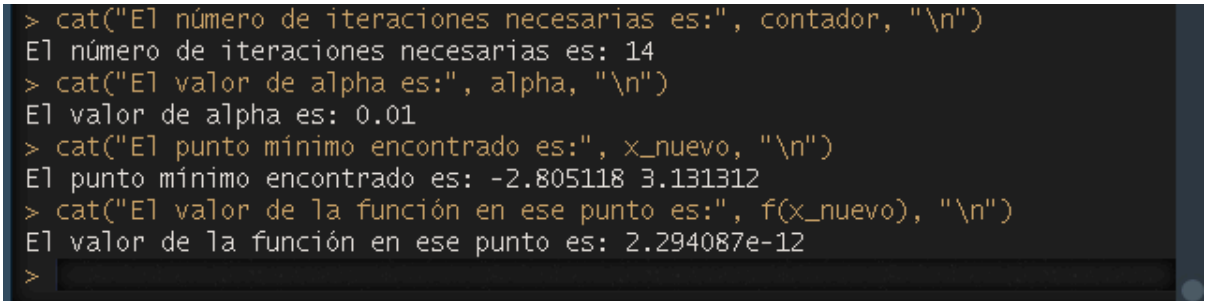
1 - La función de Himmelblau es un estándar para probar algoritmos de optimización, ya que posee cuatro mínimos locales idénticos. El objetivo es implementar y comparar el rendimiento del método de descenso de gradiente (GD) estándar con el descenso de gradiente con momento (GDM).Se debe analizar cómo la elección de los hiper parámetros (tasa de aprendizaje y momento) afecta la convergencia.
La función está definida como:

f(x,y) = (x^2 + y -11)^2 + (x + y^2 -7)^2

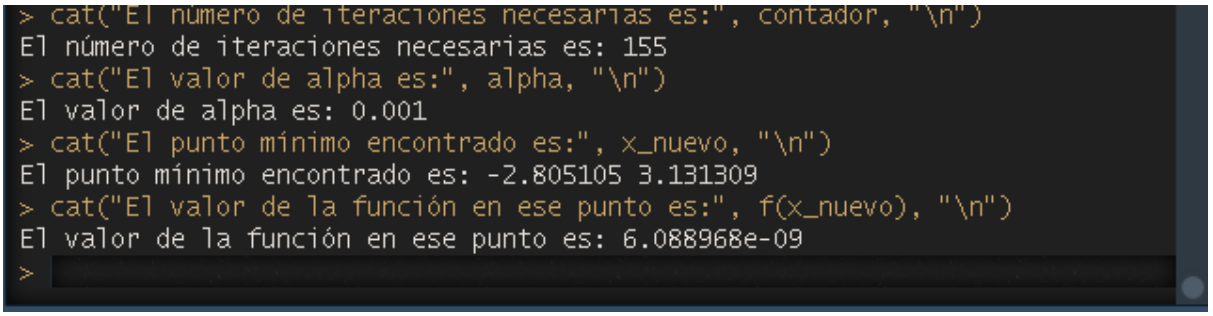
- Para GD, pruebe con tasas de aprendizaje α = [0.001,0.005,0.01].
- Para GDM, utilice una tasa de aprendizaje fija α = 0.005 y pruebe con coeficientes de momentos γ = [0.5,0.7,0.9].
- Utilice un máximo de 10000 iteraciones y una tolerancia de 1e-6 como criterios de detención

Soluciones:

