Transformada discreta de coseno. El Teorema de interpolación de la transformada discreta de coseno indica lo siguiente:

Thm 1. *Sea* $\mathbf{x} = [x_0, x_1, \ldots, x_{n-1}]^T$ *un vector con* n *coeficientes reales. Se define* $\mathbf{y} = [y_0, y_1, \ldots, y_{n-1}]^T = C\,\mathbf{x}$*, donde* C *corresponde a la matriz asociada a la Transformada Discreta de Coseno (DCT del inglés Discrete Cosine Transform) de orden* n*. Entonces la función:*

$$P_n(t) = \frac{1}{\sqrt{n}}\,y_0 + \sqrt{\frac{2}{n}} \sum_{k=1}^{n-1} y_k \cos\left(\frac{k\,(2\,t+1)\,\pi}{2\,n}\right),$$

satisface que $P_n(j) = x_j$ *para todo* $j \in \{0, 1, \ldots n-1\}$ *y los coeficientes de la matriz* $C \in \mathbb{R}^{n\times n}$ *se define de la siguiente forma:*

$$C_{i,j} = \sqrt{\frac{2}{n}}\,a_i \cos\left(\frac{i\,(2\,j+1)\,\pi}{2\,n}\right),$$

para $i \in \{0, 1, \ldots n-1\}$ *y* $j \in \{0, 1, \ldots n-1\}$*, donde*

$$a_i = \begin{cases} \dfrac{1}{\sqrt{2}}, & \text{si } i = 0,\\ 1, & \text{si } i \in \{1, 2, \ldots, n-1\}. \end{cases}$$

Por simplicidad se incluye de forma explícita la matriz C,

$$C = \sqrt{\frac{2}{n}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \cdots & \frac{1}{\sqrt{2}} \\ \cos\left(\frac{\pi}{2\,n}\right) & \cos\left(\frac{3\,\pi}{2\,n}\right) & \cdots & \cos\left(\frac{(2\,n-1)\,\pi}{2\,n}\right) \\ \cos\left(\frac{2\,\pi}{2\,n}\right) & \cos\left(\frac{6\,\pi}{2\,n}\right) & \cdots & \cos\left(\frac{2\,(2\,n-1)\,\pi}{2\,n}\right) \\ \vdots & \vdots & \vdots & \vdots \\ \cos\left(\frac{(n-1)\,\pi}{2\,n}\right) & \cos\left(\frac{(n-1)\,3\,\pi}{2\,n}\right) & \cdots & \cos\left(\frac{(n-1)\,(2\,n-1)\,\pi}{2\,n}\right) \end{bmatrix}$$

En resumen, podemos concluir que necesitamos evaluar recurrentemente la función $f(x) = \cos(\pi\,x)$ al utilizar la DCT. Más aún, es posible notar que para construir la transformada discreta de coseno para $n = 100$ necesitamos evaluar el coeficiente $C_{n-1,n-1} = \cos\left(\frac{(n-1)\,(2\,n-1)\,\pi}{2\,n}\right)$, lo que es $\cos\left(\frac{19701}{200}\,\pi\right) \approx \cos(309.462584341862582954) \approx -0.015707317311820$. Entonces, si consideramos que trabajaremos con vectores de dimensión menor o igual a 100, significa que necesitamos evaluar la función $\cos(\cdot)$ para valores en el intervalo $[0, 100\,\pi]$. Adicionalmente sabemos que,

$$\left|\frac{\mathrm{d}^m f(x)}{\mathrm{d}x^m}\right| = \begin{cases} \pi^m\,|\sin(\pi\,x)|, & \text{si } m \text{ es impar},\\ \pi^m\,|\cos(\pi\,x)|, & \text{si } m \text{ es par}. \end{cases}$$

Preguntas:

(a) Proponga un algoritmo basado en interpolación polinomial que permita evaluar la función $f(x)$ para $x \in [0, 100]$ con la ayuda de $GEN(n)$: un costoso generador de n puntos equispaciados $(\widehat{x}, \widehat{y})$ para la función $cos(x)$ en el intervalo $[0{,}2\pi]$. Debe garantizar un error de interpolación menor o igual a $\varepsilon > 0$ minimizando el costo computaciónal que supone utilizar $GEN(n)$. Usted debe explicar qué se debe hacer para asegurar esto y cómo planea determinar lo que necesite para cumplir con lo solicitado.

Su respuesta debe:

- Explicar claramente cómo **construirá** su interpolador polinomial.
- Explicar claramente cómo se debe **utilizar** su interpolador polinomial.

Solución:

- En esta pregunta se solicita interpolar la función $f(x) = \cos(\pi\,x)$ para $x \in [0, 100]$, lo cual es equivalente a interpolar la función $\cos(x)$ en el intervalo $[0, 100\,\pi]$, más aún sabemos que la función $\cos(x)$ es periódica con periodo $2\,\pi$, por lo que sería suficiente interpolar $\cos(x)$ en $[0, 2\,\pi]$ y utilizar esta propiedad para evaluarla en $[0, 100\,\pi]$. Esto se puede lograr utilizando la función módulo, en particular en NumPy se debería utilizar `x=np.mod(y,2*np.pi)` donde y correspondería a un valor en el intervalo $[0, 100\,\pi]$ y entrega $x \in [0, 2\,\pi]$, el cual se puede evaluar en el interpolador que construiremos.

