Universidad Nacional Autónoma de México

Escuela Nacional de Estudios Superiores

Unidad Juriquilla

Computación II

Practica 5

Método de interpolación Spline cuadrática

Bruno Arturo López Pacheco

Dr. Ulises Olivares Pinto

No. De Cuenta 317347140

Licenciatura en Tecnología

Objetivo

Realizar el método de la interpolación Spline cuadrática para interpolar un conjunto de puntos de entrada en un plano. Hacer uso de un dataset disponibles en bancos de datos para probar este método con datos reales.

Marco teórico

Método de interpolación Spline cuadrática

En estos splines, un polinomio cuadrático se aproxima los datos entre dos datos de puntos consecutivos. Dados $(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1}), (x_n, y_n)$, se ajustan splines cuadráticos a los datos. Estos datos están dados por

$$f(x) = a_1x^2 + b_1x + c_1, \quad x_0 \le x \le x_1$$

$$= a_2x^2 + b_2x + c_2\ , \quad x_1 \le x \le x_2$$

$$.$$

$$.$$

$$.$$

$$= a_nx^2 + b_nx + c_n\ x_{n-1} \le x \le x_n$$

Así que, ¿cómo se encuentran los coeficientes de los splines cuadráticos? Existen 3n tal que los coeficientes

$$a_i \quad i = 1,2, \ldots, n$$

$$b_i \quad i = 1,2, \ldots, n$$

$$c_i \quad i = 1,2, \ldots, n$$

Para encontrar a los 3n desconocidos, uno necesita establecer 3n ecuaciones y luego simultáneamente resolverlos. Estas 3n ecuaciones se encuentra como:

1.- Cada Spline cuadrático va a través de dos puntos de datos consecutivos

$$a_1x_0^2 + b_1x_0 + c_1 = f(x_0)$$

$$a_1x_1^2 + b_1x_1 + c_1 = f(x_1)$$

$$.$$

$$.$$

$$.$$

$$a_ix_{i-1}^2 + b_ix_{i-1} + c_1 = f(x_{i-1})$$

$$a_ix_i^2 + b_ix_i + c_1 = f(x_i)$$

$$.$$

.

$$a_n x_{n-1}^2 + b_n x_{n-1} + c_n = f(x_{n-1})$$

$$a_n x_n^2 + b_n x_n + c_n = f(x_n)$$

Esta condición da 2n ecuaciones porque hay n splines cuadráticos que van a través de dos puntos de datos consecutivos.

2.- Las primeras derivadas de dos splines cuadráticos son continuos en los puntos internos. Por ejemplo, la derivada del primer Spline $a_1x^2 + b_1x + c_1$ es $2a_1x + b_1$. La derivada del segundo Spline $a_2x^2 + b_2x + c_2$ es $2a_2x + b_2$; y los dos iguales cuando x=xi, da

$$2a_1x_1 + b_1 = 2a_2x_1 + b_2$$

$$2a_1x_1 + b_1 - 2a_2x_1 - b_2 = 0$$

De manera similar en los puntos interiores

$$2a_2x_2 + b_2 - 2a_3x_2 - b_3 = 0$$

.

.

.

$$2a_ix_i + b_i - 2a_{i+1}x_i - b_{i+1} = 0$$

.

.

.

$$2a_{n-1}x_{n-1} + b_{n-1} - 2x_nx_{n-1} - b_n = 0$$

Metodología

Pseudocódigo del método de interpolación Spline cuadrática.