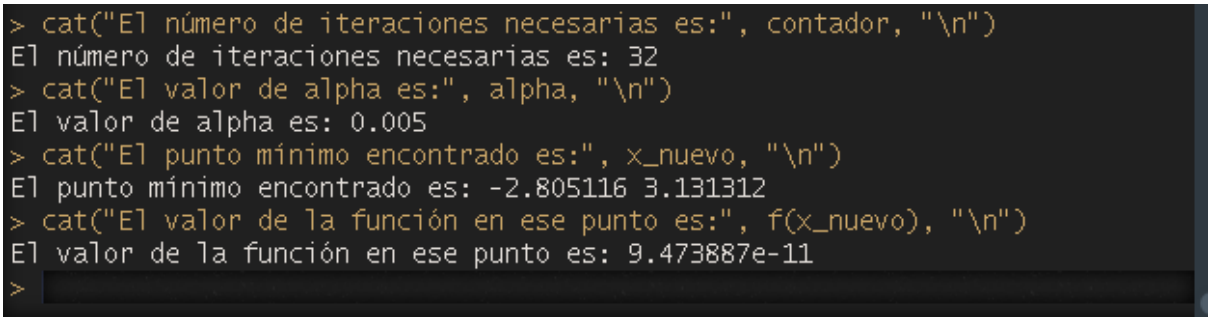
Para alpha = 0.01 gama = Null



Para alpha = 0.001 gama = Null



Para alpha = 0.005 gama = Null



Para alpha = 0.05 gama = 0.5

```
> cat("El número de iteraciones necesarias es:", contador, "\n")
El número de iteraciones necesarias es: 37
> cat("El valor de alpha es:", alpha, "\n")
El valor de alpha es: 0.005
> cat("el valor de Gama es: ",gama, "\n")
el valor de Gama es:  0.5
> cat("El punto mínimo encontrado es:", x_nuevo, "\n")
El punto mínimo encontrado es: -2.805119 3.131311
> cat("El valor de la función en ese punto es:", f(x_nuevo), "\n")
El valor de la función en ese punto es: 1.065151e-10
>
```

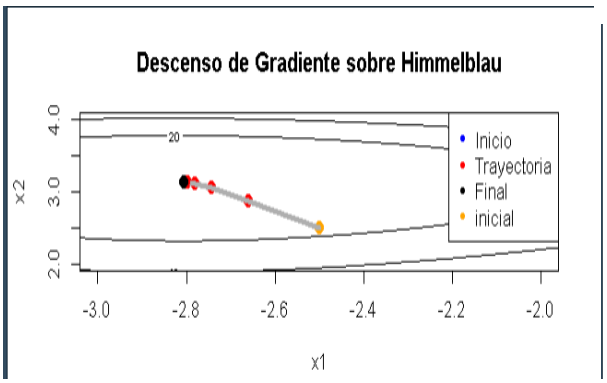
Para alpha = 0.05 gama = 0.7

```
> cat("El número de iteraciones necesarias es:", contador, "\n")
El número de iteraciones necesarias es: 69
> cat("El valor de alpha es:", alpha, "\n")
El valor de alpha es: 0.005
> cat("el valor de Gama es: ",gama, "\n")
el valor de Gama es:  0.7
> cat("El punto mínimo encontrado es:", x_nuevo, "\n")
El punto mínimo encontrado es: -2.805119 3.131315
> cat("El valor de la función en ese punto es:", f(x_nuevo), "\n")
El valor de la función en ese punto es: 2.854088e-10
>
```

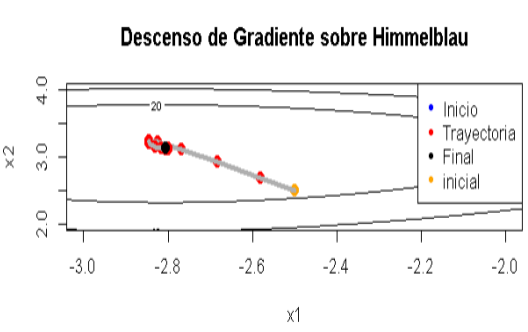
Para alpha = 0.05 gama = 0.9

```
> cat("El número de iteraciones necesarias es:", contador, "\n")
El número de iteraciones necesarias es: 224
> cat("El valor de alpha es:", alpha, "\n")
El valor de alpha es: 0.005
> cat("el valor de Gama es: ",gama, "\n")
el valor de Gama es:  0.9
> cat("El punto mínimo encontrado es:", x_nuevo, "\n")
El punto mínimo encontrado es: -2.805116 3.131316
> cat("El valor de la función en ese punto es:", f(x_nuevo), "\n")
El valor de la función en ese punto es: 6.831991e-10
>
```

