Ayudantía
SCT - Semana - 09

Interpolación Polinomial de Funciones: Error de Interpolación y Nodos de Chebyshev

1. Según lo discutido en clases, se demostró que se puede obtener la interpolación polinomial baricéntrica a partir de la interpolación polinomial de Lagrange $p(x) = \sum_{i=1}^{n} y_i L_i(x)$ para los puntos (x_j, y_j). La interpolación baricéntrica corresponde a

$$p(x) = \frac{\sum_{i=1}^{n} y_i \frac{w_i}{x-x_i}}{\sum_{i=1}^{n} \frac{w_j}{x-x_i}},$$

donde $w_i = 1/l_i(x_i)$ y $l_i(x_i) = \prod_{k=1,k\neq i}^{n} (x_i - x_k)$.

a) Demuestre que $\lim_{x\to x_j} p(x) = y_j$, donde y_j es un punto de interpolación.

Se mostrará un ejemplo con solo 2 puntos (x_1, y_1) y (x_2, y_2) para hacerlo más sencillo de entender al principio. La interpolación baricéntrica para estos dos puntos es la recta dada por

$$p(x) = \frac{y_1 \dfrac{w_1}{x-x_1} + y_2 \dfrac{w_2}{x-x_2}}{\dfrac{w_1}{x-x_1} + \dfrac{w_2}{x-x_2}}$$

A medida que x tiende a x_1, la fracción $\frac{w_1}{x-x_1}$, presente tanto en el numerador como en el denominador de $p(x)$, se indefine. Esto causa una *indeterminación* en $p(x)$. Para corregirlo, basta con multiplicar el numerador y el denominador de $p(x)$ por $(x - x_1)$ para deshacerse del denominador problemático de $\frac{w_1}{x-x_1}$:

$$p(x) = \frac{y_1 w_1 + y_2 \dfrac{w_2}{x-x_2}(x-x_1)}{w_1 + \dfrac{w_2}{x-x_2}(x-x_1)}$$

Luego, se puede ver que

$$\lim_{x\to x_1} p(x) = \frac{y_1 w_1}{w_1} = y_1.$$

Un argumento similar para x_2 muestra que $\lim_{x\to x_2} p(x) = y_2$.
En general, si hay n puntos y la interpolación luce así:

$$p(x) = \frac{\displaystyle\sum_{i=1}^{n} y_i \frac{w_i}{x-x_i}}{\displaystyle\sum_{i=1}^{n} \frac{w_j}{x-x_i}},$$

entonces, para calcular su límite cuando $x \to x_j$, primero hay que separar la fracción correspondiente $\frac{w_j}{x-x_j}$ del resto de las fracciones, tanto en el numerador como en el denominador:

$$p(x) = \frac{y_j \dfrac{w_j}{x-x_j} + \displaystyle\sum_{i=1,i\neq j}^{n} y_i \frac{w_i}{x-x_i}}{\dfrac{w_j}{x-x_j} + \displaystyle\sum_{i=1,i\neq j}^{n} \frac{w_j}{x-x_i}}.$$

Esto hace más sencillo el análisis. Luego, hay que multiplicar el numerador y el denominador por $(x - x_j)$:

$$p(x) = \frac{y_j w_j + (x-x_j) \displaystyle\sum_{i=1,i\neq j}^{n} y_i \frac{w_i}{x-x_i}}{w_j + (x-x_j) \displaystyle\sum_{i=1,i\neq j}^{n} \frac{w_j}{x-x_i}}.$$

De ahí, se puede ver que

$$\lim_{x \to x_j} p(x) = \frac{y_j w_j}{w_j} = y_j.$$

b) Como se discutió en clases, construir esta interpolación requiere $\mathcal{O}\left(n^2\right)$ operaciones para calcular los pesos w_i, mientras que evaluarla requiere solo $\mathcal{O}(n)$ operaciones. Sin embargo, si las coordenadas x_i de los puntos que se desea interpolar son los nodos de Chebyshev en un intervalo $[a, b]$, entonces se puede demostrar que w_i se reduce a la siguiente expresión:

$$w_i = -(-1)^i \frac{1}{n} \left(\frac{4}{b-a}\right)^{n-1} \sin\left(\frac{(2i-1)\pi}{2n}\right)$$

Más aún, ya que el factor $k = \frac{1}{n}\left(\frac{4}{b-a}\right)^{n-1}$ es constante y no depende de i, se puede multiplicar el numerador y denominador de $p(x)$ por $\frac{1}{k}$, lo cual equivale a dividir cada peso w_i entre k. Esto implica que se puede usar los siguientes pesos reducidos (válidos solo para la interpolación baricéntrica, no para la de Lagrange clásica):

$$w_i^\star = \frac{w_i}{k} = -(-1)^i \sin\left(\frac{(2i-1)\pi}{2n}\right)$$

Explique claramente cómo esto permite obtener un algoritmo $\mathcal{O}(n)$ tanto para la construcción como la evaluación del polinomio.

