César Isaí García Cornejo
cesar.cornejo@cimat.mx
Computo Científico

Tarea 2

Semestre 3

CIMAT

Ejercicio 1
Implementar el algoritmo de Gram-Schmidt modificado 8.1 del Trefethen (p.58) para generar la descomoposición QR.

Solución:
Usando la tería de los proyectores ortogonales es que se encuetra un algoritmo mas *estable* para hacer Gram-Schmidt. Es decir se tiene un algoritmo para hacer factorización QR reducida.

El algoritmo implementado es tal que se ingresa una matriz A de dimensiones $m \times n$ y regresa una factoriación de la forma

$$A = \hat{Q}\hat{R}$$

donde $\hat{Q}$ y $\hat{R}$ son matrices de dimensión $m \times n$ y $n \times n$, respectivamente. Además la matriz $\hat{R}$ es una matriz triangular superior.

El código implementado en Python se basa en lo siguiente

```
    for i in range(n):

        R[i,i] = np.sqrt(np.dot(V[:,i],V[:,i]))
        Q[:,i] = V[:,i] / R[i,i]

        for j in range(i+1,n):

            R[i,j] = np.dot(Q[:,i], V[:,j])
            V[:,j] = V[:,j] - R[i,j] * Q[:,i]
```

donde el vector V es una copia de la matriz de entrada A.

Por dar un ejemplo consideremos que se desea factorizar la matriz

$$\begin{bmatrix} 1 & 2 & 2 \\ 2 & 0 & 2 \\ 0 & 1 & 2 \end{bmatrix}$$

tras aplicar la función GramSchmidt a la matriz A resulta en

$$A = QR = \begin{bmatrix} 0{,}4472136 & 0{,}78072006 & -0{,}43643578 \\ 0{,}89442719 & -0{,}39036003 & 0{,}21821789 \\ 0. & 0{,}48795004 & 0{,}87287156 \end{bmatrix} \begin{bmatrix} 2{,}23606798 & 0{,}89442719 & 2{,}68328157 \\ 0. & 2{,}04939015 & 1{,}75662013 \\ 0. & 0. & 1{,}30930734 \end{bmatrix}$$

que tras hacer la multiplicación matricial de la derecha se comprueba que es en efecto una factorización de A.

Ejercicio 2
Implementar el algoritmo que calcula el estimador de mínimos cuadrados de una regresión usando la descomposición QR.

Solución:

Consideremos el problema de mínimos cuadrados donde nos interesa encontrar $\hat{x}$ que minimice la relación en $b = Ax$ bajo la norma en L^2

$$\hat{x} = argmin_x ||b - Ax|| \tag{1}$$

Sabemos que la solución a (1) es

$$y = Ax$$

donde $Pb = y$ y $P = QQ*$ es el proyector ortogonal. Luego el estimador de minimos cuadrados satisface

$$\begin{aligned} QQ * b &= QR\hat{x}, \\ Q * b &= R\hat{x} \end{aligned} \tag{2}$$

por lo que resta resolver el sistema de ecuaciones, y como R es una matriz triangular superior, entonces por medio de *backward subsection* se obtiene el estimador de mínimos cuadrados.

Por dar un ejemplo, consideremos la matriz de diseño

$$A = \begin{bmatrix} 1 & 2 & 0 \\ 1 & 7 & 2 \\ 1 & 7 & 9 \\ 1 & 4 & 5 \end{bmatrix}$$

y vector respuesta

$$b = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix}$$

por tanto usando la función construida para hacer descomposión QR y los comandos

```
A = np.array([[1,2,0],[1,7,2],[1,7,9],[1,4,5]])

b = np.array([[1],[2],[3],[1]])

Q, R = GramSchmidtModified(A)

b_hat = Q.T  bx = backward(R,bhat)
```

Tenemos las siguientes matrices

$$QR = \begin{bmatrix} 0{,}5 & -0{,}70710678 & -0{,}21320072 \\ 0{,}5 & 0{,}47140452 & -0{,}71066905 \\ 0{,}5 & 0{,}47140452 & 0{,}56853524 \\ 0{,}5 & -0{,}23570226 & 0{,}35533453 \end{bmatrix} \begin{bmatrix} 2. & 10. & 8. \\ 0. & 4{,}24264069 & 4{,}00693843 \\ 0. & 0. & 5{,}47215172 \end{bmatrix}$$

Además se obtiene el resultado, que es el estimador de mínimos cuadrados

$$\hat{x} = \begin{bmatrix} 0{,}13961039 \\ 0{,}25974026 \\ 0{,}07792208 \end{bmatrix} \tag{3}$$

