Interpolación

Isabela Acevedo García, Juan Manuel Cuéllar

Pontificia Universidad Javeriana Cali

Cali, Colombia

isa43461@javerianacali.edu.co
juanma1909@javerianacali.edu.co

Abstract— En el presente documento se explicará los procesos que se llevaron a cabo para resolver el problema de interpolación con el método de Lagrange, Newton, el método lineal a trozos y el polinómico.

I. INTRODUCCIÓN

Se desarrollaron dos archivos en Python de tal forma que cada uno cumpliera con los requisitos solicitados, el primer archivo resuelve el problema de interpolación, es decir que en este archivo se encuentra el método de Lagrange, Newton, lineal a trozos y polinómico, también las funciones que necesitan para su desarrollo. El segundo archivo se hizo para realizar las simulaciones (main).

II. DESARROLLO DE CONTENIDO.

A. Materiales y métodos.

➔ **Materiales:**

- ***Python:*** Este fue el lenguaje utilizado para desarrollar los métodos de ecuaciones normales y ortogonalización. *"Python es un lenguaje de programación de código abierto, orientado a objetos, muy simple y fácil de entender. Tiene una sintaxis sencilla que cuenta con una vasta biblioteca de herramientas, que hacen de Python un lenguaje de programación único."* [1].
- ***Función round***: Esta es una función de Python utilizada para redondear los dígitos. *"La función **round** redondea un número a una precisión dada por el número de cifras decimales indicadas como argumento. Si no se indica el número de cifras decimales, se toma 0 como valor por defecto."* [2].
- ***Librería Statistics:*** Esta librería ayudó a sacar la media del error. ***"****El módulo statistics implementa muchas fórmulas estadísticas comunes para cálculos eficientes usando varios tipos numéricos de Python (int, float, Decimal, y Fracción)." [3].*
- ***Base de Datos No 2:*** Esta base de datos hace regencia al precio máximo del dólar por día, contiene 57 registros los cuales van desde Julio hasta Octubre del 2020. Esta Base fue tomada de Investing.com
- ***Librería Numpy:*** Esta librería ayudó a hacer operaciones de álgebra lineal de forma eficiente. *"NumPy es un paquete de Python que significa "Numerical Python", es la librería principal para la informática científica, proporciona potentes estructuras de datos, implementando matrices y matrices multidimensionales. Estas estructuras de datos garantizan cálculos eficientes con matrices." [4].*
- ***Librería Matplotlib:*** Esta librería ayudó a graficar el resultado de los dos métodos. *"Matplotlib es una librería de trazado utilizada para gráficos 2D en lenguaje de programación Python, es muy flexible y tiene muchos valores predeterminados incorporados que te ayudarán muchísimo en tu trabajo."* [5].
- ***Libreria time:*** Esta librería ayudó para sacar el tiempo de los algoritmos. *"El módulo time brinda acceso a varios tipos diferentes de relojes, cada uno útil para diferentes propósitos. "*[6]

:

➔ **Metodología**

Para la elaboración del algoritmo de algoritmo de interpolación de Newton se hizo uso del polinomio interpolante de Newton, donde para un conjunto (t_i, y_i) con i = 1,2,3...n: $P_{n-1} = x_1 + x_2(t - t_1) + x_3(t - t_1)(t - t_2) + ... + x_n(t - t_1)...(t - t_{n-1})$.
para poder hallar los valores de x_i se construye una matriz A, la cual tiene una forma triangular inferior, por lo que usando el método de sustitución sucesiva hacia adelante, se pueden obtener fácilmente los valores. ya con estos valores resueltos, se puede pasar a evaluar el polinomio interpolante para cualquier valor de t tal que $t_1 \leq t \leq t_n$.

Para la construcción de la interpolación lineal a trozos se definen primero los trozos en la función los cuales están delimitados por los puntos de entrenamiento del conjunto (t_i, y_i), siendo los delimitantes cada pareja de puntos contiguos. para cada trozo se utiliza la ecuación de la recta $y = mx + b$ donde m es la razón de cambio y b es el y-intercepto. hallando m mediante $m = \frac{\Delta y}{\Delta x}$ y reemplazando en la ecuación mencionada anteriormente uno de los puntos limitantes del trozo, se obtiene b. Esto se hace para cada uno de los trozos que hayan, una vez hecho esto si se desea evaluar un punto primero se debe hallar el trozo al que pertenece para así utilizar la ecuación de la recta adecuada.

