Mínimos cuadrados lineales resueltos con ecuaciones normales y householder

Isabela Acevedo García

Pontificia Universidad Javeriana Cali

Cali, Colombia

isa43461@javerianacali.edu.co

***Abstract*—** En el presente documento se explicará los procesos que se llevaron a cabo para resolver el problema de mínimos cuadrados con el método de ecuaciones normales y el método por ortogonalización.

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 mínimos cuadrados, es decir que en este archivo se encuentra el método de ecuaciones normales, el método por ortogonalización y 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].*
- ***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**

La metodología de cada función depende del método que se iba a desarrollar. Para poder dar una solución a mínimos cuadrados con el método de ecuaciones

normales, primero debíamos tener claro cual iba a ser el orden a desarrollar, entonces si era de orden cuadrático se sabía por definición que tendría 3 parámetros. Por ejemplo, con la primera base de datos que escogí que es la temperatura en el valle del cauca, se llamaba a la función de ecuaciones normales con los parámetros t que viene siendo una lista de números decimales de 0 a 8.4 donde cada uno de los 84 datos tenía su respectivo y, que era cada una de las temperaturas. Se escogió números entre ese rango debido a que con enteros generaba error por el tamaño. Una vez en la función de ecuaciones normales tenemos que formar la matriz con los n parámetros, entonces la matriz iba a tener el número n de columnas y la longitud de la lista t, serían las filas. En cada columna se cumple la condición de que el t debe ser multiplicado por el índice correspondiente, por eso la primera columna está llena de 1 's debido a que cualquier pow(t,0) = 1 y así sucesivamente se hacía con cada elemento dependiendo en que iterador estaba cada número t, se le aplicaba el pow(t,i) hasta llenar toda esa fila. De tal manera de que si eran 3 parámetros y 5 datos. La primera fila de la matriz sería, pow(t,0), pow(t,1) y pow(t.2) y así con cada uno de los datos hasta completar la matriz A. Una vez con la matriz creada, se crea la matriz transpuesta At de esa misma matriz, y se multiplica su transpuesta con la matriz. Aparte, con la At se multiplica con la columna de los Y (se llama b en la función). Después con la matriz A se realiza la descomposición de Cholesky L, se le saca su transpuesta Lt y se llama a la función inferior (sistemas triangulares inferiores) para que realice su proceso con L y b. Con el resultado de este llamado (y). Se llama a la función de sistema triángulos superiores con Lt y el y sacado últimamente. Finalmente se retorna el resultado de la función superior.

Para resolver el método de ortogonalización. Primero debíamos tener claro cual iba a ser el orden a desarrollar, entonces si era de orden cuadrático se sabía por definición que tendría 3 parámetros. Por ejemplo, con la primera base de datos que escogí que es la temperatura en el valle del cauca, se llamaba a la función de HouseHolder con los parámetros t que viene siendo una lista de números decimales de 0 a 8.4 donde cada uno de los 84 datos tenía su respectivo y que era cada una de las temperaturas. Se escogió números entre ese rango debido a que con enteros generaba error por el tamaño. Una vez en la función de HouseHolder tenemos que formar la matriz con los n parámetros, de la misma forma como la creamos en la función de ecuaciones normales. Una vez ya creada nuestra matriz A, se hace un for desde 0 hasta el número de parámetros que hayan. Esto es para recorrer cada columna. Dentro de este ciclo creamos un arreglo auxiliar lleno de 0's para poner la norma, cogemos la columna la cual está apuntando en ese momento el iterador. Aniquilamos los números que estén por encima del pivote y le sacamos la norma a la columna. Una vez tenemos la norma, me aseguro de poner el signo contrario del pivote y se pone la norma con el signo contrario en el arreglo auxiliar. Después, restamos la columna en la que apunta el iterador con este arreglo auxiliar y esto nos da un un vector que llamaremos v. A esta v se le saca su transpuesta y se aplica la fórmula H = I - 2(v*vt / vt * v). Teniendo la H, multiplicamos esta por A y por b. y así sucesivamente se va a acumulando el proceso. Finalmente para obtener el resultado final, escogemos la matriz final de A definiendola en el rango de 0 al n (# de parámetros) tanto para columna como para fila y al vector b también se le pone ese rango. Estos resultados se mandan a la función superior donde finalmente obtenemos la respuesta final.