Para crear la matriz con los polinomios Spline necesitamos dos entradas, los arreglos “x” y “y”; que debido a la implementación, “x” será “t”, y “y” será “v”.

```
n=tamaño(t)
numeroSplines=n-1
numeroDerivadas=numeroSplines-1 #numero de derivadas

splines=[]
derivatives=[]

Para i en rango(nsplines):
  primerSpline=[t[i]**2,t[i],1]
```

```
    segundoSpline=[t[i+1]**2,t[i+1],1]
    splines.append(primerSpline)
    splines.append(segundoSpline)

  Para i en rango(numeroDerivadas):
    eqn=[2*t[i+1],1,0,-2*t[i+1],-1]
    derivatives.append(eqn)

  M=np.zeros((3*n-3,3*n-2))

  paso=-3

  Para i en rango(tamaño(splines)):
    Si (i%2==0):
      paso=paso+3
    Para j en rango(0,tamaño(splines[0])):
      M[i][j+paso]=splines[i][j]

  paso=-3

  Para i en rango(tamaño(splines),tamaño(splines)+numeroDerivadas):
    paso=paso+3
    Para j en rango(tamaño(derivatives[0])):
      M[i][j+paso]=derivatives[i-len(splines)][j]

  M[len(M)-1][0]=1

  paso=0
  Para i en rango(numeroSplines):
    M[i+paso][tamaño(M)]=v[i]
    paso=paso+1
    M[i+paso][tamaño(M)]=v[i+1]
  Regresar M
```

Después de tener nuestra matriz, se puede utilizar algún método para resolver un sistema de ecuaciones lineal. Para este caso, se usó el método de eliminación Gauss Jordan.

Los valores de los coeficientes serán los coeficientes de los polinomios Spline, para así tener la interpolación de los puntos.

Pruebas

Primero se hicieron pruebas con arreglos que se introducían manualmente, para así comprobar el buen funcionamiento del algoritmo.

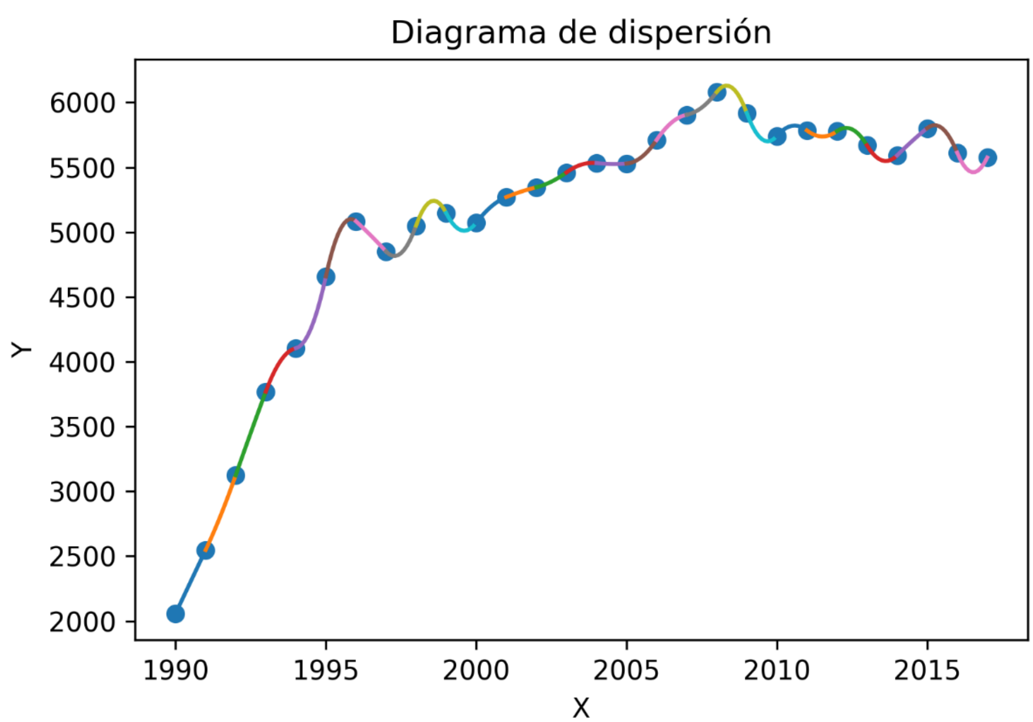
Después de hizo un módulo para probar con datos aleatorios, pero de mayor tamaño, incluso de hicieron pruebas con arreglos de 1000 datos. El pseudocódigo es el siguiente:

```
def valuesRandomInt(self,minX,maxX,minRandom,maxRandom):
    #Entradas:minimo valor de X, máximo valor de X
    #          intervalo mínimo y máximo de valores aleatorios
    X=arreglo(minX,maxX+1,1)
    Y=arreglo De Ceros Como El Vector X

    Para i en rango(len(Y)):
      Y[i]=EnterosAleatorios(minRandom,maxRandom+1)

    return X,Y
```