Para hacer lagrange, primero se debían estudiar unas fórmulas para poder desarrollar el algoritmo.

$$l_j(t) = \frac{\prod_{k=1, k\neq j}^{n} (t - t_k)}{\prod_{k=1, k\neq j}^{n} (t_j - t_k)}$$

$$l_j(t_i) = \left\{ \begin{array}{l} 1, \ si \ i = j \\ 0, \ si \ i \neq j \end{array} \right\}$$

Basado en estas fórmulas, se pasa a evaluar los t cuestión con la información de las tablas y así lograr el resultado, se aplica tal cual la fórmula.

Resultados de la primera base de datos (Temperatura).

➢ ***Ejemplo número 1 (Ambos métodos)***

- y = [31.0, 23.2, 22.4, 21.6, 20.8, 19.8, 19.8, 19.4, 19.1, 18.5, 18.1, 17.8, 17.6, 17.3, 17.2, 17.2, 18.4, 19.5, 21.7, 24.7, 26.5, 27.6, 28.0, 27.6, 26.3, 24.9, 22.9, 20.9, 19.9, 19.6, 19.0, 18.6, 18.3, 18.0, 17.7, 17.4, 17.1, 16.9, 16.7, 16.4, 17.6, 19.6, 21.6, 23.8, 25.9, 26.5, 26.8, 26.4, 25.6, 24.1, 21.9, 20.0, 19.0, 18.7, 18.5, 18.4, 18.1, 18.0, 17.6, 17.4, 17.6, 17.6, 17.4, 17.0, 18.0, 19.6, 21.2, 23.5, 25.5, 26.9, 26.6, 25.5, 23.7, 22.0, 20.8, 19.4, 18.7, 18.6, 18.1, 18.0, 18.1, 18.3, 17.9, 17.6]
- t = np.arange(0, 8.4, 0.1)
- c = 8

- **Polinomial**
  Error Medio: 3.7789573361709454
  Desviación Estándar: 5.299529819831897

- **A Trozos**
  Error Medio: 3.1936507936507934
  Desviación Estándar: 3.139788715764522

- **Lagrange**
  Error Medio: 3.77895733602239
  Desviación Estándar: 5.299529819284454

- **Newton**
  Error Medio: 3.7789573360224606
  Desviación Estándar: 5.299529819284883

- **Tiempos de Ejecución:**
  Polinomial: 0.0027103424072265625
  A Trozos: 0.15982437133789062
  Newton: 0.013099908828735352
  Lagrange: 0.001992464065551758

Fig. 1 Es el ejemplo #1 graficado en Python con la librería matplotlib.

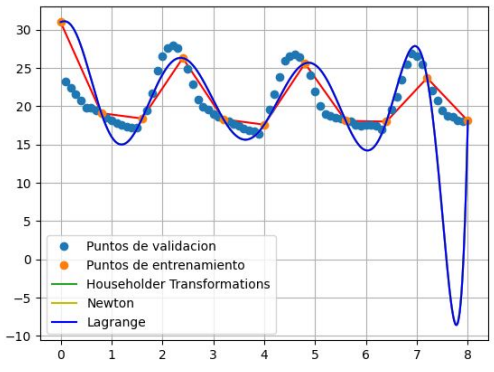

Fig. 1 Ejemplo c =8.

- ***Análisis y discusión***

Se puede ver que Newton y Lagrange son prácticamente iguales, es decir que la diferencia en resultados en ambas funciones es mínima, se puede ver que de igual manera todas las funciones pasan por los puntos exactos de entrenamiento y que la función de trozos es mucho más rígida y notable (La función de color rojo). Partiendo de la base de datos de 8 en 8, se puede ver que hace que las funciones se asimilen a la función inicial es decir a los puntos de validación, sin embargo hay ruido muy notable entre los dos últimos puntos. Se puede ver claramente que la interpolación a trozos tiene un error menor, a las demás, esto es debido a que se evitan que se creen oscilaciones como las que hay entre los 2 últimos puntos de entrenamiento

