Programación: Construcción del polinomio mónico de grado mínimo con raíces dadas

Objetivos. Escribir una función que calcule los coeficientes del polinomio mónico de grado mínimo que tengas las raíces dadas.

Requisitos. Multiplicación de un polinomio por un binomio mónico, evaluación de un polinomio en un arreglo de puntos, teorema del resto y sus corolarios.

Revisamos los requisitos

1. Representación de polinomios como arreglos de sus coeficientes (repaso). Por ejemplo, representamos el polinomio $3x^4 - 7x^3 - 2x + 6$ como `[6; -2; 0; -7; 3]`.

2. Multiplicación de un polinomio por un binomio mónico (repaso). En clases pasadas programamos una función `mulpolbinom` que multiplica un polinomio (dado por el arreglo de sus coeficientes) por un binomio de la forma $x + b$ dado por el número `b`. Por ejemplo,

`mulpolbinom([-2; 0; 6; 1; 5], 3)` regresa `[-6; -2; 18; 9; 16; 5]`.

3. Evaluación de un polinomio en un arreglo de puntos (repaso). En clases pasadas programamos un función `poleval` que evalúa un polinomio (dado por el arreglo de sus coeficientes) en un arreglo de puntos y regresa el arreglo de los valores correspondientes. Por ejemplo,

`poleval([-5; 3; 4; -2], [3; -2; 4])` regresa `[-14; 21; -57]`.

Polinomio con raíces dadas: idea

4. Dados algunos puntos $a_1, a_2, a_3, a_4 \in \mathbb{R}$, el polinomio

$$f(x) = \prod_{j=1}^{4}(x - a_j) = (x - a_1)(x - a_2)(x - a_3)(x - a_4)$$

tiene raíces a_1, a_2, a_3, a_4 y es mónico. Si algunos de los puntos a_1, a_2, a_3, a_4 coinciden, entonces son raíces múltiples de f. Por ejemplo, si $a_1 = a_2$, pero $a_1 \neq a_3$ y $a_1 \neq a_4$, entonces

$$(x - a_1)^2 \mid f(x), \qquad \text{pero} \qquad (x - a_1)^3 \nmid f(x).$$

Además f es el polinomio de grado mínimo con estas propiedades.

5. Fórmula general. Sean $a_1, \ldots, a_n$ algunos números. Entonces el polinomio mónico de grado mínimo con raíces $a_1, \ldots, a_n$ es

$$f(x) = \prod_{j=???}^{???} (??? - ???). \tag{1}$$

6. Definición natural del producto vacío. Los productos $\prod$ se pueden definir mediante la fórmula recursiva

$$\prod_{k=1}^{n+1} c_k = \left(\prod_{k=1}^{???} ??? \right) c_{???} \tag{2}$$

Queremos que esta fórmula sea correcta no solamente para $n = 1, 2, 3, \ldots$, sino también en el caso $n = 0$. Calculamos el lado izquierdo y el lado derecho en este caso:

$$\text{L.I.} = \qquad\qquad , \quad \text{L.D.} = \qquad\qquad .$$

Para garantizar la igualdad L.I. = L.D. (para cada c_1), ponemos

$$\prod_{k=1}^{0} c_k = \underbrace{\qquad}_{?}. \tag{3}$$

7. El caso de 0 puntos.

- ¿Qué polinomio está determinado por el lado derecho de (1) en el caso $n = 0$?
- ¿Cuál es el polinomio mónico de grado mínimo que no tiene raíces?

Estas dos preguntas tienen la misma respuesta, la cual nos ayuda iniciar bien el proceso iterativo (el ciclo `for`) que realiza la fórmula (1).

8. Fórmulas iterativas. Sean a_1, a_2, a_3, a_4 algunos números dados. Calculamos por pasos el polinomio $(x - a_1)(x - a_2)(x - a_3)(x - a_4)$.

- Empezamos con el polinomio neutro multiplicativo: $f_0(x) := 1x^0$.
- Calculamos $f_1(x)$ como el producto del polinomio $f_0(x)$ por el binomio $(x - a_1)$.
- $f_2(x) := f_1(x)(x - a_2)$.
- $f_3(x) :=$.
- $f_4(x) :=$.

9. Cálculo del polinomio con raíces dadas, ejemplo. Constuir el polinomio mónico de grado mínimo con raíces

$$-2, \quad 3, \quad 3, \quad 5.$$

Solución. Hay que calcular el producto de binomios

$$f(x) = 1 \cdot (x+2) \cdot (x-3) \cdot (x-3) \cdot (x-5).$$

Usamos el algoritmo de multiplicación de un polinomio por un binomio:

	1		
2	2	1	
-3			
-5	-90		1

Respuesta:

$$f(x) = -90 + 33x + 17x^2 - 9x^3 + x^4.$$

10. El polinomio constante 1. El polinomio $1x^0$ representamos como el arreglo [1]. En algunos lenguajes de programación es lo mismo que el número 1, en otros lenguajes 1 y [1] son objetos diferentes.

11. Programar una función que calcule los coeficientes del polinomio mónico de grado mínimo que tenga raíces dadas. Escribir una función `polfromroots` de un argumento vectorial `a` que calcule los coeficientes del polinomio

$$\prod_{j=1}^{n}(x - a_j),$$

donde denotamos la longitud de `a` por `n`. Usamos un ciclo `for` y la función `mulpolbinom`:

```
function [c] = polfromroots(a),
   n = length(a);
   c = ???;
   for j = 1 : n,
      ??? = mulpolbinom(???, ???);
   end
end
```

12. Complejidad del algoritmo polfromroots. Calcular el número de operaciones de multiplicación que se realizan en el algoritmo anterior, si el arreglo `a` tiene longitud n. Recordamos que al ejecutar `mulpolbinom(p, q)`, donde `p` es de longitud m, se realizan m operaciones de multiplicación. Respuesta:

$$\sum_{j=1}^{n} \underbrace{\qquad}_{?} = \qquad .$$

13. Una prueba pequeña de la función polfromroots. Hacemos una prueba de la función `polfromroots` en el intérprete de GNU Octave, usando los datos del Ejemplo 9:

```
f = polfromroots([-2; 3; 3; 5])
```

La respuesta debe coincidir con la respuesta que calculamos a mano en el Ejemplo 9. También verificamos que el polinomo `f` se anula en los puntos $-2, 3, 3, 5$:

```
poleval(f, [-2; 3; 3; 5])
```

14. Pruebas pseudoaleatorias de la función polfromroots. Guardamos la siguiente función en un archivo con título `testpolfromroots.m`:

```
function [] = testpolfromroots(n, nrep),
   rootsbase = randn(n, nrep);
   maxerror = 0;
   t1 = cputime();
   for rep = 1 : nrep,
      a = rootsbase(:, rep);
      f = polfromroots(a);
      v = poleval(f, a);
      error = norm(v);
      maxerror = max(error, maxerror);
   end
   t2 = cputime();
   disp(t2 - t1);
   disp(maxerror);
end
```

En el intérprete de GNU Octave ejecutamos

```
testpolfromroots();
```