por tanto el modelo de regresión lineal es

$$b_i = x_1 + a_{i1}x_2 + a_{i3}x_3 \tag{4}$$

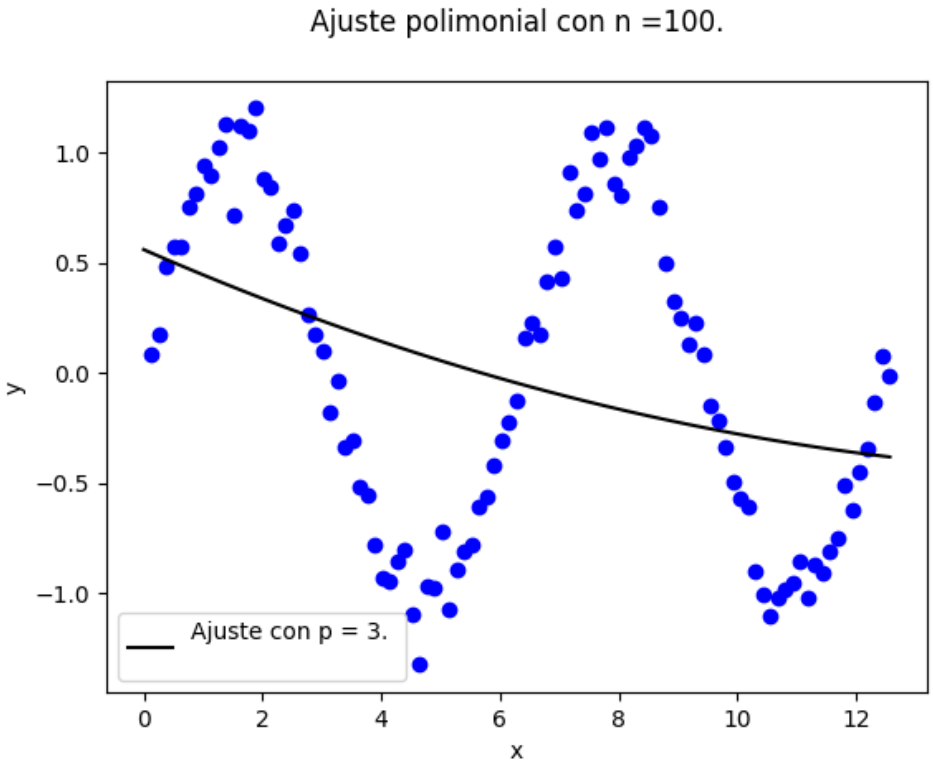
donde los x_i están dados en (3).

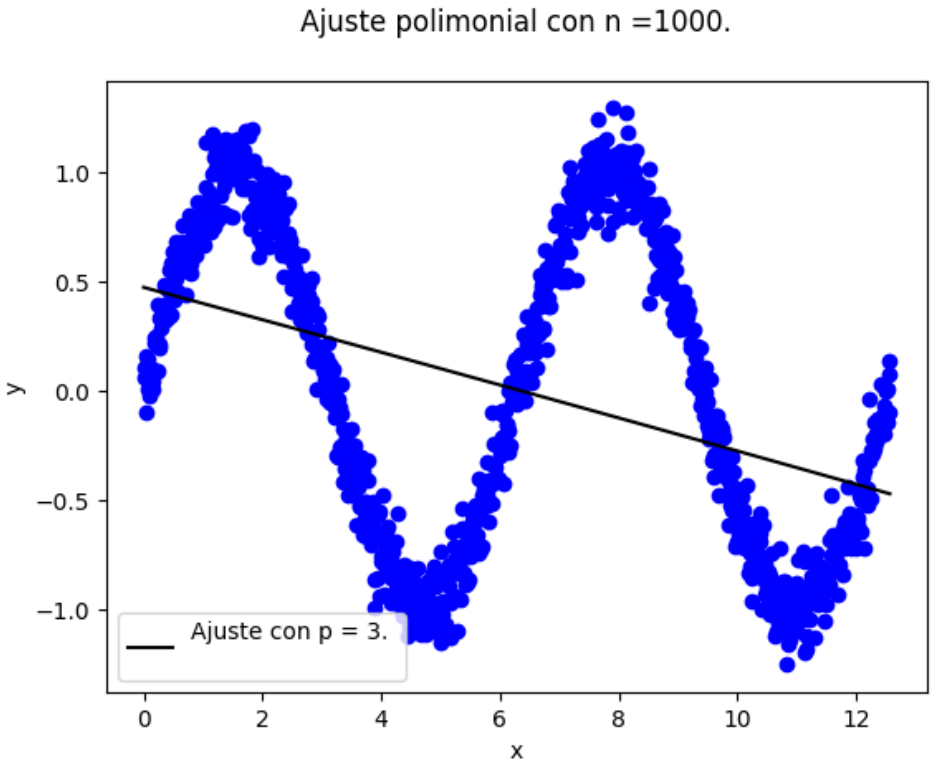
Ejercicio 3
Generar **Y** compuesto de $y_i = sen(x_i) + \epsilon_i$ donde $\epsilon_i \sim N(0, \sigma)$ con $\sigma = 0{,}11$ para $x_i = \frac{4\pi i}{n}$ para $i = 1, \cdots, n$.