➢ ***Ejemplo número 2 (Ambos métodos)***

- y = [31.0, 23.2, 22.4, 21.6, 20.8, 19.8, 19.8, 19.4, 19.1, 18.5, 18.1, 17.8, 17.6, 17.3, 17.2, 17.2, 18.4, 19.5, 21.7, 24.7, 26.5, 27.6, 28.0, 27.6, 26.3, 24.9, 22.9, 20.9, 19.9, 19.6, 19.0, 18.6, 18.3, 18.0, 17.7, 17.4, 17.1, 16.9, 16.7, 16.4, 17.6, 19.6, 21.6, 23.8, 25.9, 26.5, 26.8, 26.4, 25.6, 24.1, 21.9, 20.0, 19.0, 18.7, 18.5, 18.4, 18.1, 18.0, 17.6, 17.4, 17.6, 17.6, 17.4, 17.0, 18.0, 19.6, 21.2, 23.5, 25.5, 26.9, 26.6, 25.5, 23.7, 22.0, 20.8, 19.4, 18.7, 18.6, 18.1, 18.0, 18.1, 18.3, 17.9, 17.6]
- t = np.arange(0, 8.4, 0.1)
- c = 9

- **Polinomial**
  Error Medio: 5.6348603277655585
  Desviación Estándar: 8.140345083215104

- **A Trozos**
  Error Medio: 4.9984375
  Desviación Estándar: 3.7541293453911635

- **Lagrange**
  Error Medio: 5.6348603282247325
  Desviación Estándar: 8.140345083754731

- **Newton**
  Error Medio: 5.634860328224745
  Desviación Estándar: 8.14034508375473

- **Tiempos de Ejecución:**
  Polinomial: 0.002496004104614258
  A Trozos: 0.15509247779846191
  Newton: 0.008265256881713867
  Lagrange: 0.001992464065551758

Fig. 2 Es el ejemplo #2 graficado en Python con la librería matplotlib.

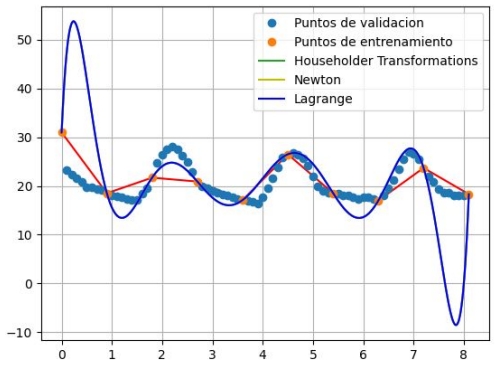

Fig. 2 Ejemplo c =9.

- ***Análisis y discusión***

Al igual que con el ejemplo anterior los polinomios creados por Newton, Lagrange y House Holder, son prácticamente iguales.. Partiendo de la base de datos de 9 en 9, se puede ver que hace que las funciones se asimilen a la función inicial es decir a los puntos de validación, sin embargo hay ruido bastante grande entre los primeros 2 puntos y los últimos 2 de entrenamiento puntos debido a que se alarga mucho la curva. Esto genera que el error de trozos sea menor a pesar de que los demás métodos se ajustan mucho mejor a los puntos en distintas partes de la gráfica, esto se ve reflejado en la desviación estándar, donde la desviación de los demás métodos es mucho mayor.

➢ ***Ejemplo número 3***

- y = [31.0, 23.2, 22.4, 21.6, 20.8, 19.8, 19.8, 19.4, 19.1, 18.5, 18.1, 17.8, 17.6, 17.3, 17.2, 17.2, 18.4, 19.5, 21.7, 24.7, 26.5, 27.6, 28.0, 27.6, 26.3, 24.9, 22.9, 20.9, 19.9, 19.6, 19.0, 18.6, 18.3, 18.0, 17.7, 17.4, 17.1, 16.9, 16.7, 16.4, 17.6, 19.6, 21.6, 23.8, 25.9, 26.5, 26.8, 26.4, 25.6, 24.1, 21.9, 20.0, 19.0, 18.7, 18.5, 18.4, 18.1, 18.0, 17.6, 17.4, 17.6, 17.6, 17.4, 17.0, 18.0, 19.6, 21.2, 23.5, 25.5, 26.9, 26.6, 25.5, 23.7, 22.0, 20.8, 19.4, 18.7, 18.6, 18.1, 18.0, 18.1, 18.3, 17.9, 17.6]
- t = np.arange(0, 8.4, 0.1)
- c = 10