Resultados de la primera base de datos.

- ***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)
- n = 4
- Elapsed time of EcN: 0.0004737377 seconds.
- Elapsed time of HouseHolder: 0.0012068748 seconds.
- El error medio con EcN es: 2.9
- El error medio es HH: 2.9
- El error típico es Ec: 1.7873197716082951
- El error típico es HH: 1.7873197716082532

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

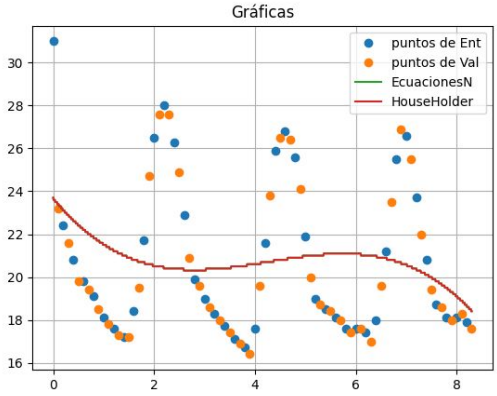

Fig. 1 Ejemplo de la bases de datos de temperatura con n=4.

- ***Análisis y discusión***
  Se puede ver que al ser de orden 3 (cubico), no se ve el polinomio con la forma de los puntos por los que debe pasar, sin embargo se trata de acomodar en los puntos iniciales y finales. Ambos métodos se dieron muy similares con diferencias tan mínimas que no son notables en la gráfica. Los tiempos son muy pequeños pero se nota la diferencia entre los dos ya que se ve más eficiente el de ecuaciones normales que el de HouseHolder.

➢ ***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)
- n = 8
- Elapsed time of EcN: 0.0005016327 seconds.
- Elapsed time of HouseHolder: 0.0018961430 seconds.
- El error medio con EcN es: 2.72
- El error medio es HH: 2.72
- El error típico es Ec: 1.5747180786026014
- El error típico es HH: 1.5747180717656941

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

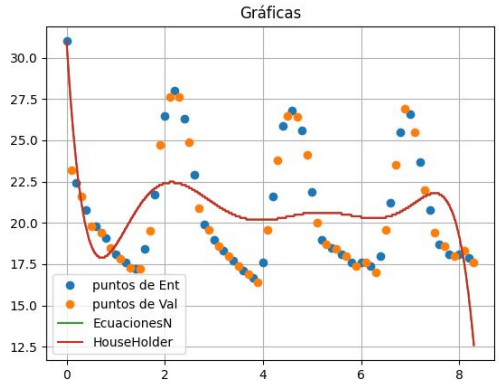

Fig. 2 Ejemplo n=8.

- ***Análisis y discusión***
  Se puede ver en la gráfica que al ser de orden 7, va tomando un poco más la forma del polinomio, en este punto ya ambos métodos tocan el punto inicial y se nota más clara la curvatura que tomara, Ambos métodos se dieron muy similares con diferencias tan mínimas que no son notables en la gráfica. Los tiempos son muy pequeños pero se nota la diferencia entre los dos ya que se ve más eficiente el de ecuaciones normales que el de HouseHolder.

➢ ***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)
- n = 12
- Elapsed time of EcN: 0.0008275509 seconds.
- Elapsed time of HouseHolder: 0.0032656193 seconds.
- El error medio con EcN es: 1.17
- El error medio es HH: 1.19
- El error típico es Ec: 0.9097187266122704
- El error típico es HH: 0.9598936263861073

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

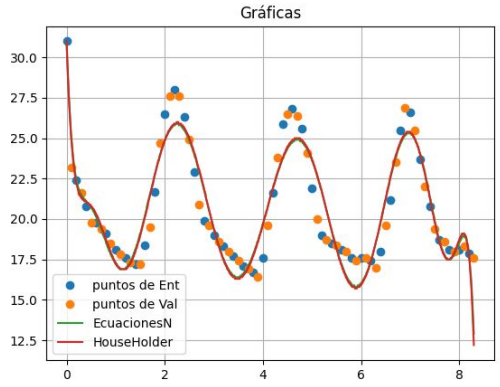

Fig. 3 Ejemplo con n=12.