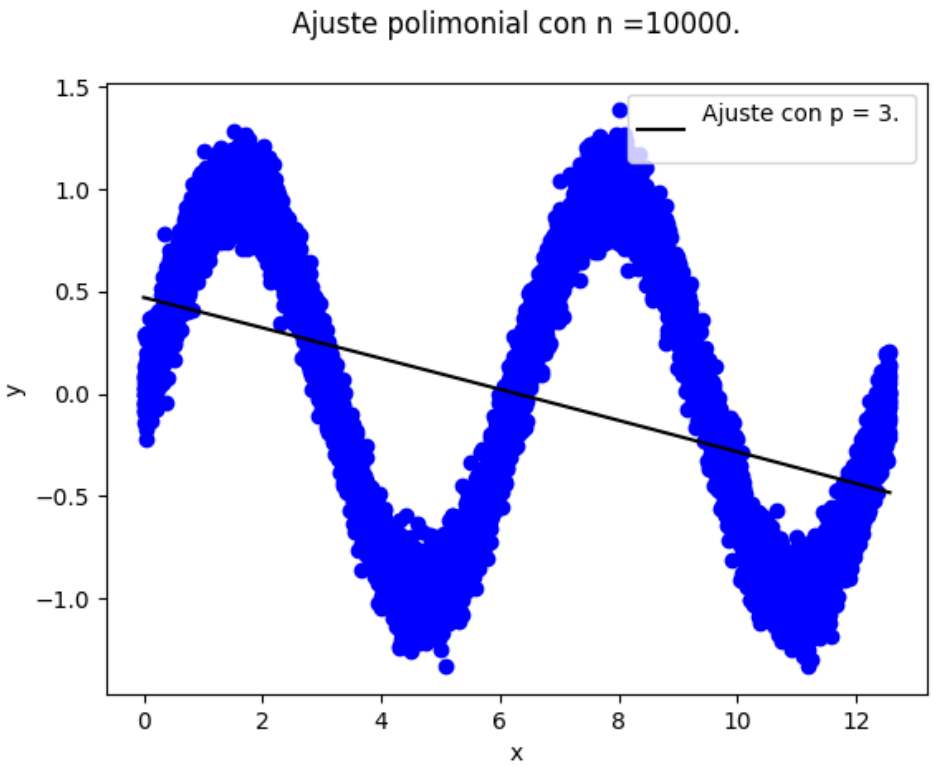
Hacer un ajuste de mínimos cuadrados a **Y**, con descomposición QR, ajustando un polinomio de grado $p - 1$.

1. Considerar los 12 casos: p = 3,4,6, 100 y n = 100, 1000, 10000.
2. Graficar el ajuste en cada caso.
3. Medir el tiempo de ejecución de su algoritmo, comparar con la descomposición QR de scipy y graficar los resultados.

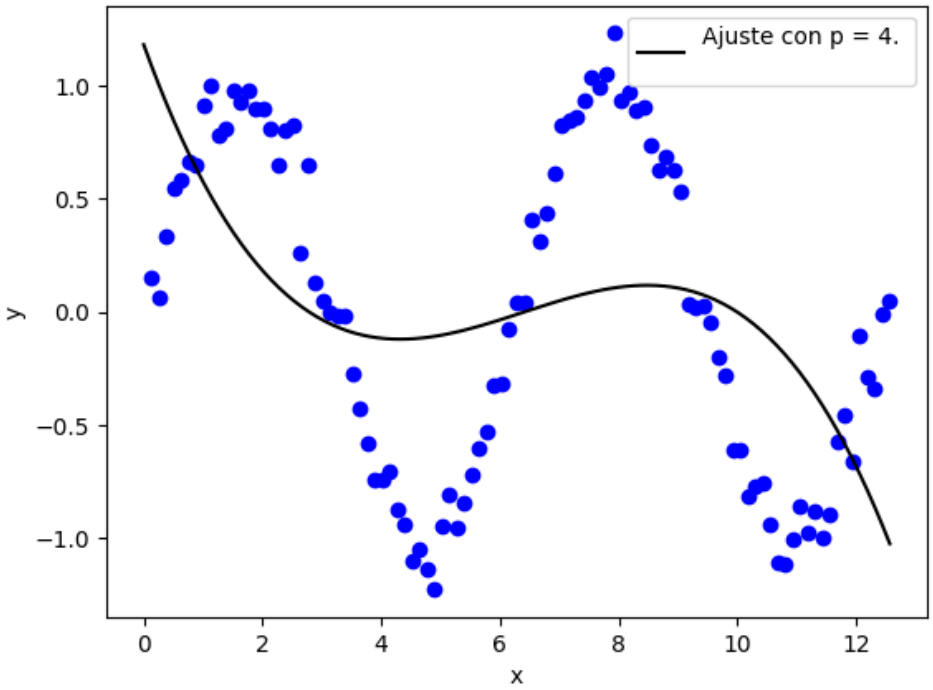
Solución:
Se construyo una función que recrea el experimento mencionado en el enunciado superior. Es decir, tomamos una regresión polinomial de grado $p - 1$. El regresor x es una particion del intervalo $[0, 4\pi]$ y la variable respuesta es una tranformación del regresor más el error que distribuye normal estandar. Se generaron los n puntos. Se uso regresión con factorización QR tal como se detallo en el ejercicio previo. Luego se graficaron los resultados de la regresión que se muestran a continuación.