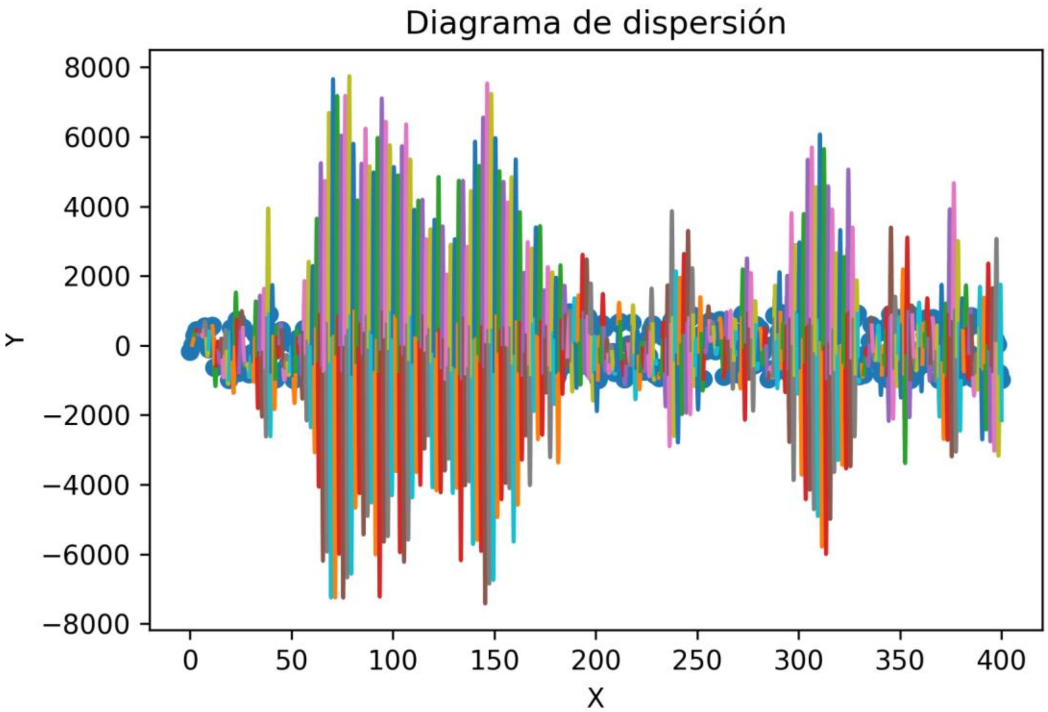
Después se usó un conjunto de datos obtenido de la colección de datos de "Our world in data", el dataset contiene información de las muertes por WIH, incidencia de casos y prevalencia de casos del VIH. Viene información de cada país por año. Para probar este algoritmo se usó la información del país de México acotando el tiempo entre el año 2013 y 2017.

Resultados

Se obtuvo la siguiente interpolación dados los puntos del dataset.



Y en el rendimiento del algoritmo, haciendo uso del módulo de datos aleatorios, se pudieron analizar hasta 400 datos. Teniendo una interpolación como



Teniendo un tiempo de ejecución de 754.1829025745392 segundos.

Conclusión

Este método de interpolación Spline es muy fácil de implementar, pues se trata de “armar” una matriz y resolverla por el método de eliminación Gauss Jordan. Es también un buen método muy siempre se ajusta totalmente a los datos, existe un polinomio entre dos datos, y es por esto que no tiene alteraciones o errores en los puntos extremos como en la interpolación de Newton.

Este algoritmo permite introducir más datos sin producir errores muy grandes en las interpolaciones. También este algoritmo es muy eficiente porque las interpolaciones no tardan demasiado, en lo que más puede tardar, es en resolver el sistema de ecuaciones.

Referencias

Chapra, S., & Canale, R. (2011). METODOS NUMERICOS PARA INGENIEROS (Spanish Edition) (6th ed.). McGraw-Hill Interamericana de España S.L.

Datos recabados de: Our World in Data https://ourworldindata.org/health-meta#infectious-diseases