Alpha = 0.01 y Gama = Null



Alpha = 0.005 y Gama = 0.5



GD o GDM	Gama	Alpha	Iteracion	Minimo X	Minimo Y	Valor funcion
GD	null	0,01	14	-2,805118	3,131312	1,09893E-11
GD	null	0,001	155	-2,805104	3,131309	6,87751E-09
GD	null	0,005	32	-2,805116	3,131312	1,50831E-10
GDM	0,5	0,005	37	-2,805119	3,131311	1,21593E-10
GDM	0,7	0,005	69	-2,80512	3,131315	3,60376E-10
GDM	0,9	0,005	224	-2,805115	3,131317	1,13538E-09

Esta función el mejor algoritmo es el de Alfa igual a 0,001 sin Gama, ya que converge en un mínimo más cercano al (-2.8 y 3.1), aunque su nivel de iteraciones es más alta su acercamiento es macho aunque solo por unos decimales muy pequeños (1X10^-6)

La razón por que la tasa de aprendizaje el momento no de mucha fue de ayuda en este caso es porque hay que encontrar un Alpha y un Gama exacto para que el nivel de exactitud suba y su número de iteración baje (normalmente entre más iteraciones major exactitud en el punto buscado) es casi un trabajo de tanteo, pero si solo se busca exactitud habría que aumentar la tolerancia y el número de iteraciones

2 - Una tasa de aprendizaje fija no siempre es óptima. Las estrategias de decaimiento pueden acelerar la convergencia y evitar oscilaciones cerca del mínimo. El objetivo, es implementar y comparar tres estrategias diferentes de decaimiento de la tasa de aprendizaje para minimizar la función de Rosenbrock:

f(x,y) = (1 - x)^2 +100*(y - x^2)^2

Modifique el algoritmo de descenso de gradiente para incorporar un planificador de tasa de aprendizaje.

- Implemente las siguientes tres estrategias de decaimiento para αk en la iteración k:

    1) Decaimiento por tiempo: αk = α0 / (1 + k^t) .
    2) 2. Decaimiento por pasos: αk = α0 * d^[k / s] , donde d es un factor de decaimiento y “s” es el tamaño del paso.
    3) 3. Decaimiento exponencial: αk = α0 *(e^(−k / s)

- Minimice la función de Rosenbrock partiendo desde (−1,−1).

    1) Utilice los siguientes parámetros: α0 = 0.002, max_iter = 10000.
    2) Para el decaimiento por tiempo y exponencial, use: “τ” = α0 10000.
    3) Para el decaimiento por pasos, use un factor d = 0.5 y un tamaño de paso “s” = 2000.
    4) Compare estos tres métodos con el descenso de gradiente con α fijo de 0.002

Solución

Alpha constante = 0.002

```
> cat("El número de iteraciones necesarias es:", contador, "\n")
El número de iteraciones necesarias es: 7778
> cat("El valor de alpha es:", alpha, "\n")
El valor de alpha es: 0.002
> cat("El punto mínimo encontrado es:", x_nuevo, "\n")
El punto mínimo encontrado es: 0.999442 0.998882
> cat("El valor de la función en ese punto es:", f(x), "\n")
El valor de la función en ese punto es: 3.124095e-07
>
```

```
12  #alfa
13  alpha <- 0.002
14  #valor inicial
15  x <- c(-1,-1)
16  #----------------
17  #constantes
18  tolerancia <- 1e-6
19  max_inter <- 10000
20  #----------------
```

Alpha por el tiempo

```
> cat("El número de iteraciones necesarias es:", contador, "\n")
El número de iteraciones necesarias es: 7782
> cat("El valor de alpha inicial:", alpha, "\n")
El valor de alpha inicial: 0.002
> cat("El valor de alpha final:", alpha_k, "\n")
El valor de alpha final: 0.001996008
> cat("El punto mínimo encontrado es:", x_nuevo, "\n")
El punto mínimo encontrado es: 0.999441 0.9988802
> cat("El valor de la función en ese punto es:", f(x_nuevo), "\n")
El valor de la función en ese punto es: 3.129362e-07
>
```

```
#alfa
alpha <- 0.002
alpha_k <- 0

#gama
gama <-alpha/10000

#valor inicial
x <- c(-1,-1)
```

```
for (i in 1:max_inter){

  #Cada vez que entra suma 1 para tener la cantidad de iteracion
  contador <- contador + 1
  #-----------------

  #calcular alfa k
  alpha_k <- (alpha)/(1 + contador*gama)
  #calculo del nuevo X
  x_nuevo <- x - alpha_k*f_gran(x)
  #-----------------
```

