Ampliación de interpolación con Splines

Miguel Anguita Ruiz Pablo Baeyens Fernández Pablo David Medina Sánchez
Ruben Morales Pérez Francisco Javier Morales Piqueras

Splines cuadráticos

El espacio de splines de clase 2 se nota $S_2(x_1, x_2, ..., x_n)$. Una base sería $\{1, x, x^2, (x - x_1)_+, (x - x_2)_+, ..., (x - x_n)_+\}$

Descripción del espacio de splines cuadráticos

El espacio de splines de clase 2 con n nodos se denota $S_2(x_1, x_2, ..., x_n)$. Los splines de clase 2 están constituidos por parábolas, de forma que además de tener una función continua, su derivada también lo es. Por lo tanto, para $i = 1, ..., n - 1$ tenemos la siguiente condición:

$$\boxed{\begin{array}{l} s_i(x_i) = s_{i+1}(x_{i+1}) \\ s'_i(x_i) = s'_{i+1}(x_{i+1}) \end{array}}$$

Proposición: El conjunto $S_2(x_1, x_2, ..., x_n)$ satisface las propiedades siguienes:

1. Es un espacio vectorial con $\dim(S_2(x_1, x_2, ..., x_n))$

Ejemplos

Splines cúbicos

Construcción a partir de los valores de $s''(x)$ en los nodos $\{x_i\}$

Propiedades de minimización

Ejemplos

Implementación en ordenador: Octave

Implementaremos las siguientes funciones en Octave:

1. `Spline31` : Calcula spline de **clase 1**.
2. `SplineNat`: Calcula spline **natural**.
3. `SplinePer`: Calcula spline **periódico**.

4. `SplineSuj`: Calcula spline **sujeto**.
5. `ConvierteApp`: Transforma los coeficientes de un spline en una estructura `pp`.
6. `SplineCuad`: Calcula spline **cuadrático** de clase 1.

Splines cúbicos

Spline natural

Para crear la función `SplineNat`, creamos primero una función que calcule las derivadas definiendo un sistema de ecuaciones:

```
function d = DersSplineNat(x, y)
  h     = diff(x);
  h2_n  = h(2:end);    # h del segundo elemento en adelante
  h1_n1 = h(1:end-1); # h sin el último elemento
  p     = diff(y)./h;

  b = 3.*[p(1) (h2_n.*p(1:end-1) + h1_n1.*p(2:end)) p(end)];

  A  = diag([h2_n 1], -1);
  A += diag(2*[1 (h2_n + h1_n1) 1]);
  A += diag([1 h1_n1], 1);

  d = A\b'; # Derivadas
end
```

Las funciones que utilizamos son:

- `diff(x)` es un vector con elemento `i` `x(i) - x(i-1)`.
- `end` en un rango indica el último elemento de un vector.
- `diag(d, n)` define una matriz $n-$diagonal con diagonal `d`.

Obtenidas las derivadas basta calcular cada polinomio s_i con la función `Spline31` definida para los splines cúbicos de clase 1:

```
SplineNat = @(x,y) Spline31(x, y, DersSplineNat(x,y))
```

Spline sujeto

Spline periódico

Splines cuadráticos

Resolviendo a trozos

Mediante resolución de sistema

Utilizando el sistema que vimos anteriormente, podemos definir fácilmente una función que calcule los coeficientes de un spline cuadrático de clase 1:

```
function s = coefsSplineCuad(x, y, d_k, k)
  # Número de intervalos
  n = length(x) - 1;

  # 1, x, x²
  A(:,1) = [ones(n+1,1); 0];
  A(:,2) = [x'          ; 1];
  A(:,3) = [x'.^2       ; 2.*x(k+1)];

  # Potencias truncadas
  for j = 4 : n + 2
    pot    =  @(t) (t > x(j-2)) .* (t - x(j-2));
    A(:,j) = [pot(x').^2; 2.*pot(x(k+1))];
  end

  # Resolución del sistema
  s = A \ [y' ; d_k];

end
```

La función toma como argumentos los nodos y valores en los nodos, así como la derivada en un nodo k. Para obtener el spline aplicamos el método global:

Definimos la matriz columna a columna: las **3 primeras columnas** corresponden a los valores de `x` en $1, x, x^2$, así como los valores de la derivada:

```
A(:,1) = [ones(n+1,1); 0];
A(:,2) = [x'          ; 1];
A(:,3) = [x'.^2       ; 2.*x(k+1)];
```

Una vez hecho esto pasamos a las **potencias truncadas**. Para ello, en cada columna:

- Definimos una función `pot` correspondiente a la potencia truncada en el valor correspondiente: `pot = @(t) (t > x(j-2)) .* (t - x(j-2))`. Como *Octave* tiene tipos dinámicos convertirá `(t > x(j-2))` a 1 o 0. De esta forma, $pot(x) = (x - x_{j-1})_+$.

- Aplicamos `pot` a `x` en cada columna, añadiendo el valor de la derivada.

Definida la matriz del sistema podemos calcular finalmente la solución empleando la **división izquierda**.