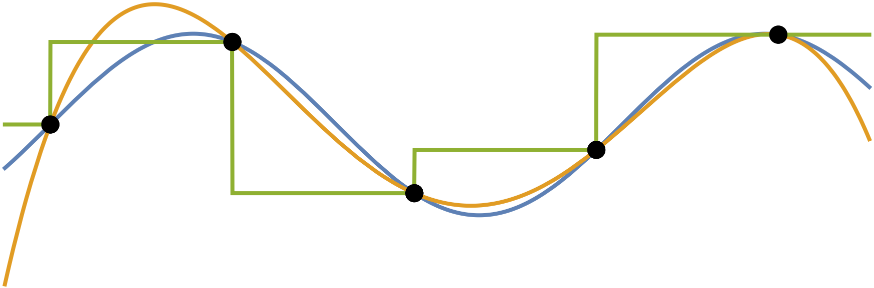
Programación y Métodos Numéricos

Interpolación

Profesor: B. Toledo

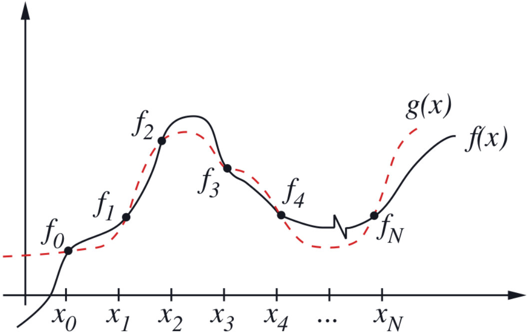
26 de abril de 2021

Problema general



Dados n puntos, encuentre una curva que pase por ellos, dadas ciertas restricciones.
Estos puntos pueden venir de simulaciones computacionales o mediciones experimentales.
El contexto fija las restricciones.

Interpolación de Lagrange



En la figura, $f(x)$ representa la función original y $g(x)$ su interpolación (aproximación).
Los puntos $(x_k, f_k),\ k = 0, 1, 2, \ldots, N$, son los **puntos de interpolación** o **nodos**. Entonces, queremos encontrar $g(x)$ tal que,

$$\boxed{f_k = g(x_k)}$$

Base de Lagrange (álgebra lineal)

Sean $\{V_j(x)\}_{j=0,\ldots,N}$, **polinomios** de grado $m \geq N$, tales que

$$\boxed{V_j(x_k) = \delta_{jk}}$$

De la definición para la base de Lagrange, resulta que

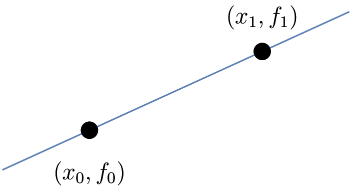
$$g(x) = \sum_{j=0}^{N} f_j V_j(x),$$

ya que

$$g(x_k) = \sum_{j=0}^{N} f_j V_j(x_k) = \sum_{j=0}^{N} f_j \delta_{jk} = f_k,$$

lo que define al interpolador g.

Base de Lagrange (álgebra lineal)



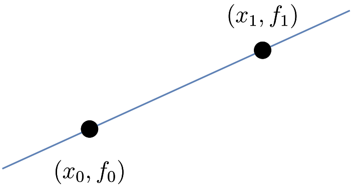
Encontremos la ecuación de la recta,

$$y = ax + b \Rightarrow \begin{array}{l} f_0 = a\,x_0 + b \\ f_1 = a\,x_1 + b \end{array}$$

de donde resulta

$$y = \frac{f_0 - f_1}{x_0 - x_1}\,x + \frac{f_1 x_0 - f_0 x_1}{x_0 - x_1} = f_0\,\frac{x - x_1}{x_0 - x_1} - f_1\,\frac{x - x_0}{x_0 - x_1}$$

Base de Lagrange (álgebra lineal)



Para estos dos puntos, es claro que la base es de la forma

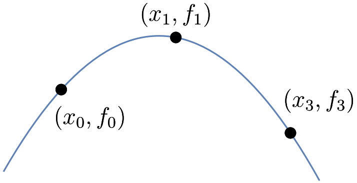
$$V_j(x_k) = \delta_{jk} \Rightarrow V_{0,1} \sim \alpha(x - x_{0,1}),$$

entonces,

$$V_0(x) = \frac{x - x_1}{x_0 - x_1}, \quad V_1(x) = \frac{x - x_0}{x_1 - x_0}$$

$$g(x) = f_0\, V_0(x) + f_1\, V_1(x) = f_0\, \frac{x - x_1}{x_0 - x_1} + f_1\, \frac{x - x_0}{x_1 - x_0}$$

Base de Lagrange (álgebra lineal)



Para tres puntos se tendrá

$$V_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}, \quad V_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}$$

$$V_2(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

$$g(x) = f_0\, V_0(x) + f_1\, V_1(x) + f_2\, V_2(x).$$

Base de Lagrange (álgebra lineal)