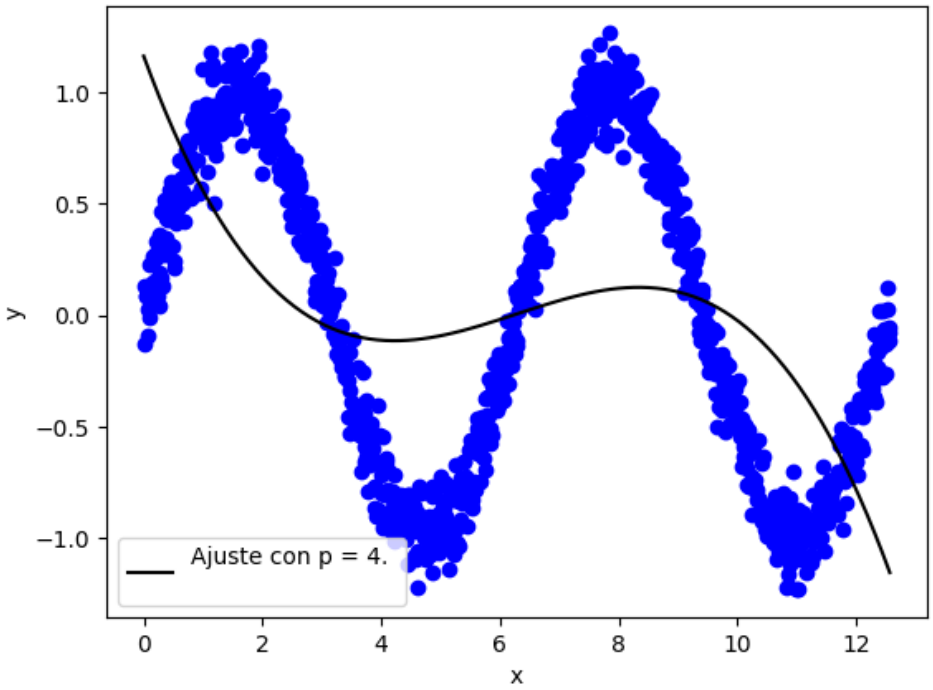

Ajuste polimonial con n =1000.
1.0
0.5
0.0
−0.5
−1.0
y
Ajuste con p = 3.
0
2
4
6
8
10
12
x

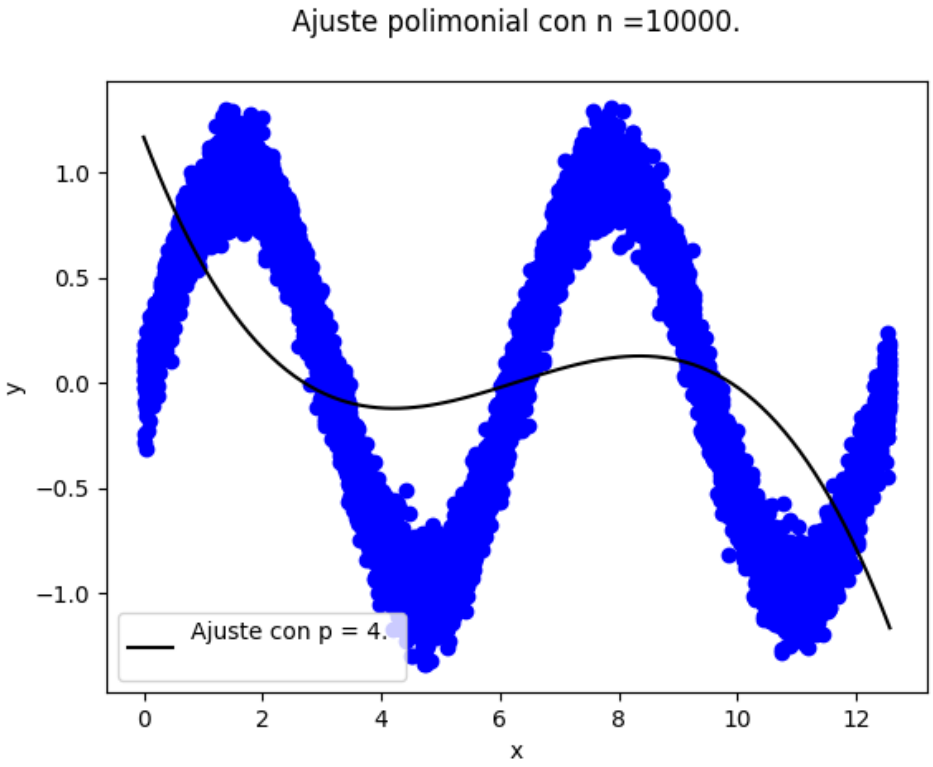

Ajuste polimonial con n =10000.
1.5
1.0
0.5
0.0
−0.5
−1.0
y
Ajuste con p = 3.
0
2
4
6
8
10
12
x

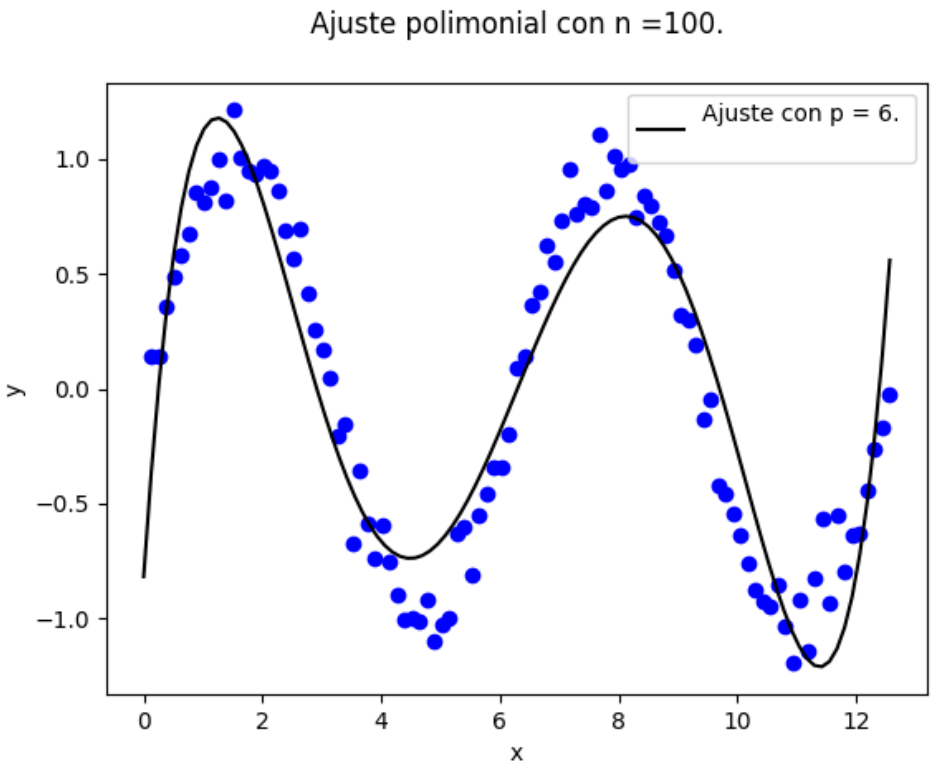
Ajuste polimonial con n =100.

