PRÁCTICA 9
INTERPOLACIÓN DE LAGRANGE EN LA FORMA DE NEWTON

Algoritmo 1. Polinomio de interpolación (Newton).
Dados unos nodos $x_0, \ldots, x_N$ distintos dos a dos y unos valores $f[x_0], \ldots, f[x_N]$, sabemos que existe un único polinomio p de grado $\leq N$ tal que $p(x_i) = f[x_i]$ para $i = 0, \ldots, N$ (Teorema 1). Este polinomio se puede expresar en la forma de Newton:

$$p(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \ldots + c_N(x - x_0)(x - x_1)\cdots(x - x_{N-1}),$$

donde los coeficientes $c_i = f[x_0, \ldots, x_i]$ se calculan recursivamente:

$$\begin{array}{cccccccccc}
x_0 & f[x_0] & & & & & & & & \\
 & & \searrow & & & & & & & \\
x_1 & f[x_1] & \rightarrow & f[x_0, x_1] & & & & & & \\
 & & \searrow & & \searrow & & & & & \\
x_2 & f[x_2] & \rightarrow & f[x_1, x_2] & \rightarrow & f[x_0, x_1, x_2] & & & & \\
 & & \searrow & & \searrow & & \searrow & & & \\
\ldots & \ldots & & \ldots & & \ldots & & & & \\
 & & & & & & & & & \\
\ldots & \ldots & & \ldots & & \ldots & & \ldots & & \\
 & & \searrow & & \searrow & & \searrow & & \searrow & \\
x_N & f[x_N] & \rightarrow & f[x_{N-1}, x_N] & \rightarrow & f[x_{N-2}, x_{N-1}, x_N] & \rightarrow & \cdots & \rightarrow & f[x_0, \ldots, x_N] \\
 & & & & & & & & & \\
 & (0) & & (1) & & (2) & & \cdots & & (N)
\end{array}$$

donde las flechas indican cómo obtener la diferencia dividida (Teorema 2):

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

Paso (0): Inicializamos el vector de coeficientes $c = (c_i)$ como $c_i \leftarrow f[x_i]$ para $i = 0, \ldots, N$.
paso (k): Calculamos $c_i \leftarrow f[x_{i-k}, \ldots, x_i]$ para $i = k, \ldots, N$. Es conveniente hacerlo de abajo a arriba (comenzando por las flechas rojas) ¿Por qué?

Algoritmo 2. Evaluación del polinomio (Horner).
Una vez conocidos los coeficientes $c_0, \ldots, c_N$, la manera más eficiente de evaluar p en un nuevo punto x es mediante el método de Horner. Aprovechando los factores comunes $(x - x_k)$ de la forma de Newton podemos escribir p como sigue:

$$p(x) = c_0 + (x - x_0)\underbrace{\left(c_1 + (x - x_1)\underbrace{\left(c_2 + \ldots + (x - x_{N-2})\underbrace{\left(c_{N-1} + (x - x_{N-1})c_N\right)}_{q_{N-1}}\ldots\right)}_{q_2}\right)}_{q_1}$$

Inicializando el polinomio (constante) $q_N = c_N$, calculamos el polinomio q_k recursivamente:

$$q_k(x) = c_k + (x - x_k)q_{k+1}(x), \qquad k = N - 1, \ldots, 0,$$

obteniendo finalmente $p(x) = q_0(x)$.

Ejercicio 1. Escribe una función diferencias.m en Matlab con:

$$\textit{(inputs)} \qquad \text{nodos} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_N \end{bmatrix}, \qquad \text{valores} = \begin{bmatrix} f[x_0] \\ f[x_1] \\ \vdots \\ f[x_N] \end{bmatrix},$$

$$\textit{(output)} \qquad c = \begin{bmatrix} f[x_0] \\ f[x_0, x_1] \\ \vdots \\ f[x_0, \dots, x_N] \end{bmatrix},$$

mediante el algoritmo 1.

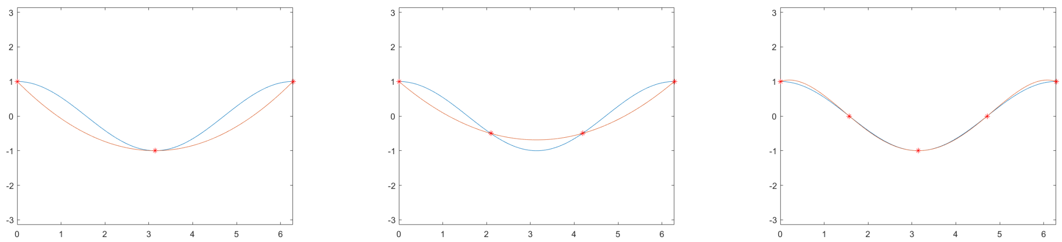
Ejercicio 2. Escribe una función newton.m en Matlab con:

$$\textit{(inputs)} \qquad \text{nodos}, \qquad \text{valores}, \qquad x,$$
$$\textit{(output)} \qquad y = p(x).$$

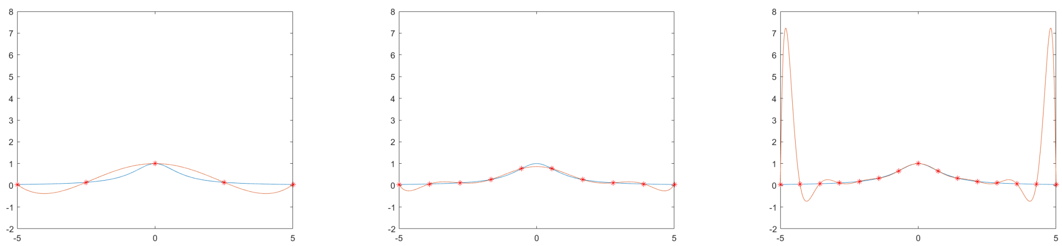
mediante el algoritmo 2 (llamando a diferencias para obtener c).

Ejercicio 3. Escribe un programa inter_cos.m en Matlab que realice la siguiente tarea:

- Pide al usuario un número natural n.
- Calcula el polinomio de interpolación p (diferencias) de la función $f(x) = \cos(x)$ en n nodos equiespaciados[1] en el intervalo $[0, 2\pi]$.
- Saca por pantalla la gráfica de la función f y del polinomio p (newton) así como los puntos de interpolación:



Ejercicio 4. Escribe un programa fenomeno_Runge.m en Matlab que realice la tarea del ejercicio 3 para la función $f(x) = \frac{1}{1+x^2}$ en n nodos equiespaciados en el intervalo $[-5, 5]$. Observa el fenómeno de Runge en los extremos del intervalo.



[1] linspace(a,b,n) devuelve un vector de n puntos equiespaciados en el intervalo $[a, b]$.