En general, construir el polinomio toma $\mathcal{O}(n^2)$ operaciones, debido a que eso es lo que cuesta construir los pesos w_i que se calculan de esta manera:

$$w_i = \frac{1}{\displaystyle\prod_{k=1, k\neq i}^{n} (x_i - x_k)}$$

Según la expresión anterior, cada peso w_i requiere, en general, $\mathcal{O}(n)$ operaciones para ser calculado, debido al cálculo del producto entre $n-1$ términos $(x_i - x_k)$. Al ser n pesos, calcularlos todos requiere $\mathcal{O}(n^2)$ operaciones. Sin embargo, si se nos permite usar los siguientes pesos reducidos:

$$w_i^\star = -(-1)^i \sin\left(\frac{(2i-1)\pi}{2n}\right),$$

entonces cada peso se puede calcular en tiempo constante ($\mathcal{O}(1)$) y los n pesos se calculan, en total, en tiempo $\mathcal{O}(n)$. Al construir los pesos, se puede evaluar el polinomio, como siempre, en tiempo $\mathcal{O}(n)$.

c) Implemente, en Python, con ayuda de NumPy y sin usar ciclos, una función `baricentrica_chebyshev(f, a, b, n)`, donde `f` es una función que se quiere interpolar polinomialmente, `a` y `b` son los límites inferior y superior del intervalo $[a, b]$ en el que se va a interpolar y `n` es la cantidad de puntos de interpolación que se usará. La función `baricentrica_chebyshev` debe construir y retornar otra función `p` que interpola polinomialmente los n puntos (x_i, y_i), donde cada x_i es el correspondiente nodo de Chebyshev en el intervalo $[a, b]$ y cada y_i es el resultado correspondiente de evaluar $f(x_i)$. **Tanto la construcción como la evaluación de `p` deben requerir solo $\mathcal{O}(n)$ operaciones.**

```python
def baricentrica_chebyshev(f, a, b, n):
    # Puntos de Chebyshev en el intervalo [-1, 1]
    i_arr = np.arange(1, n+1)  # [1, 2, 3, ..., n]
    angles = (2*i_arr - 1) * np.pi / 2 / n  # [pi/2n, 3pi/2n, 5pi/2n, ..., (2n-1)pi/2n]
    x_arr = np.cos(angles)

    # Ajuste de puntos a un intervalo [a, b]
    x_arr = (b - a)/2 * x_arr + (a + b)/2

    y_arr = f(x_arr)

    # Pesos reducidos
    w_star_arr = -np.power(-1, i_arr) * np.sin(angles)

    def p(x: float) -> float:
        restas = x - x_arr  # [x-x1, x-x2, x-x3, ..., x-xn]
        fracciones = w_star_arr / restas  # [w1/(x-x1), w2/(x-x2), w3/(x-x3), ...]
        return np.dot(fracciones, y_arr) / np.sum(fracciones)
```

```python

    return p
```

d) Dada la pregunta anterior, construya un polinomio interpolador con 3 puntos de Chebyshev en el intervalo $[1, 7]$ de la función $\sin^3(x)$ y obtenga el error al interpolar en $x = 2$.

```python
>>> def sin_cubed(x):
...     sin = np.sin(x)
...     return sin * sin * sin
...
>>> p = baricentrica_chebyshev(sin_cubed, 1, 7, 3)
>>> err = abs(p(2) - sin_cubed(2))
>>> err
0.2785157206748617
```

2. Considere la siguiente función en 2 variables $f : \mathbb{R}^2 \to \mathbb{R}$:

$$f(x, y) = \frac{4 - y^2}{x}$$

a) Determine el grado mínimo del polinomio que interpole la función $f(x, y)$ a lo largo de la curva paramétrica $\langle x(s), y(s)\rangle = \langle \cos(s), 2\sin(s)\rangle$ para $s \in [2, 5]$, tal que el error de interpolación sea menor que 10^{-5}. *Hint: The function along the parametric function is defined as $f(x(s), y(s))$ and it only depends on one variable!*

Al sustituir $x = \cos(s)$ e $y = 2\sin(s)$ en la función $f(x, y) = \frac{4-y^2}{x}$, se obtiene una función de una sola variable $g(s) = f(x(s), y(s))$:

$$\begin{aligned} g(s) = f(x(s), y(s)) &= \frac{4 - 4\sin^2(s)}{\cos(s)} \\ &= \frac{4(1 - \sin^2(s))}{\cos(s)} \\ &= \frac{4\cos^2(s)}{\cos(s)} \\ &= 4\cos(s) \end{aligned}$$