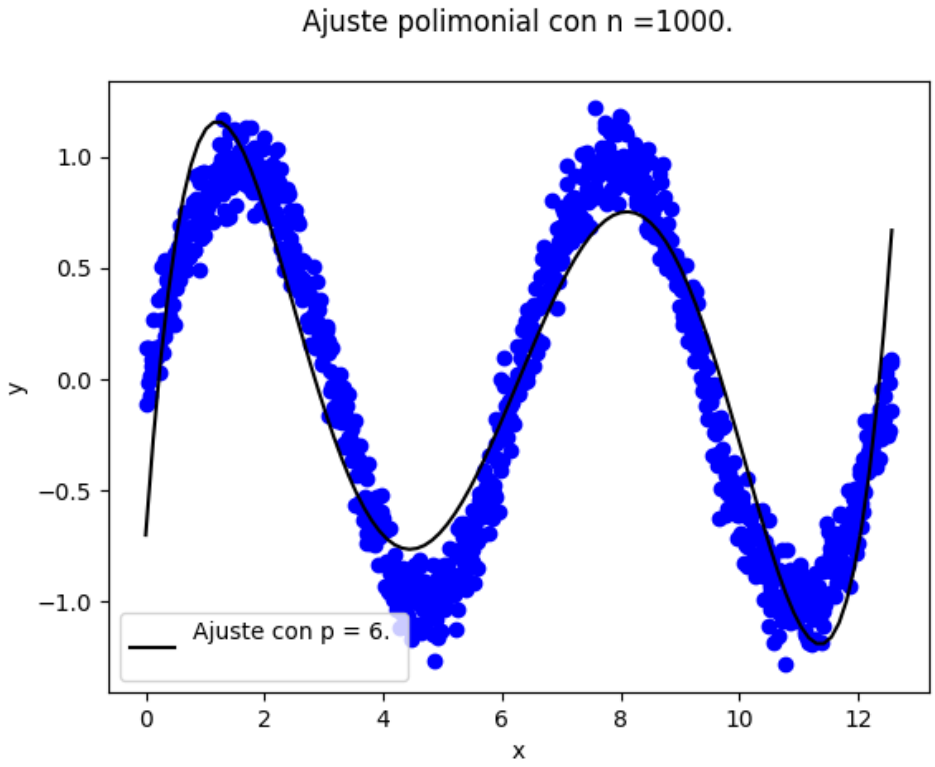


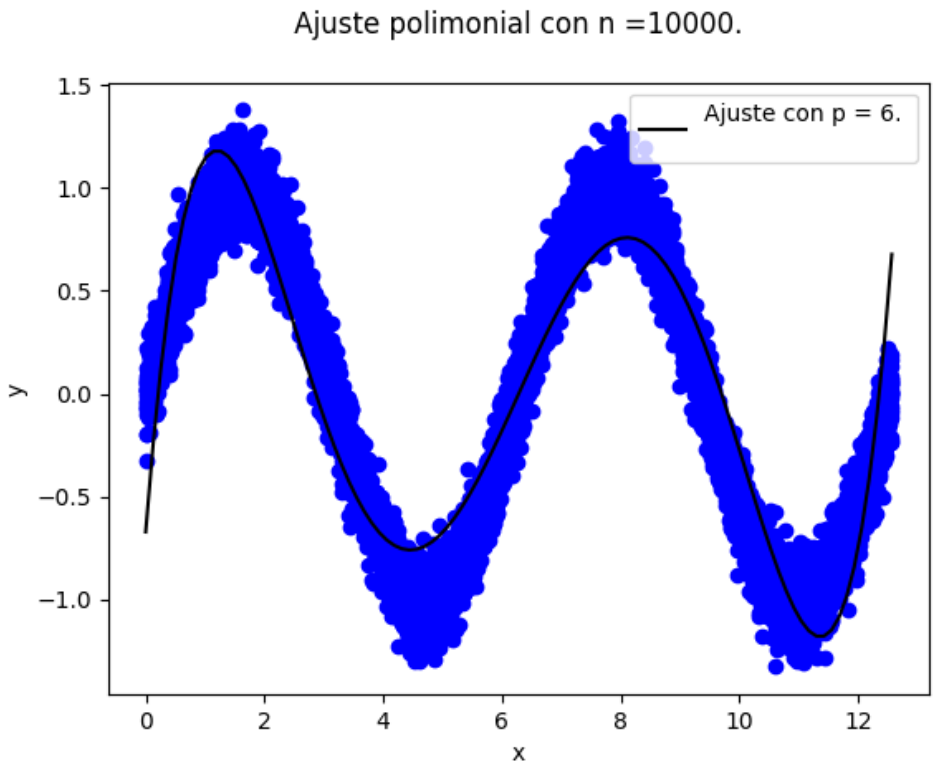
Ajuste polimonial con n =1000.