En general se obtiene

$$V_k(x) = \frac{(x - x_0)(x - x_1)\dots(x - x_{k-1})(x - x_{k+1})\dots(x - x_N)}{(x_k - x_0)(x_k - x_1)\dots(x_k - x_{k-1})(x_k - x_{k+1})\dots(x_k - x_N)},$$

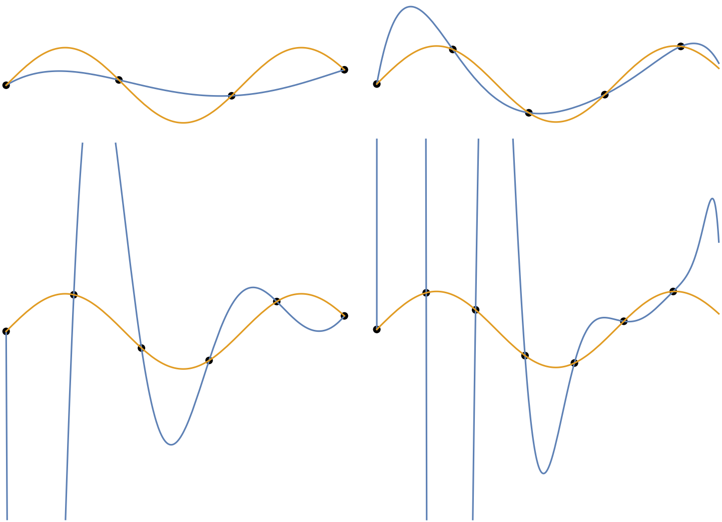
con lo que se define el interpolador

$$g(x) = \sum_{k=0}^{N} f_k \, V_k(x),$$

que pasa por los puntos (x_k, f_k), para $k = 0, 1, 2, \dots, N$.

Base de Lagrange (no convergencia)

Problemas!



Interpolación de Lagrange

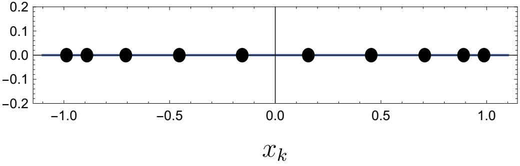
Una posible solución es usar nodos no equiespacidos. Una de estas soluciones corresponde a los **nodos de Chebyshev**, que son las raices de los polinomios del mismo nombre,

$$T_n(x) = \cos(n \ \mathrm{arc}\cos(x)),$$

con ceros,

$$x_k = \cos\left(\frac{\pi}{2}\frac{2k-1}{n}\right), \quad k = 1, \ldots, n.$$

Para el caso $n = 10$, la distribución de nodos es



Interpolación de Lagrange (nodos de Chebyshev)

Para mapear a un untervalo $[a,b]$ cualquiera escribimos

$$x_{n-k+1} = \frac{1}{2}(a+b) + \frac{1}{2}(b-a)\cos\left(\frac{\pi}{2}\frac{2k-1}{n}\right), \quad k = n, \ldots, 1.$$

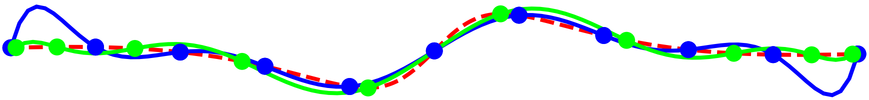
Supongamos la función,

$$f(x) = \frac{\sin(x)}{1+x^2},$$

que tiene variaciones de amplitud y es oscilante.

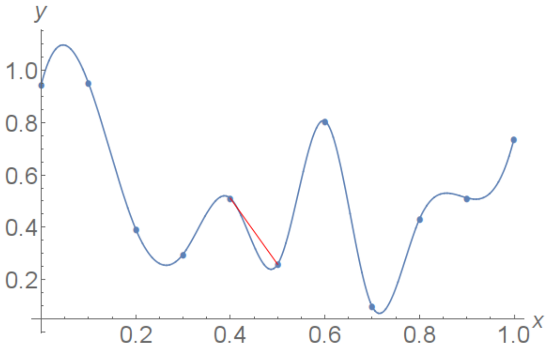
Interpolación de Lagrange (nodos de Chebyshev)

$$f(x) = \frac{\sin(x)}{1+x^2},$$



Splines (cúbicos)

A diferencia de la interpolación de Lagrange (incluso usando nodos de Chebyshev), que corresponde a un interpolador global, un spline es un interpolador local. Es decir, considera un entorno acotado de cada punto. Consideremos la siguiente figura,



Splines (cúbicos)