En general, si se usa un polinomio p para interpolar g en $[2, 5]$, el error máximo de interpolación es

$$\begin{aligned} E_{\text{máx}} &= \max_{s\in[2,5]} |g(s) - p(s)| \\ &= \max_{s\in[2,5]} \left(\frac{|(s - s_1)(s - s_2)\cdots(s - s_n)|}{n!} \left|g^{(n)}(c)\right| \right) \\ &\leq \frac{\max_{s\in[2,5]} |(s - s_1)(s - s_2)\cdots(s - s_n)|}{n!} \max_{s\in[2,5]} \left|g^{(n)}(s)\right| \end{aligned}$$

Para minimizar este error máximo, se puede usar las n raíces $s_1, s_2, ..., s_n$ del n-ésimo polinomio de Chebyshev ajustado al intervalo $[2, 5]$. En tal caso:

$$\begin{aligned} \max_{s\in[2,5]} |(s - s_1)(s - s_2)\cdots(s - s_n)| &= \frac{1}{2^{n-1}} \left(\frac{5-2}{2}\right)^n \\ &= \frac{2}{2^n} \left(\frac{3}{2}\right)^n \\ &= 2\left(\frac{3}{4}\right)^n \end{aligned}$$

Por otro lado, se debe encontrar $\max_{s\in[2,5]} \left|g^{(n)}(s)\right|$. Al ser $g(s) = 4\cos(s)$, se cumple que

$$\left|g^{(n)}(s)\right| = \begin{cases} 4|\cos(s)|, \text{ si } n \text{ es par} \\ 4|\sin(s)|, \text{ si } n \text{ es impar} \end{cases}$$

y, debido a que el intervalo $[2,5]$ contiene los siguientes múltiplos de $\frac{\pi}{2}$:

- el punto $s=\pi$ donde $4|\cos(s)|$ alcanza su valor máximo 4
- el punto $s=\frac{3\pi}{2}$ donde $4|\sin(s)|$ alcanza también su valor máximo 4

podemos concluir que

$$\max_{s\in[2,5]} \left|g^{(n)}(s)\right| = 4,\ \forall n\in\mathbb{N}$$

Reemplazando ambos valores en la expresión para el error máximo de interpolación:

$$\begin{aligned} E_{\text{máx}} &\leq \frac{\max_{s\in[2,5]}|(s-s_1)(s-s_2)\cdots(s-s_n)|}{n!}\max_{s\in[2,5]}\left|g^{(n)}(s)\right| \\ &= \frac{2\left(\frac{3}{4}\right)^n}{n!}\cdot 4 \\ &= \frac{8}{n!}\left(\frac{3}{4}\right)^n \end{aligned}$$

Se requiere que esta cota al error máximo de interpolación, $\frac{8}{n!}\left(\frac{3}{4}\right)^n$, sea menor a 10^{-5}:

```python
>>> n = 0
>>> error = 8
>>> while error > 1e-5:
...     n += 1
...     error *= 0.75 / n # error = 8/n! * 0.75**n
...
>>> print(n)
9
>>> print(error)
1.6553061349051336e-06
>>> print(8/(1*2*3*4*5*6*7*8*9) * (0.75 ** 9))
1.655306134905134e-06
```

Por lo tanto, el grado mínimo del polinomio interpolador requerido es $n=9$, en cuyo caso el error máximo de interpolación es $E_{\text{máx}} \leq 1.655\times 10^{-6}$.

b) Construya el polinomio interpolador de grado mínimo determinado en la pregunta anterior.

Los 9 puntos de Chebyshev requeridos $s_1, s_2, ..., s_9$, ajustados al intervalo $[2,5]$, están dados por:

$$s_i = 3.5 + 1.5\cos\left(\frac{(2i-1)\pi}{18}\right),\ i\in\{1,2,...,9\}$$

y cada valor $z_i = g(s_i)$ (usamos z y no y, porque la función original era $z=f(x,y)$) está dado por

$$z_i = g(s_i) = 4\cos\left(3.5+1.5\cos\left(\frac{(2i-1)\pi}{18}\right)\right),\ i\in\{1,2,...,9\}$$

Una opción para construir el polinomio interpolador es mediante la interpolación de Lagrange:

$$p(s) = \sum_{i=1}^{9} z_i L_i(s), \quad \text{donde } L_i(s) = \frac{\prod_{j=1,j\neq i}^{9}(s-s_j)}{\prod_{j=1,j\neq i}^{9}(s_i-s_j)}.$$

Otra opción, más eficiente en número de operaciones elementales, es la interpolación baricéntrica:

$$p(s) = \frac{\sum_{i=1}^{9} z_i \frac{w_i}{s-s_i}}{\sum_{i=1}^{9}\frac{w_i}{s-s_i}}, \quad \text{donde cada peso } w_i = \frac{1}{\prod_{j=1,j\neq i}^{9}(s_i-s_j)}.$$