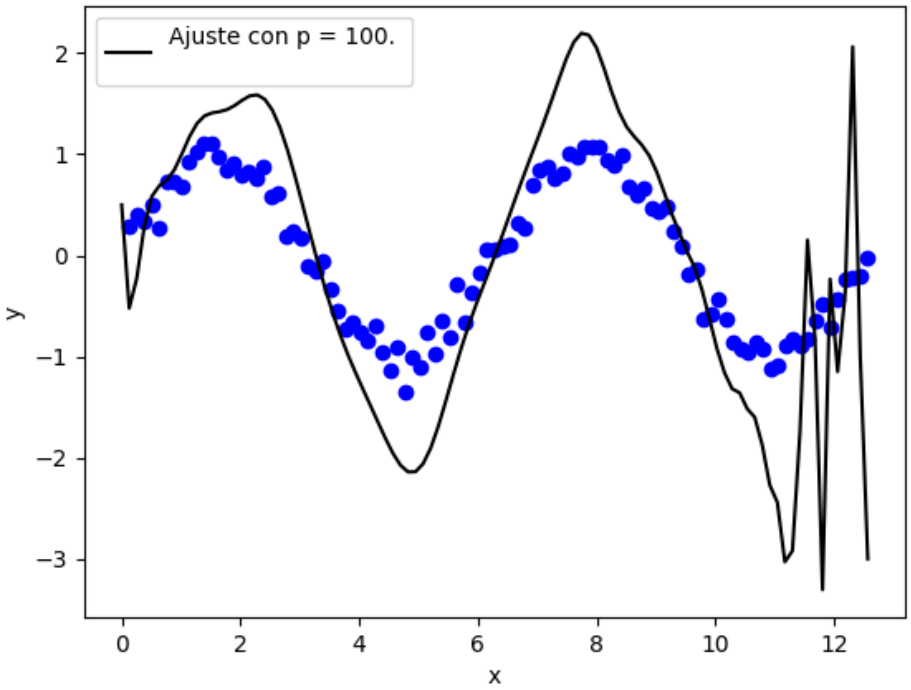

Ajuste polimonial con n =10000.
Ajuste con p = 4.
x
y


Ajuste polimonial con n =100.
Ajuste con p = 6.
x
y

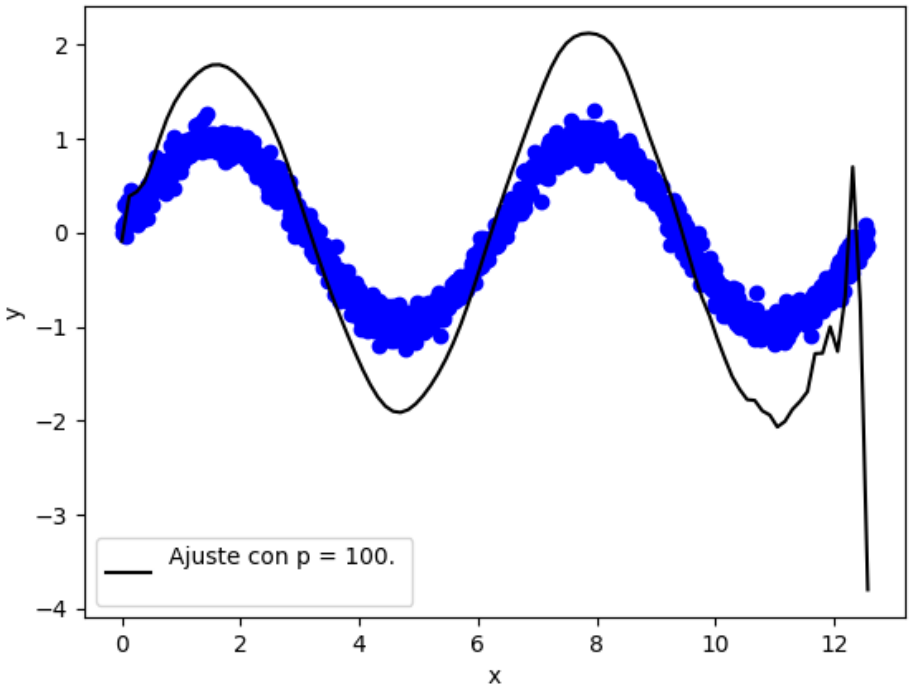

Ajuste polimonial con n =1000.