Alpha por los pasos

```
> cat("El número de iteraciones necesarias es:", contador, "\n")
El número de iteraciones necesarias es: 10000
> cat("El valor de alpha inicial:", alpha, "\n")
El valor de alpha inicial: 0.002
> cat("El valor de alpha final:", alpha_k, "\n")
El valor de alpha final: 0.001996008
> cat("El punto mínimo encontrado es:", x_nuevo, "\n")
El punto mínimo encontrado es: 0.9871138 0.9743416
> cat("El valor de la función en ese punto es:", f(x_nuevo), "\n")
El valor de la función en ese punto es: 0.0001663247
>
```

```
#alfa
alpha <- 0.002
#valor de d
d <- 0.5
#valro de s
s <- 2000
#valor inicial
x <- c(-1,-1)
#-----------------
```

```
for (i in 1:max_inter){
  #Cada vez que entra suma 1 para tener la cantidad de iteracion
  contador <- contador + 1
  #-----------------

  #calcular alfa k
  alpha_k <- (alpha)*(d^floor(i/s))
  #calculo del nuevo X
  x_nuevo <- x - alpha_k*f_gran(x)
  #-----------------
```

Alpha por el exponencial

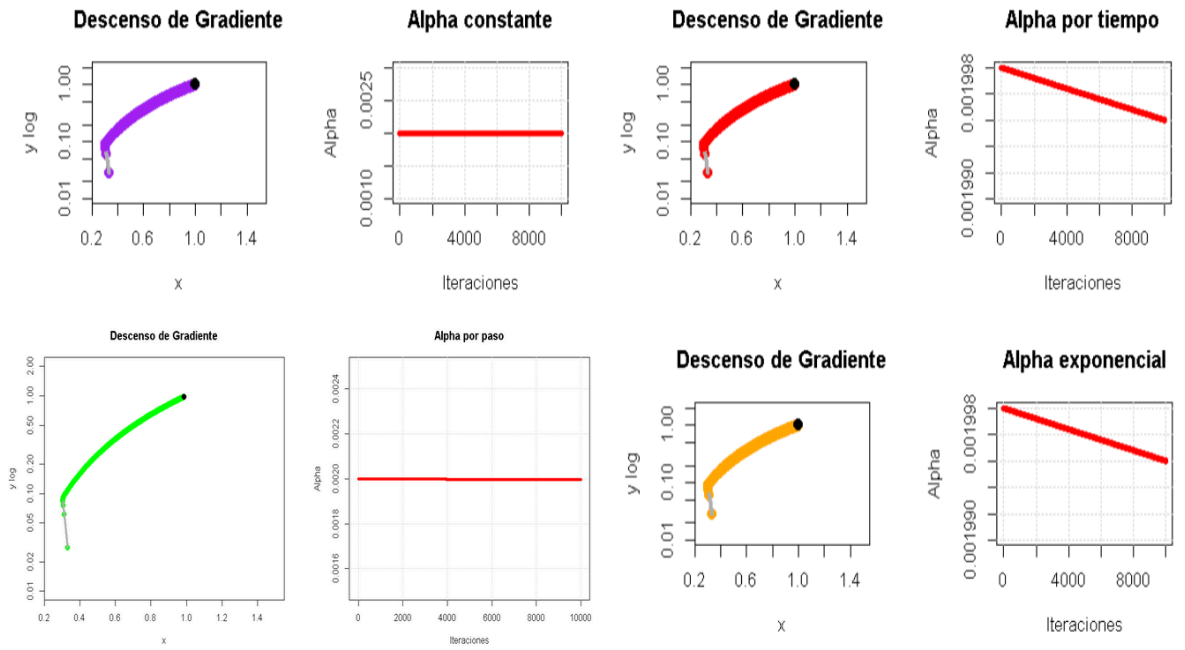
```
> cat("El número de iteraciones necesarias es:", contador, "\n")
El número de iteraciones necesarias es: 7782
> cat("El valor de alpha inicial:", alpha, "\n")
El valor de alpha inicial: 0.002
> cat("El valor de alpha final:", alpha_k, "\n")
El valor de alpha final: 0.001996008
> cat("El punto mínimo encontrado es:", x_nuevo, "\n")
El punto mínimo encontrado es: 0.999441 0.9988802
> cat("El valor de la función en ese punto es:", f(x_nuevo), "\n")
El valor de la función en ese punto es: 3.129377e-07
>
```

```
#alfa
alpha <- 0.002
#gama
gama <-alpha/10000
#valor inicial
x <- c(-1,-1)
#-----------------

#constantes
tolerancia <- 1e-6
max_inter <- 10000
#-----------------
```

```
for (i in 1:max_inter){
  #Cada vez que entra suma 1 para tener la cantidad de iteracion
  contador <- contador + 1
  #-----------------

  #calcular alfa k
  alpha_k <- alpha * exp(-i * gama)
  #calculo del nuevo X
  x_nuevo <- x - alpha_k*f_gran(x)
  #-----------------
```