- Para asegurar un error de interpolación menor o igual a $\varepsilon > 0$ se calcula la siguiente cota del error para la función $\cos(x)$ con interpolación Baricentrica y considerando que los puntos $\widehat{x}_i$ estan equiespaciados *(warning!: notar que se están considerando $n+1$ puntos, esto debido a la definición de la cota superior para equiespaciados)*:

$$|\cos(x) - p(x)| \leq \frac{|(x-\widehat{x}_0)(x-\widehat{x}_1)\dots(x-\widehat{x}_n)|}{(n+1)!}\left|\frac{\mathrm{d}^{(n+1)}(\cos(x))}{\mathrm{d}x^{(n+1)}}\right|,$$

  donde

$$|(x-\widehat{x}_0)(x-\widehat{x}_1)\dots(x-\widehat{x}_n)| \leq \frac{n!}{4}h^{n+1}, h = \frac{b-a}{n}$$
$$\left|\frac{\mathrm{d}^{(n+1)}(\cos(x))}{\mathrm{d}x^{(n+1)}}\right| \leq 1.$$

  Notar que las derivadas de la función $\cos(x)$ oscilan entre $\{\pm\cos(x), \pm\sin x\}$, por lo que es directo obtener que el valor absoluto de todas ellas está acotado por 1. Por lo tanto la cota superior al interpolar con n puntos de equiespaciados en el intervalo $[0, 2\,\pi]$ es la siguiente:

$$|\cos(x) - p(x)| \leq \frac{n!\cdot h^{n+1}\cdot 1}{4\cdot(n+1)!} = \frac{\left(\frac{2\pi}{n}\right)^{n+1}}{4(n+1)} = \frac{\left(\frac{2\pi}{n}\right)\cdot\left(\frac{2\pi}{n}\right)^{n}}{4(n+1)} = \frac{\pi\cdot\left(\frac{2\pi}{n}\right)^{n}}{2n(n+1)}.$$

  Para lograr asegurar que el error sea menor o igual a ε se debe encontrar n tal que cumpla la siguiente ecuación:

$$\frac{\pi\cdot\left(\frac{2\pi}{n}\right)^{n}}{2n(n+1)} \leq \varepsilon.$$

  La cota superior obtenida es lo más pequeño posible considerando los puntos equispaciados y que $\cos(x)$ es periódica, haber utilizado el intervalo $[0, 100\,\pi]$ implicaría que la cota superior sea mucho mayor a la obtenida y solicitada. Con esto ya podemos crear una función que busque el n más pequeño que satisfaga la ecuación para un ε dado, y con ello los n puntos llamando a $GEN(x)$.

- La construcción del polinomio se puede hacer directamente con la interpolación Baricéntrica de la siguiente forma:

$$p(x) = \frac{\displaystyle\sum_{i=1}^{n}\widehat{y}_i\frac{w_i}{(x-\widehat{x}_i)}}{\displaystyle\sum_{i=1}^{n}\frac{w_i}{(x-\widehat{x}_i)}},$$
$$w_i = \frac{1}{l_i(\widehat{x}_i)},$$
$$l_i(\widehat{x}_i) = \prod_{k=1, i\neq k}^{n}(\widehat{x}_i - \widehat{x}_k)$$

- La evaluación de $f(x)$ se puede hacer directamente con $p(\pi\,x)$ si $x \in [0, 2]$, sin embargo, si se necesita evaluar para valores de x mayores a $2\,\pi$ se debe utilizar la función módulo antes descrita, es decir se debe utilizar la siguiente expresión `p(np.mod(np.pi*x,2*np.pi))`, la cual puede evaluarse para $x \in [0, 100]$, que fue justamente lo solicitado.

(b) Explique cómo puede utilizar la interpolación polinomial anterior para evaluar $\sin(\pi\,x)$ sin construir un nuevo interpolador.

En este caso podemos considerar la identidad $\cos\left(\pi\,x + \frac{\pi}{2}\right) = -\sin(\pi\,x)$, por lo que para evaluar $\sin(\pi\,x)$ se puede obtener como $-\cos\left(\pi\,x + \frac{\pi}{2}\right)$, es decir, `-p(np.mod(np.pi*x+np.pi/2,2*np.pi))`.

- Hint 1: It is important to recall that the cosine function is periodic, so, what can you do if you want to evaluate it for values greater that $2\,\pi$? The same applies for negative values, but they are not needed for this problem.
- Hint 2: You may find useful to consider the modulus operator, which returns the reminder of the operation. For instance in NumPy we can performe the following computation `np.mod(6.2,2.5)` which returns `1.2`.
- Hint 3: The lowest upper bound should contain the coefficient π^n but not $\pi^{2\,n}$ nor other additional coefficients that makes it larger. In this case n corresponds to the number of points used in the interpolation.