- ***Análisis y discusión***
  Se puede ver que con un n más grande (orden 11) se va viendo muchísimo más claro el polinomio y que va tomando su respectiva forma, se trata de acercar bastante a los puntos. Ambos métodos se dieron muy similares con diferencias tan mínimas que no son notables en la gráfica. Los tiempos son muy pequeños pero se nota la diferencia entre los dos ya que se ve más eficiente el de ecuaciones normales que el de HouseHolder.

B. ***Resultados de la base de datos de punto de rocío.***

➢ ***Ejemplo número 1***

- y = [16.2, 18.1, 18.9, 18.5, 17.7, 17.7, 17.5, 17.3, 16.9, 16.7, 16.5, 16.3, 16.2, 15.9, 16.0, 16.7, 16.8, 17.6, 19.3, 18.2, 17.8, 17.8, 17.3, 16.0, 18.1, 18.1, 17.6, 17.0, 17.1, 16.8, 16.3, 16.2, 16.2, 16.0, 16.0, 15.7, 15.6, 15.3, 15.2, 15.8, 16.8, 17.4, 17.6, 17.6, 17.1, 18.0, 17.7, 17.4, 18.3, 18.3, 17.9, 17.3, 17.1, 16.9, 16.9, 16.3, 16.3, 16.1, 16.1, 16.8, 16.8, 16.5, 16.3, 16.9, 17.5, 17.9, 18.0, 17.7, 17.4, 17.2, 17.8, 18.0, 18.3, 17.9, 17.1, 16.7, 16.5, 16.1, 16.0, 16.4, 16.5, 16.3, 16.1]
- t = np.arange(0, 8.3, 0.1)
- n = 4
- Elapsed time of EcN: 0.0004022121 seconds.
- Elapsed time of HouseHolder: 0.0011303425 seconds.
- El error medio con EcN es: 0.67
- El error medio es HH: 0.67
- El error típico es Ec: 0.4009296918962632
- El error típico es HH: 0.4009296918962596

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

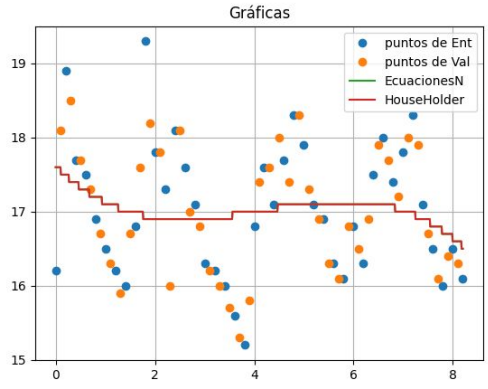

Fig. 1 Ejemplo con n=4

- ***Análisis y discusión***
  Se puede ver que al ser de orden 3 (cubico), no se ve el polinomio con la forma de los puntos por los que debe pasar, sin embargo se trata de acomodar en los puntos iniciales y finales. Ambos métodos se dieron muy similares con diferencias tan mínimas que no son notables en la gráfica. Los tiempos son muy pequeños pero se nota la diferencia entre los dos ya que se ve más eficiente el de ecuaciones normales que el de HouseHolder.

➢ ***Ejemplo número 2***

- y = [16.2, 18.1, 18.9, 18.5, 17.7, 17.7, 17.5, 17.3, 16.9, 16.7, 16.5, 16.3, 16.2, 15.9, 16.0, 16.7, 16.8, 17.6, 19.3, 18.2, 17.8, 17.8, 17.3, 16.0, 18.1, 18.1, 17.6, 17.0, 17.1, 16.8, 16.3, 16.2, 16.2, 16.0, 16.0, 15.7, 15.6, 15.3, 15.2, 15.8, 16.8, 17.4, 17.6, 17.6, 17.1, 18.0, 17.7, 17.4, 18.3, 18.3, 17.9, 17.3, 17.1, 16.9, 16.9, 16.3, 16.3, 16.1, 16.1, 16.8, 16.8, 16.5, 16.3, 16.9, 17.5, 17.9, 18.0, 17.7, 17.4, 17.2, 17.8, 18.0, 18.3, 17.9, 17.1, 16.7, 16.5, 16.1, 16.0, 16.4, 16.5, 16.3, 16.1]
- t = np.arange(0, 8.3, 0.1)
- n = 8

