– Práctica 4 –

Interpolación polinómica de Lagrange

1. **Cálculo de las diferencias divididas.**
   Como se ha visto en clase, la forma más eficiente de calcular el polinomio de interpolación de Lagrange asociado a los datos

   $$\{(x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

   es haciendo uso de las diferencias divididas (forma de Newton). Recordemos cómo se definen.

   - Diferencias divididas de orden 0:

     $$f[x_i] = y_i, \quad i = 0, \ldots, n.$$

   - Diferencias divididas de orden 1:

     $$f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}, \quad i = 0, \ldots, n-1.$$

   - Diferencias divididas de orden k:

     $$f[x_i, x_{i+1}, \ldots, x_{i+k}] = \frac{f[x_{i+1}, x_{i+2}, \ldots, x_{i+k}] - f[x_i, x_{i+1}, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}, \quad i = 0, \ldots, n-k.$$

   La función de Python `tabla_diferencias_divididas` (véase fichero en Campus Virtual) calcula la tabla de diferencias divididas a partir de los datos

   $$x = [x_0, x_1, x_2, \ldots, x_n], \quad y = [y_0, y_1, y_2, \ldots, y_n].$$

   Esta función devuelve una matriz triangular inferior que en la columna k-ésima contiene las diferencias divididas de orden k.

   a) Usar la función anterior para calcular la tabla de diferencias divididas asociada a los datos

   $$x = [0, 0.25, 0.5, 0.75, 1.0], \quad y = \exp([0, 0.25, 0.5, 0.75, 1.0]).$$

2. **Cálculo del polinomio de interpolación de Lagrange mediante la fórmula de Newton.**
   Una vez calculadas las diferencias divididas, el polinomio de interpolación de Lagrange asociado a los datos

   $$\{(x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

   se escribe:

   $$p(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \ldots + f[x_0, x_1, x_2, \ldots, x_n](x - x_0)(x - x_1)\ldots(x - x_{n-1})$$

   La función de Python `eval_forma_newton` (véase Campus Virtual) evalúa el polinomio de interpolación en un punto $z_0 \in \mathbb{R}$. También funciona si z_0 es un vector de puntos, devolviendo un vector de la misma dimensión que z_0 que, en cada componente, contiene el valor del polinomio en la componente correspondiente de z_0.

a) Usar la función anterior para evaluar el polinomio que interpola los datos

$$x = [0, 0.25, 0.5, 0.75, 1.0], \quad y = \exp([0, 0.25, 0.5, 0.75, 1.0]).$$

en $x = 1/3$ y verificar que, efectivamente, el polinomio de interpolación pasa por los datos utilizados en su definición.

b) Implementar un programa de Python que evalúe un polinomio

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

en un punto z_0 usando el algoritmo de Horner:

$$p(z_0) = (\ ((a_n z_0 + a_{n-1}) z_0 + a_{n-2}) z_0 + \ldots + a_1) z_0 + a_0.$$

Recordemos que el algoritmo de Horner optimiza el número de operaciones necesarias para realizar dicha evaluación.

c) A fin de optimizar también el número de operaciones que se realizan al evaluar el polinomio de interpolación de Lagrange usando la forma de Newton, podemos considerar una variante del algoritmo de Horner. En efecto, si $a_k = f[x_0, x_1, \ldots, x_k]$, entonces

$$p(z_0) = a_0 + a_1(z_0 - x_0) + \ldots + a_n(z_0 - x_0) \cdot \ldots \cdot (z_0 - x_{n-1})$$

puede escribirse como

$$p(z_0) = (\ ((a_n(z_0 - x_{n-1}) + a_{n-1})(z_0 - x_{n-2}) + a_{n-2})(z_0 - x_{n-3}) + \ldots + a_1)(z_0 - x_0) + a_0.$$

Implementar una función Python para evaluar el polinomio de interpolación usando el algoritmo de Horner modificado.

d) Programar una función Python que, dada una función $f : [a, b] \mapsto \mathbb{R}$, calcule el polinomio de interpolación $p(x)$ asociado a los datos

$$\{(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_N))\},$$

siendo $\{x_0, \ldots, x_N\}$ una partición equidistante del intervalo $[a, b]$, es decir:

$$x_k = a + k\,h, \quad k = 0, \ldots, N, \tag{1}$$

donde $h = (b - a)/N$. Utilizar para ello las funciones utilizadas en los ejercicios anteriores. La función tendrá como parámetros de entrada la función f, los extremos del intervalo, el número de intervalos de la partición N y el vector en el que se desea evaluar el polinomio de interpolación ($z_0 = [z_{0,1}, \ldots, z_{0,L}]$). Los parámetros de salida serán el vector y de valores del polinomio p evaluado en z_0 y el máximo error cometido

$$error = \text{máx}_{i=1,\cdots,L} |f(z_{0,i}) - p(z_{0,i})|.$$

e) Aplicar la función anterior a $f(x) = e^x$ en el intervalo $I = [-3, 3]$, con $N = 5, 10, 15, 20$, siendo z_0 el vector que resulta al dividir el intervalo en porciones de longitud 0.01. Comparar en todos los casos las gráficas de f y p y analizar el comportamiento de los errores de interpolación al ir aumentando N.

f) Repetir el apartado anterior anterior para $f(x) = 1/(1 + x^2)$ en el intervalo $I = [-5, 5]$.

g) Repetir el apartado anterior para los nodos de Chebyshev, que en el intervalo $I = [-5, 5]$ están dados por:

$$x_k = 5 \cos\left(\frac{(2k+1)\pi}{2(N+1)}\right), \qquad k = 0, 1, \ldots, N.$$

3. **Interpolación de tipo spline.**
El objetivo de este ejercicio es programar una función Python que, dada una función $f\colon [a,b] \mapsto \mathbb{R}$, calcule la función lineal a trozos p que interpola los datos

$$\{(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_N))\},$$

con nodos x_k definidos en (1). Asímismo se escribirá otra función Python que calcule un interpolante *spline* cúbico para los mismos datos. Para ello, usaremos la función de Python `interp1d`. Para poder hacerlo, debemos importarla mediante

```
from scipy.interpolate import interp1d
```

y usar el comando

```
pol=interp1d(xd,yd,kind='linear')
```

para definir la función p lineal a trozos, siendo xd e yd el conjunto de datos, o bien el comando

```
pol=interp1d(xd,yd,kind='cubic')
```

para definir la función *spline* cúbica correspondiente. Para realizar la evaluación en un x dado basta con ejecutar

```
pol(x) .
```

a) Programar ambas funciones usando como parámetros de entrada la función f, los extremos del intervalo $[a,b]$ y el número de intervalos de la partición N. Como parámetro de salida se asignará la *handle* del interpolante generado por la función `interpd1`.

b) Aplicar ambas funciones a los ejemplos considerados en el ejercicio anterior.