- **Polinomial**
  Error Medio: 4.093154624100556
  Desviación Estándar: 4.616803343463669

- **A Trozos**
  Error Medio: 6.112063492063492
  Desviación Estándar: 4.72127712030997

- **Lagrange**
  Error Medio: 4.093154624098611
  Desviación Estándar: 4.616803343461719

- **Newton**
  Error Medio: 4.093154624098656
  Desviación Estándar: 4.61680334346182

- **Tiempos de Ejecución:**
  Polinomial: 0.0049479007720947266
  A Trozos: 0.14868497848510742
  Newton: 0.010619163513183594
  Lagrange: 0.002280712127685547

Fig. 3 Es el ejemplo #3 Graficado en Python con la librería matplotlib.

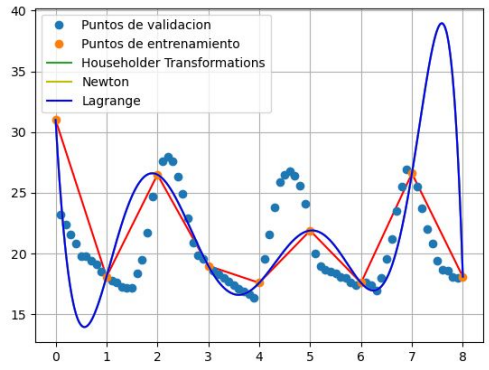

Fig. 3 Ejemplo con c=10.

- ***Análisis y discusión***

Al igual que con los ejemplos anteriores, la grafica arrojada por los metodos Lagrange, Newton y Polinomial (Householder) es basicamente la misa, y sus diferencias son apreciables unicamente si se acerca mucho a la grafica,, se puede ver que de igual manera todas las funciones pasan por los puntos exactos de entrenamiento y que la función de trozos es mucho más rígida y notable (La función de color rojo). Partiendo de la base de datos de 10 en 10, se puede ver que hace que las funciones se asimilen a la función inicial es decir a los puntos de validación, sin embargo hay oscilaciones demasiado pronunciadas entre los primeros 2 u los últimos 2 puntos de entrenamiento. Aquí se ve un claro beneficio al usar el método a trozos pues elimina por completo dichas oscilaciones.

B. *Resultados de la segunda base de datos*

➢ ***Ejemplo número 1***

- y = [3624, 3644, 3666, 3658, 3653, 3640, 3669, 3703, 3694, 3739, 3731, 3752, 3739, 3773, 3803, 3786, 3788, 3767, 3782, 3778, 3763, 3777, 3796, 3803, 3799, 3795, 3805, 3848, 3859, 3888, 3881, 3850, 3837, 3767, 3741, 3691, 3696, 3723, 3717, 3773, 3739, 3716, 3726, 3723, 3704, 3720, 3729, 3739, 3802, 3841, 3875, 3908, 3893, 3887, 3896, 3907, 3858]
- t = np.arange(0, 5.8, 0.1)
- c = 8
- **Polinomial**

  Error Medio: 38.646799723692745

Desviación Estándar: 36.896690734999126

- **A Trozos**

  Error Medio: 31.053571428571427

  Desviación Estándar: 28.35699479690266

- **Lagrange**

  Error Medio: 38.646799723713016

  Desviación Estándar: 36.89669073507116

- **Newton**

  Error Medio: 38.646799723712846

  Desviación Estándar: 36.89669073507108

- **Tiempos de Ejecución:**

  Polinomial: 0.004281759262084961

  A Trozos: 0.15628981590270996

  Newton: 0.0066454410552978516

  Lagrange: 0.0009982585906982422

Fig. 1 Es el ejemplo #1 graficado en Python con la librería matplotlib.