Los puntos representan datos, la curva que los une, un spline cúbico, $S(x)$. Como un spline es un interpolador local, vamos a considerar dos puntos, los que aparecen unidos por la linea roja. Esta linea roja, es una interpolación de *Lagrange* lineal. En general, la partición del intervalo en x no necesita ser uniforme, definamos,

$$h_i = X_{i+1} - X_i,$$

como la distancia entre dos puntos consecutivos. Ahora, un punto importante es que nuestro interpolador lineal, interpola la segunda derivada. Ya que el spline cúbico corresponde a un polinomio de tercer grado, su segunda derivada es una función lineal. Tenemos,

$$S_i''(x) = \frac{M_{i+1}(x - X_i)}{h_i} - \frac{M_i(x - X_{i+1})}{h_i}$$

Splines (cúbicos)

$$S_i''(x) = \frac{M_{i+1}(x - X_i)}{h_i} - \frac{M_i(x - X_{i+1})}{h_i}$$

que corresponde a un interpolador de Lagrange lineal, asociado a la linea roja en la figura. En esta ecuación, M_i, es el valor de la segunda derivada del spline $S(x)$, evaluado en X_i, que por ahora desconocemos.

Integrando dos veces, obtenemos,

$$S_i(x) = B_i + A_i(x - X_i) + \frac{M_{i+1}(x - X_i)^3}{6h_i} - \frac{M_i(x - X_{i+1})^3}{6h_i}$$

donde A_i y B_i son constantes integración por determinar.

Splines (cúbicos)

Teniendo en cuenta que,

$$y_i = S_i(X_i),$$

además de la continuidad de la función, de su primera y segunda derivada, podemos resolver los B_i,

$$B_i = y_i - \frac{1}{6}h_i^2 M_i,$$

insertando estos B_i en la expresión general para $S_i(x)$, podemos resolver ahora los A_i,

$$A_i = \frac{1}{6}h_i(M_i - M_{i+1}) + \frac{y_{i+1} - y_i}{h_i},$$

Splines (cúbicos)

de lo que resulta,

$$h_{i-1}M_{i-1}+2(h_{i-1}-h_i)M_i+h_iM_{i+1}+\frac{6(y_i-y_{i-1})}{h_{i-1}}+\frac{6(y_i-y_{i+1})}{h_i}=0.$$

Definamos,

$$\begin{aligned} a_i &= h_{i-1}, \\ b_i &= 2(h_i+h_{i-1}), \\ c_i &= h_i, \\ r_i &= -\frac{6(y_i-y_{i-1})}{h_{i-1}}-\frac{6(y_i-y_{i+1})}{h_i}, \end{aligned}$$

Splines (cúbicos)

y suponemos que $M_0 = 0$ y $M_{n-1} = 0$, lo que le da el adjetivo "natural" a este spline. Notemos que puede escribirse en forma matricial,

$$\begin{pmatrix} b_1 & c_1 & 0 & 0 & 0 & 0 & \dots \\ a_2 & b_2 & c_2 & 0 & 0 & 0 & \dots \\ 0 & a_3 & b_3 & c_3 & 0 & 0 & \dots \\ 0 & 0 & a_4 & b_4 & c_4 & 0 & \dots \\ 0 & 0 & 0 & a_5 & b_5 & c_5 & \dots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & & \dots & a_{n-2} & b_{n-2} \end{pmatrix} \begin{pmatrix} M_1 \\ M_2 \\ M_3 \\ M_4 \\ M_5 \\ \vdots \\ M_{n-2} \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ \vdots \\ r_{n-2} \end{pmatrix}$$

al ser tridiagonal la matriz, puede resolverse para los M_i de forma muy eficiente por el **algoritmo de Thomas**.

interpolacion.cc

```
#include <fstream>
#include <iostream>
#include <string>
#include <iomanip>
#include "ilagrange.h"

using namespace std;

int main()
{
  int N = 100, count = 0;
  double x, y, xmin, xmax;
  Matrix datos(20,2);
```

```cpp
ifstream infile("data_low.dat");
ofstream outfile("interp.dat");

while(infile>>x>>y)
{
  datos(count,0) = x;
  datos(count,1) = y;
  count++;
}

int nf = datos.f();

xmin = datos(0,0);
xmax = datos(nf-1,0);
```

```
  ILagrange intp(datos);//Spline sp(datos);

  for (int j=0;j<N;j++)
  {
    x = xmin + j*(xmax-xmin)/(N-1);
    y = intp(x);//sp(x);
    outfile
    <<scientific<<setw(15)<<setfill('␣')
    <<x<<"␣"
    <<scientific<<setw(15)<<setfill('␣')
    <<y<<endl;
  }

  return 0;
}
```

ilagrange.h

```
#ifndef LAGR_H
#define LAGR_H

#include <iostream>
#include <cmath>
#include "matrix.h"

using namespace std;

typedef unsigned int uint;
```

ilagrange.h

```
class ILagrange
{
  private:
  Matrix puntos;
  uint N;

  public:
  ILagrange(const Matrix &);
  ~ILagrange();
  double operator()(double);
  double lagrange(double,uint);
};

#endif
```