FUENTE
  Técnicas de calculo para sistemas de ecuaciones, programación lineal y programación entera.
- G.H. Golub and C.F. Van Loan.
  Matrix Computations, 4th Edition
- Biswa Datta
  Numerical methods for linear control systems.
- Dennis, J. E. and Schnabel, Robert B.
  Numerical Methods for Unconstrained Optimization and Nonlinear Equations
- Varga, Richard S.
  Geršgorin and His Circles.