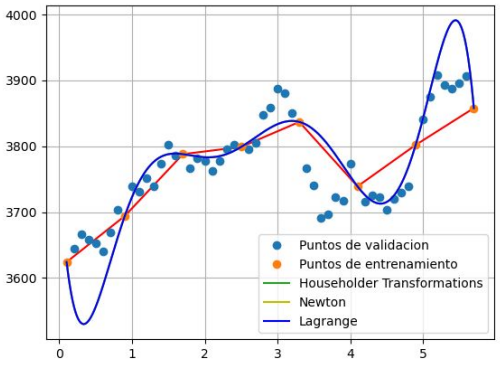

Fig. 1 Ejemplo con c = 8

.

- ***Análisis y discusión***

De igual manera que con la base de datos anterior, los metodos Lagrange, Newton y polinomial (Householder) aroojan una grafica casi identica , se puede ver que de igual manera todas las funciones pasan por los puntos exactos de entrenamiento y que la función de trozos es mucho más rígida y notable (La función de color rojo). Partiendo de la base de datos de 8 en 8, se puede ver que hace que las funciones se asimilen a la función inicial es decir a los puntos de validación, sin embargo hay ruido en todos los puntos pero entre algunos puntos más notorios que en otros, la figura aún no es muy clara, no tiene mucha forma.

➢ ***Ejemplo número 2***

- y = [3624, 3644, 3666, 3658, 3653, 3640, 3669, 3703, 3694, 3739, 3731, 3752, 3739, 3773, 3803, 3786, 3788, 3767, 3782, 3778, 3763, 3777, 3796, 3803, 3799, 3795, 3805, 3848, 3859, 3888, 3881, 3850, 3837, 3767, 3741, 3691, 3696, 3723, 3717, 3773, 3739, 3716, 3726, 3723, 3704, 3720, 3729, 3739, 3802, 3841, 3875, 3908, 3893, 3887, 3896, 3907, 3858]
- t = np.arange(0, 5.8, 0.1)
- c = 9
- **Polinomial**

  Error Medio: 47.85275961698482

  Desviación Estándar: 31.396133125072556

- **A Trozos**

  Error Medio: 86.57499999999995

  Desviación Estándar: 65.14863294805185

- **Lagrange**

  Error Medio: 47.85275961698679

  Desviación Estándar: 31.396133125070282

- **Newton**

  Error Medio: 47.852759616986965

  Desviación Estándar: 31.3961331250702

- **Tiempos de Ejecución:**

  Polinomial: 0.00431370735168457

  A Trozos: 0.1522355079650879

  Newton: 0.0071353912353515625

  Lagrange: 0.0009989738464355469

Fig. 2 Es el ejemplo #2 graficado en Python con la librería matplotlib.

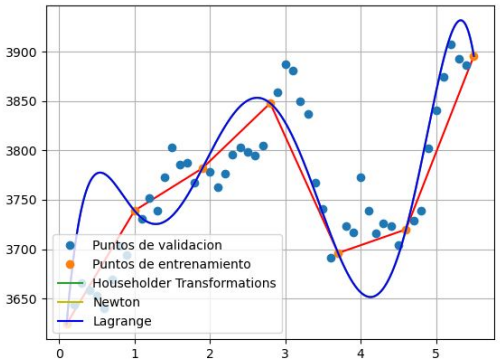

Fig. 2 Ejemplo con c=9.

- ***Análisis y discusión***

Nuevamente los demas metodos, aquellos que no es interpolacion lineal a trozos, dan un grafica visualmente identica, se puede ver que de igual manera todas las funciones pasan por los puntos exactos de entrenamiento y que la función de trozos es mucho más rígida y notable (La función de color rojo). Partiendo de la base de datos de 9 en 9, se puede ver que hace que las funciones se asimilen a la función inicial es decir a los puntos de validación, sin embargo hay ruido al principio ya que se forma la curva muy pronunciada, de igual manera ya empieza a tener más forma la función. Esta vez entre los últimos 2 puntos de entrenamiento no hubo tanto ruido como había ocurrido en los demás ejemplos.