Parametros	Alfa fijo	alfa por tiempo	alfa por pasos	alfa exponencial
alfa inical	0,002	0,002	0,002	0,002
alfa final	0,002	0,001996892	0,0000625	0,00199689
puntro minimo x	0,999442	0,999441	0,9685593	0,999441
punto minimo y	0,998882	0,9988802	0,9379788	0,9988802
valor funcion	3,12E-07	3,13E-07	9,90E-04	3,13E-07
iteraciones	7778	7782	10000	7782
diferencia del punto x	0,000558	0,000559	0,0314407	0,000559
diferencia del punto y	0,001118	0,0011198	0,0620212	0,0011198

En este caso el número más exacto y con el número más bajo de iteraciones a la vez (complicado de ver y de obtener) es cuando Alpha es fijo en 0,002. Esto puede ser porque la tolerancia que le dimos en muy baja, tal vez con una mayor se podría obtener el Apha por el tiempo sea más exacta ya que es la segunda más cercana al punto mínimo buscado (1,1).

Pero en el caso de optimización, es decir, que tenga menos iteración el mejor es el alfa fijo porque se demora menos y llega a un valor aceptable con su tolerancia dada

3 - El objetivo es implementar el algoritmo de regresión robusto utilizado el descenso de sub-gradiente para minimizar el MAE y comprobrlo con el modelo estándar de minimizar cuadrados en presencia de datos atípicos.

- Utilice los datos de frecuencia reloj y consumo energía del apunte de clases.
- Introduzca dos valores atípicos (outliers) en los datos para simular errores de medición, para la frecuencia de reloj [2.5,4.5] y para el consumo de energía [130,70].
- Defina la función objetivo MAE y su subgradiente.
- Implemente un algoritmo de descenso de gradiente que utilice el subgradiente MAE para encontrar los parámetros β óptimos.
- Ejecuta el algoritmo de regresión lineal original (basado en MSE) sobre el conjunto de datos con outliers.
- Ejecute el nuevo algoritmo de regresión robusta (basado en MAE) sobre el mismo conjunto de datos con outliers.
- Gráfico: Cree un único diagrama de dispersión que muestre los datos con los outliers. Sobre este gráfico, dibuje las dos rectas de regresión obtenidas:
    1) La recta del modelo de Mínimos Cuadrados (MSE).
    2) La recta del modelo de Error Absoluto Medio (MAE).
- Análisis: En una breve conclusión, explique cuál de las dos rectas se ajusta mejor a la tendencia general de los datos originales. ¿Por qué el modelo basado en MAE es considerado más “robusto” ante valores atípicos? ¿Cómo se relaciona esto con la penalización que cada función de objetivo impone a los errores grandes?

