Programación: Interpolación segmentaria cúbica

Objetivos. Programar las funciones que realicen la interpolación segmentaria cúbica.

Requisitos. Vamos a usar algunas funciones escritas en clases pasadas:

> `FindSegment` de dos argumentos `xs`, `x` determina a cuál intervalo de la partición `xs` pertenece el punto `x`.
>
> `EvalPol` de dos argumentos `fcoefs`, `x` calcule el valor en el punto `x` del polinomio dado por su lista de coeficientes `fcoefs`.
>
> `SolveTriDiag` resuelve el sistema tridiagonal de ecuaciones lineales. Los argumentos son las diagonales `a`, `b`, `c` y el vector del lado derecho `d`.

Notación para listas. Denotemos por `xs` a la variable que guarda la lista de los puntos $x_1, \ldots, x_n$, así que el punto x_1 se va a denotar por `xs[[1]]` en Wolfram Mathematica. De manera similar usemos las variables `ys`, `as`, `bs`, `cs`, `ds` para denotar las listas correspondientes.

1. Cálculo de los valores de una función cúbica a trozos (2 %). Escriba una función `CubicSpline` de argumentos `xs`, `as`, `bs`, `cs`, `ds`, `x` que calcule la función cúbica a trozos definida por la regla:

$$f(x) = a_k + b_k(x - x_k) + c_k(x - x_k)^2 + d_k(x - x_k)^3, \qquad \text{si} \qquad x_k \le x < x_{k+1}.$$

Para el punto $x = x_n$ la función tiene que aplicar la misma regla que en el último intervalo $[x_{n-1}, x_n)$, esto es, tiene que regresar

$$a_{n-1} + b_{n-1}(x_n - x_{n-1}) + c_{n-1}(x_n - x_{n-1})^2 + d_{n-1}(x_n - x_{n-1})^3.$$

Para determinar a cuál intervalo pertenece el punto `x` llame la función `FindSegment` escrita en clases pasadas. Para calcular el valor del polinomio cúbico puede llamar la función `EvalPol`.

2. Coeficientes de la interpolación segmentaria cúbica (3 %). Escriba una función `CoefsCubicSpline` de dos argumentos `xs`, `ys` que regrese las listas `as`, `bs`, `cs`, `ds` de los coeficientes de la interpolación segmentaria cúbica. Dichos coeficientes luego se usarán como argumentos de la función `CubicSpline`. Utilice las fórmulas deducidas en clases o escritas en libros.

3. Prueba total.

```
xs = {0,1.5,3,4,5,7}; ys = Map[Sin, xs];

{as, bs, cs, ds} = CoefsCubicSpline[xs, ys];

Plot[{Sin[x], CubicSpline[xs, as, bs, cs, ds, x]}, {x, 0, 7}]
```