➢ ***Ejemplo número 3.***

- y = [3624, 3644, 3666, 3658, 3653, 3640, 3669, 3703, 3694, 3739, 3731, 3752, 3739, 3773, 3803, 3786, 3788, 3767, 3782, 3778, 3763, 3777, 3796, 3803, 3799, 3795, 3805, 3848, 3859, 3888, 3881, 3850, 3837, 3767, 3741, 3691, 3696, 3723, 3717, 3773, 3739, 3716, 3726, 3723, 3704, 3720, 3729, 3739, 3802, 3841, 3875, 3908, 3893, 3887, 3896, 3907, 3858]
- t = np.arange(0, 5.8, 0.1)
- c = 10
- **Polinomial**
  Error Medio: 50.31428690555611
  Desviación Estándar: 36.39319574082442

- **A Trozos**
  Error Medio: 82.70555555555555
  Desviación Estándar: 64.78328807323874

- **Lagrange**
  Error Medio: 50.314286905555555
  Desviación Estándar: 36.393195740822996

- **Newton**
  Error Medio: 50.31428690555541
  Desviación Estándar: 36.39319574082302

- **Tiempos de Ejecución:**
  Polinomial: 0.0025517940521240234
  A Trozos: 0.14927339553833008
  Newton: 0.007778644561767578
  Lagrange: 0.0006589889526367188

Fig. 3 Es el ejemplo #3 Graficado en Python con la librería matplotlib.

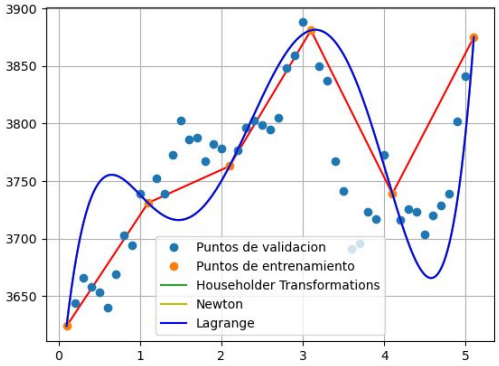

Fig. 3 Ejemplo con c=10.

- ***Análisis y discusión***

Los metodos Polinomial, Newton y Lagrange siguen dando una grafica practicamente identica, se puede ver que de igual manera todas las funciones pasan por los puntos exactos de entrenamiento y que la función de trozos es mucho más rígida y notable (La función de color rojo). Partiendo de la base de datos de 10 en 10, se puede ver que hace que las funciones se asimilen a la función inicial es decir a los puntos de validación, sin embargo hay ruido pero en esta parte se puede ver que ya la función va teniendo mucho más sentido respecto a los puntos de validación.

III. CONCLUSIONES

Primeramente, en términos de rendimiento Lagrange es el claro vencedor, siendo siempre el algoritmo con el tiempo de ejecución menor, tomando en cuenta que la diferencia

entre este método y Newton y polinomial (Householder), si lo que se busca es redimiento la decisión sobre cual método escoger es clara. Además de esto resultó curioso el hecho de que en la mayoría de ejemplos se presentan oscilaciones entre los primeros 2 puntos y los últimos puntos de entrenamiento, mientras que en el resto de puntos se realizaba un buen ajuste, esto hacía que la desviación estándar se disparara. Otra observación fue que el uso de la interpolación lineal a trozos elimina por completo dichas oscilaciones, y en algunos caso el error medio y la desviación estándar es mucho menor debido a esto, sin embargo para la segunda base de datos donde la dispersión de los puntos es un poco mayor, la interpolación lineal a trozos da un error mayor.

IV. REFERENCIAS

[1] A. Soloaga (2018, Octubre 19) "Principales Usos de Python".[Online]. Disponible en: https://www.akademus.es/blog/programacion/principales-usos-python/

[2] D. Barrueco. (2019, Enero 19). "ROUND". [Online]. Disponible en: https://www.interactivechaos.com/python/function/round

[3]. E. Rico (2018). "Statistics-Cálculos estadísticos" [Online]. Disponible en: https://rico-schmidt.name/pymotw-3/statistics/index.html

[4]. L. González. (2018, Septiembre 21) "Introducción a la librería NumPy de Python – Parte 1 ``.[Online]. Disponible en: https://ligdigonzalez.com/introduccion-a-numpy-python-1/

[5]. L. González. (2018, Octubre 19). "Introducción a la Librería Matplotlib de Python – Parte 1 ``.[Online]. Disponible en:

https://ligdigonzalez.com/libreria-pandas-de-matplotlib-tutorial/

[6]. E. Rico (2018). "time- Hora del reloj" [Online]. Disponible en: https://rico-schmidt.name/pymotw-3/time/