solución

```
22  #-------------------------------------------------
23  #DATOS ORIGINALES
24  # Freceuncia Reloj
25  X = np.array([4.29, 4.34, 2.72, 4.08, 3.60,
26  3.30, 3.84, 2.34, 3.64, 3.76,
27  3.14, 3.80, 4.34, 2.64, 3.16,
28  4.35, 4.45, 2.29, 3.19, 3.40,
29  4.26, 2.35, 4.47, 4.37, 2.21])
30
31  #consumo de Energia
32  Y = np.array([122, 130, 88, 121, 108,
33  102, 107, 75, 105, 119,
34  102, 112, 125, 79, 98,
35  127, 121, 73, 105, 101,
36  117, 78, 123, 119, 70])
37  #-------------------------------------------------
```

```python
#DATOS EXTRAS
# Freceuncia Reloj
x = np.array([4.29, 4.34, 2.72, 4.08, 3.60,
3.30, 3.84, 2.34, 3.64, 3.76,
3.14, 3.80, 4.34, 2.64, 3.16,
4.35, 4.45, 2.29, 3.19, 3.40,
4.26, 2.35, 4.47, 4.37, 2.21, 2.5,4.5])

#consumo de Energia
y = np.array([122, 130, 88, 121, 108,
102, 107, 75, 105, 119,
102, 112, 125, 79, 98,
127, 121, 73, 105, 101,
117, 78, 123, 119, 70, 130,70])
#-----------------------------------------------
```

Funciones para el cálculo de mínimos cuadrados

```python
def hteta(beta0: float, beta1: float, x: np.array):
    teta0=beta0
    teta1=beta1
    X=x
    resultado = (teta1*X)+teta0
    return resultado

def sumatori(beta0: float, beta1: float, y: np.array, x: np.array):
    teta0 = beta0
    teta1 = beta1
    Y = y
    X = x
    resultado = (hteta(teta0,teta1,X) - Y)
    return resultado

```

Cálculo de mínimos cuadrados

```python
def hteta(beta0: float, beta1: float, x: np.array):
    teta0=beta0
    teta1=beta1
    X=x
    resultado = (teta1*X)+teta0
    return resultado

def sumatori(beta0: float, beta1: float, y: np.array, x: np.array):
    teta0 = beta0
    teta1 = beta1
    Y = y
    X = x
    resultado = (hteta(teta0,teta1,X) - Y)
    return resultado

```

Gráfico de pendiente con datos Sin tocar (ORIGINAL)

