Programación: Multiplicación de un polinomio por un binomio

Objetivos. Escribir una función que multiplique un polinomio por un binomio.

Requisitos. Ciclos y otros elementos de programación.

1. Arreglos en Python3 + NumPy. Hay que tener en cuenta que

en Python los índices de elementos de arreglos empiezan en 0.

Para acostumbrarse a la sintaxis ejecute los siguientes comandos uno por uno:

```
a = array([10, 20, 30], dtype = double)
```
(∗ *crear un arreglo con entradas 10, 20, 30 guardadas como números reales* ∗)

```
len(a)
```
(∗ *longitud del arreglo* ∗)

```
a[0]
```
(∗ *obtener el valor del elemento con índice* 0 ∗)

```
a[1]
```
(∗ *obtener el valor del elemento con índice* 1 ∗)

```
a[1] = 70
```
(∗ *modificar el valor del elemento con índice* 1 ∗)

```
a
```
(∗ *ver el arreglo modificado* ∗)

```
b = zeros((7,), dtype = double)
```
(∗ *crear un arreglo de 7 ceros* ∗)

2. Guardar polinomios como arreglos de sus coeficientes. Vamos a representar los polinomios como arreglos de sus coeficientes, empezando con el término independiente. Por ejemplo, el polinomio

$$f(x) = 7x^4 - 3x^2 + 5x + 4$$

se guardará como un arreglo de números $4, 5, -3, 0, 7$. En general, el polinomio

$$f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}$$

guardamos como el arreglo con entradas $a_0, a_1, a_2, \ldots, a_{n-1}$. Aquí n es el número de coeficientes, o sea la *longitud* del arreglo de coeficientes. En vez de n se puede usar la variable $d = n - 1$, es decir, el *grado* del polinomio. Nótese que para todo índice k de 0 a $n - 1$, el coeficiente de x^k es a_k.

Fórmulas de multiplicación de un polinomio por un binomio (se recomienda deducirlas antes de la clase práctica)

3. Fórmulas para multilicar un polinomio de grado 3 por un binomio mónico. Sean $f(x)$ un polinomio de grado 3 y $g(x)$ un binomio mónico, es decir, un binomio cuyo coeficiente del grado mayor es 1:

$$f(x) = a_0 + a_1x + a_2x^2 + a_3x^3, \qquad g(x) = b + x.$$

Notemos que el polinomio $f(x)$ es de grado 3 y por lo tanto tiene $\underbrace{\qquad}_{\text{¿cuántos?}}$ coeficientes.

En este caso el producto $f(x)g(x)$ es de grado $\underbrace{\qquad}_{?}$, esto es, tiene $\underbrace{\qquad}_{\text{¿cuántos?}}$ coeficientes.

Denotemos los coeficientes del producto $f(x)g(x)$ por $c_0, c_1, \ldots, \underbrace{\qquad}_{?}$:

$$c_0 + c_1x + c_2x^2 + c_3x^3 + c_4x^4 = (a_0 + a_1x + a_2x^2 + a_3x^3)(b + x).$$

Exprese los coeficientes $c_0, \ldots, c_4$ a través de $a_0, \ldots, a_3$ y b:

$$c_0 = \underbrace{\qquad\qquad}_{?},$$

$$c_1 = \underbrace{\qquad\qquad}_{?},$$

$$c_2 = \underbrace{\qquad\qquad}_{?},$$

$$c_3 = \underbrace{\qquad\qquad}_{?},$$

$$c_4 = \underbrace{\qquad\qquad}_{?}.$$

Se puede ver que las fórmulas para c_1, c_2, c_3 tienen la misma estructura:

$$c_k = \underbrace{\qquad\qquad}_{?}, \qquad \text{para} \qquad \underbrace{\qquad}_{?} \le k < \underbrace{\qquad}_{?}.$$

Las fórmulas para los "coeficientes extremos" c_0 y c_4 son diferentes ("degeneradas").

4. Fórmulas para multiplicar un polinomio por un binomio mónico.
Sean

$$f(x) = a_0 + a_1x + \ldots + a_{n-1}x^{n-1}, \qquad g(x) = b + x.$$

Entonces el producto $f(x)g(x)$ es de grado $\underbrace{\qquad}_{?}$, esto es, tiene $\underbrace{\qquad\qquad}_{\text{¿cuantos?}}$ coeficientes.

Denotemos por $c_0, \ldots, c_n$ los coeficientes del producto $f(x)g(x)$:

$$c_0 + c_1x + \cdots + c_nx^n = (a_0 + a_1x + \ldots + a_{n-1}x^{n-1})(b + x).$$

Exprese los coeficientes $c_0, \ldots, c_n$ a través de los coeficientes $a_0, \ldots, a_{n-1}$ y b.

$c_0 = \underbrace{\qquad\qquad}_{?}$;

$c_k = \underbrace{\qquad\qquad\qquad\qquad}_{?}$ para $\underbrace{\qquad}_{?} \le k < \underbrace{\qquad}_{?}$;

$c_n = \underbrace{\qquad\qquad}_{?}$.

5. Algoritmo MulPolBinom (pseudocódigo).

función MulPolBinom(a, b):

variables locales: n, c, k;

$n \coloneqq$ longitud(a);

$c \coloneqq$ arreglo nulo de longitud $\underbrace{\qquad\qquad}_{?}$;

$c_0 \coloneqq \underbrace{\qquad\qquad}_{?}$;

para cada k en el intervalo entero $\big[\underbrace{\qquad}_{?}, \underbrace{\qquad}_{?}\big)$:

$c_k \coloneqq \underbrace{\qquad\qquad\qquad\qquad}_{?}$;

$c_n \coloneqq \underbrace{\qquad\qquad}_{?}$;

regresar $\underbrace{\qquad}_{?}$.

Programar la multiplicación de un polinomio por un binomio en algún lenguaje de programación

6. Problema MulPolBinom.
Traduzca el algoritmo MulPolBinom a un lenguaje de programación. En otras palabras, escriba una función que calcule los coeficientes del producto de un polinomio $f(x)$ por un binomio $b + x$.

- Entrada (argumentos de la función): a, b,
  donde a es el arreglo de los coeficientes de $f(x)$.
- Salida (que debe regresar la función): el arreglo de los coeficientes del producto.

Por ejemplo, en Python3 + NumPy la función MulPolBinom se puede definir de la siguiente manera (hay que sustituir `...` por expresiones apropiadas):

Programa 1: MulPolBinom

```
from numpy import *

def MulPolBinom(a, b):
    n = len(a)
    c = ...
    c[0]  = ...
    ...

    ...
    return c

print(MulPolBinom(array([7, -2, 4, -5]), 3))  # test 1
print(MulPolBinom(array([3, -7, 5]), 3))  # test 2
```

7. Primera comprobación. Es fácil ver que

$$(7 - 2x + 4x^2 - 5x^3)(3 + x) = 21 + x + 10x^2 - 11x^3 - 5x^4,$$

Por eso el comando

```
print(MulPolBinom[array([7, -2, 4, -5]), 3])
```

debe imprimir en la pantalla el arreglo `21, 1, 10, -11, -5`.

8. Segunda comprobación. Multiplique a mano el polinomio $5x^2-7x+3$ por el binomio $x + 2$. Luego calcule `MulPolBinom[array([3, -7, 5]), 2]`.