- Elapsed time of EcN: 0.0004901886 seconds.
- Elapsed time of HouseHolder: 0.0020096302 seconds.
- El error medio con EcN es: 0.68
- El error medio es HH: 0.68
- El error típico es Ec: 0.3813907074781343
- El error típico es HH: 0.3813907070702699

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

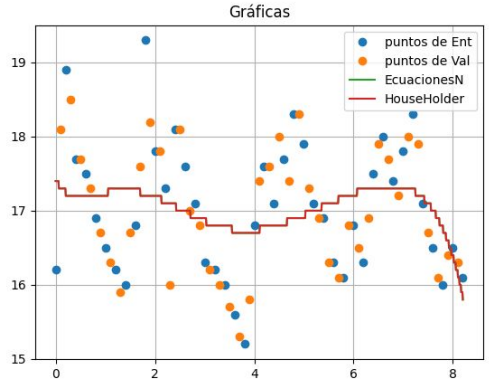

Fig. 2 Ejemplo con n=8.

- ***Análisis y discusión***
  Se puede ver en la gráfica que al ser de orden 7, va tomando un poco más la forma del polinomio, en este punto aún ambos métodos no tocan el punto inicial y se nota un poco más clara la curvatura que tomara, Ambos métodos se dieron muy similares con diferencias tan mínimas que no son notables en la gráfica. Los tiempos son muy pequeños pero se nota la diferencia entre los dos ya que se ve más eficiente el de ecuaciones normales que el de HouseHolder.

➢ ***Ejemplo número 3.***

- y = [16.2, 18.1, 18.9, 18.5, 17.7, 17.7, 17.5, 17.3, 16.9, 16.7, 16.5, 16.3, 16.2, 15.9, 16.0, 16.7, 16.8, 17.6, 19.3, 18.2, 17.8, 17.8, 17.3, 16.0, 18.1, 18.1, 17.6, 17.0, 17.1, 16.8, 16.3, 16.2, 16.2, 16.0, 16.0, 15.7, 15.6, 15.3, 15.2, 15.8, 16.8, 17.4, 17.6, 17.6, 17.1, 18.0, 17.7, 17.4, 18.3, 18.3, 17.9, 17.3, 17.1, 16.9, 16.9, 16.3, 16.3, 16.1, 16.1, 16.8, 16.8, 16.5, 16.3, 16.9, 17.5, 17.9, 18.0, 17.7, 17.4, 17.2, 17.8, 18.0, 18.3, 17.9, 17.1, 16.7, 16.5, 16.1, 16.0, 16.4, 16.5, 16.3, 16.1]
- t = np.arange(0, 8.3, 0.1)
- n = 12
- Elapsed time of EcN: 0.0006515980 seconds.
- Elapsed time of HouseHolder: 0.0034573078 seconds.
- El error medio con EcN es: 0.33
- El error medio es HH: 0.33
- El error típico es Ec: 0.34213722315145473
- El error típico es HH: 0.3442857588082975

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

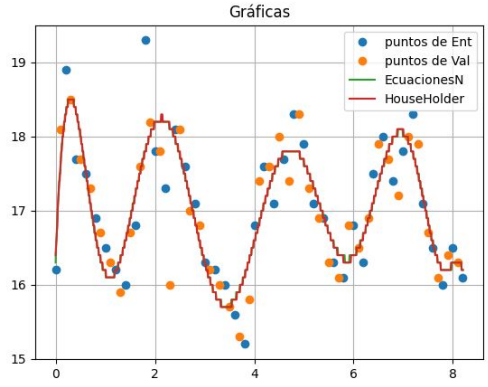

Fig. 3 Ejemplo con n=12.

- ***Análisis y discusión***
  Se puede ver que con un n más grande (orden 11) se va viendo muchísimo más claro el polinomio y que va tomando su respectiva forma, se trata de acercar bastante a los puntos. Ambos métodos se dieron muy similares con diferencias tan mínimas que no son notables en la gráfica. Los tiempos son muy pequeños pero se nota la diferencia entre los dos ya que se ve más eficiente el de ecuaciones normales que el de HouseHolder.

III. CONCLUSIONES

Finalmente para concluir, se puede decir que el método de ecuaciones normales es mucho más rápido que el householder pero ya que en el de ecuaciones normales se aplicó la función cholesky definida por Python, este no permite n mayores a cierto número. Con ambos métodos podemos llegar a una solución casi exacta con los 2 y a graficas muy similares.

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/