Ajuste con p = 6.
x
y

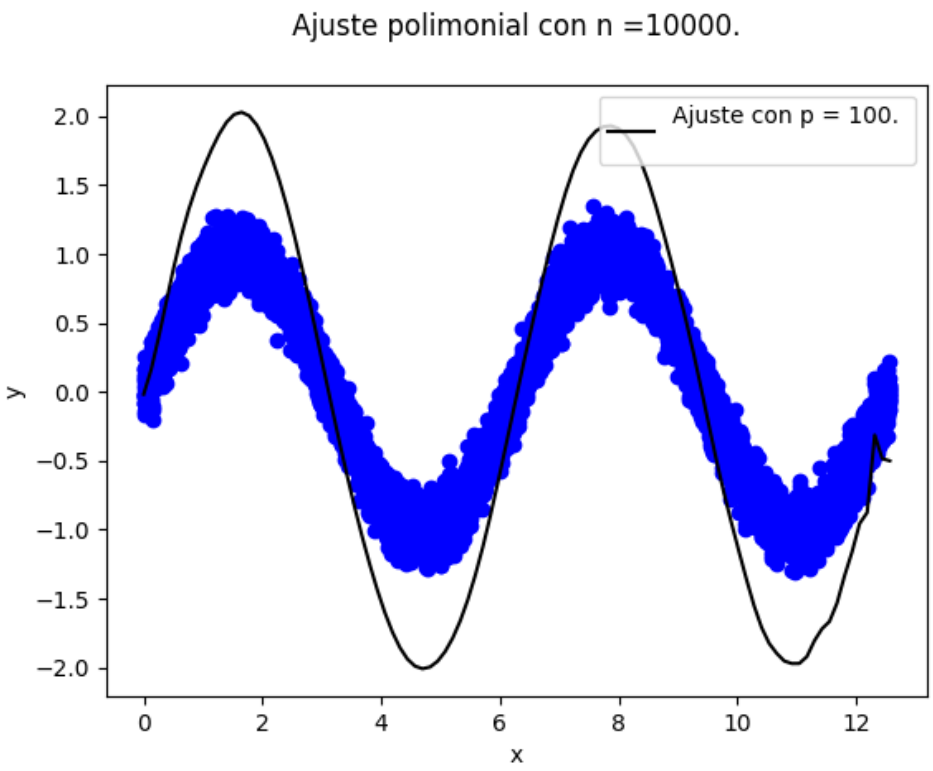

Ajuste polimonial con n =10000.
Ajuste con p = 6.
x
y

Ajuste polimonial con n =100.

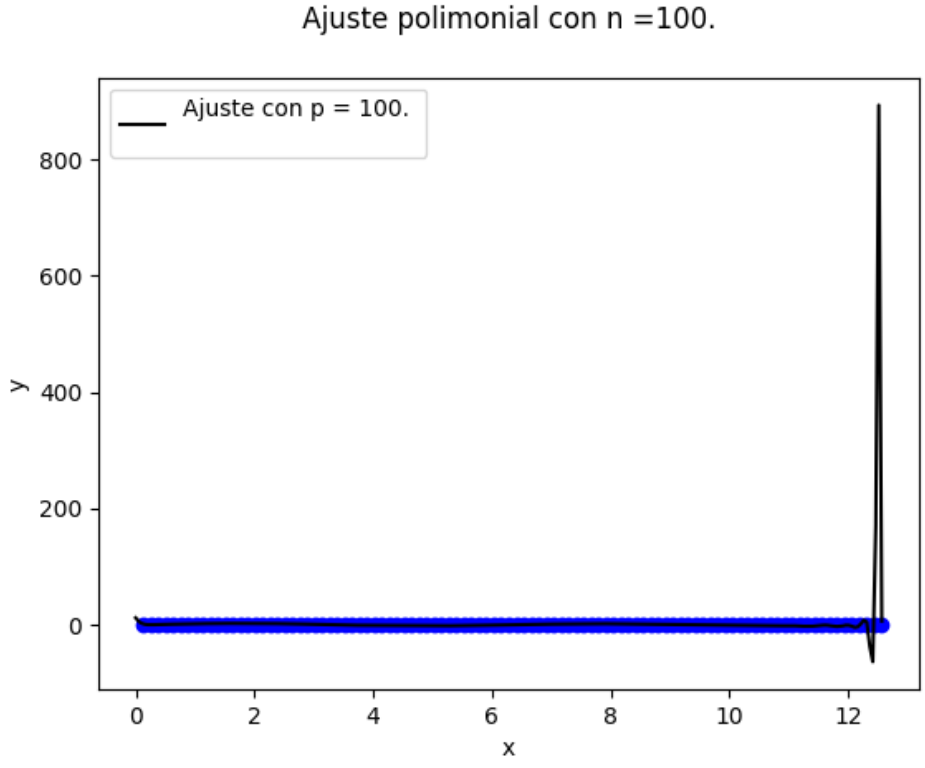


Ajuste polimonial con n =1000.