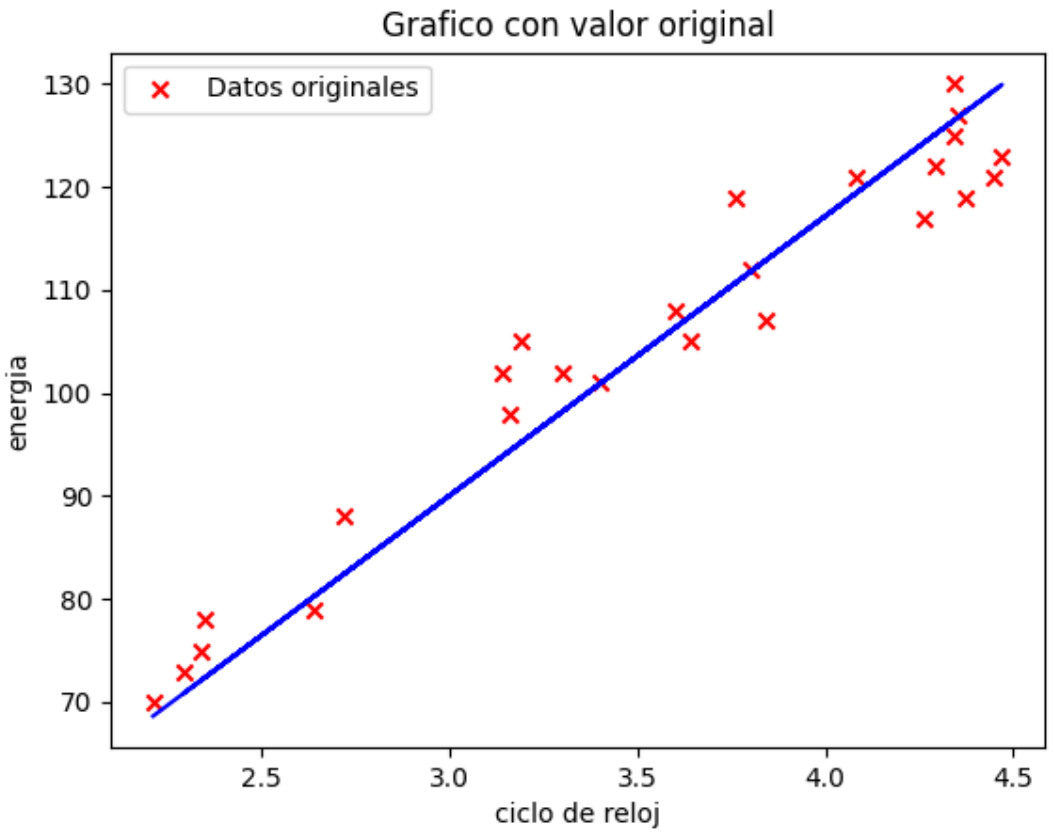
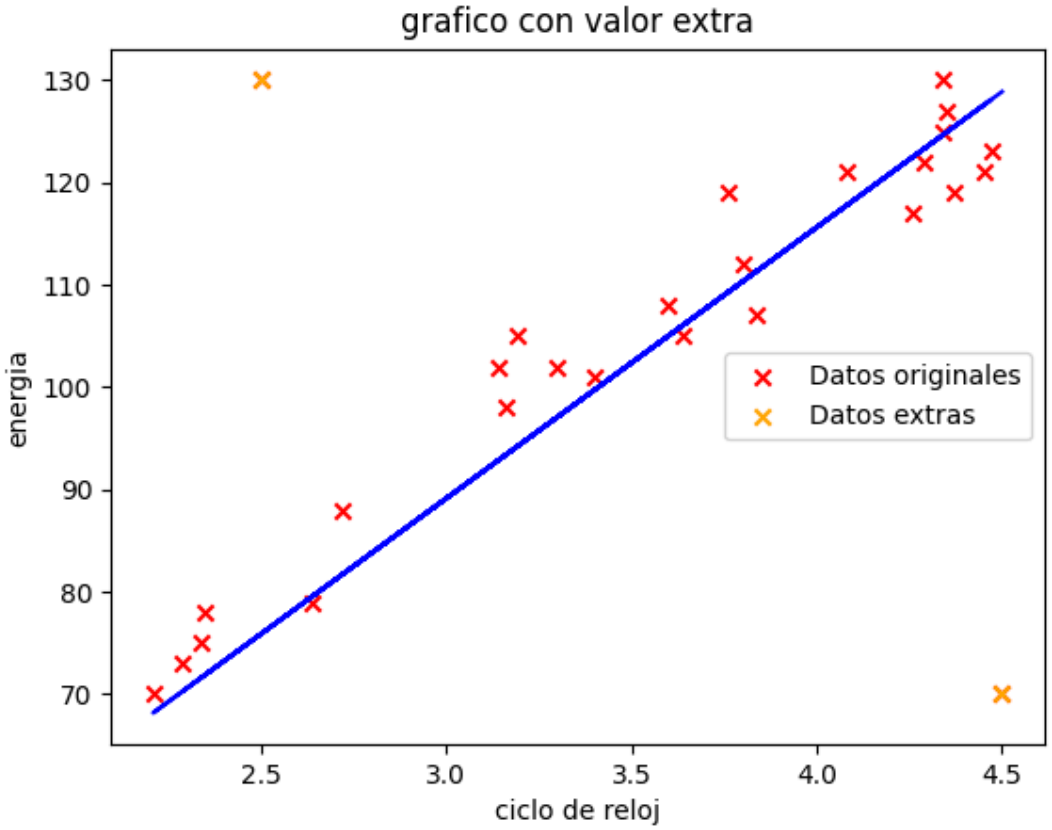


Gráfico de pendiente con datos nuevos (EXTRAS)



Comparacion de ambos gráficos