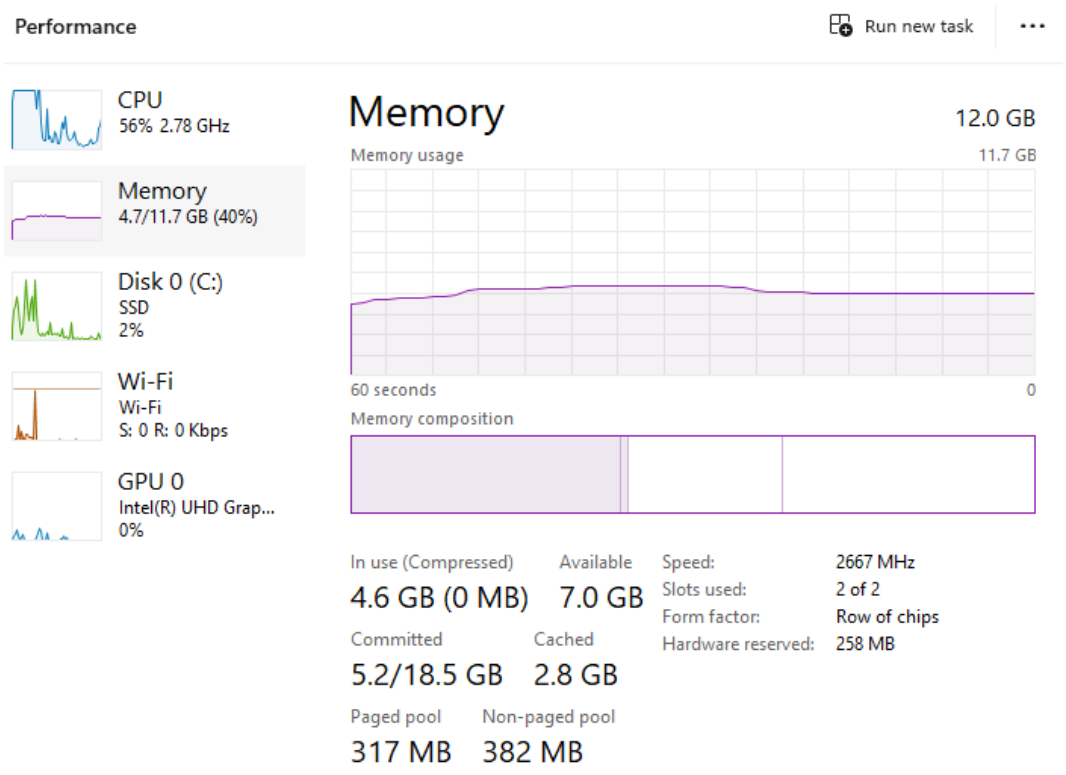
Observemos que las gráficas son bastante intuitivas a excepción de la gráfica con $n = p = 100$ pues el ajuste debería pasar por cada valor en la muestra ya que la matriz de diseño esta completamente determinada. Sin embargo este es un problema de la graficación, pues el dominio se partio con un *linspace* con solo 100 subdivisiones. Se puede ver en la siguiente figura que si aumentamos dicho número para mayor presición vemos que el ajuste pasa por los datos, aunque por defecto del ajuste lineal los extremos tienden a crecer desproporcionadamente.



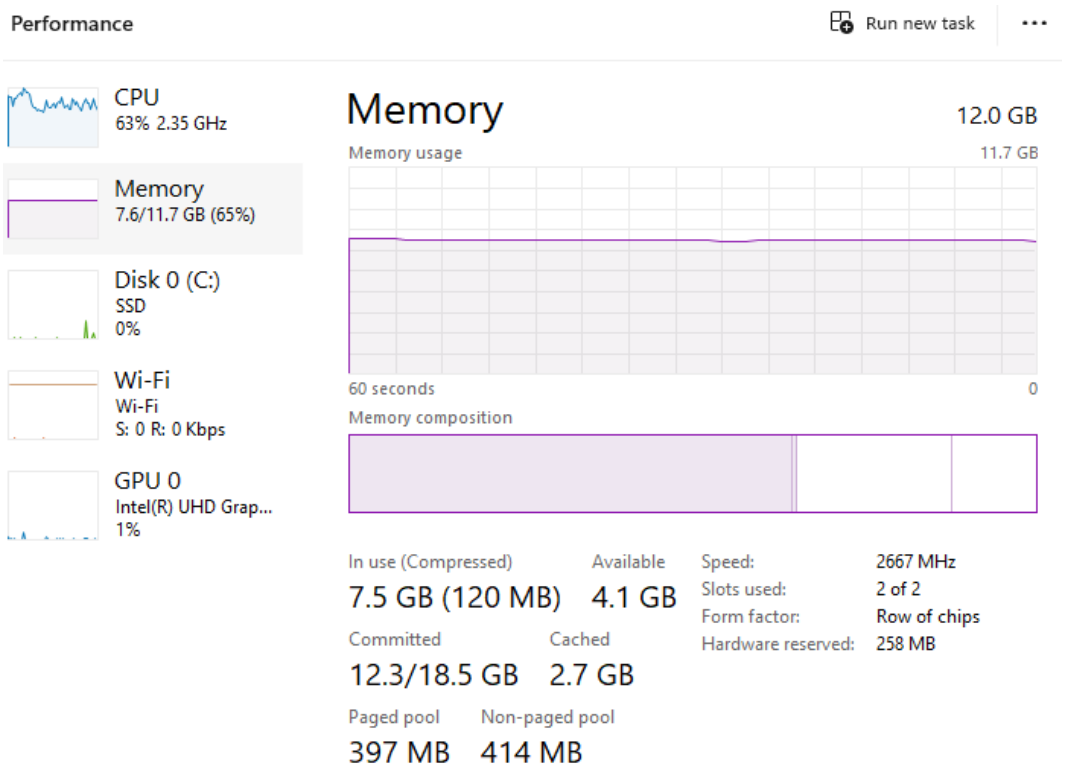
Ejercicio 4
Hacer $p = 0{,}1n$, o sea, diez veces más observaciones que coeficientes en la regresión, ¿ Cuál es la n máxima que puede generar su computadora?

Solución:

Vemos que el valor de n máximo es 50000. En un estado natural el comportamiento de la memoria esta cerca del 40 por cieto de uso, como lo vemos en la siguiente figura



En el caso de n cercano a 50000 la memoria se mantine estable pero el proceso no termina. Vemos los requerimientos en la siguiente figura.



Sin embargo en casos mayores se observa una creciente consumo de memoria y capacidad de CPU por lo que el proceso termina por fracturarse.