3. Ajustes y regresiones (lineal)

En esta guía vas a encontrar herramientas básicas para efectuar *estimaciones de parámetros* de tus modelos. Esto se hará mediante el ajuste (regresión) lineal de tus parámetros que minimize las diferencias entre las predicciones de tu modelo y tus observaciones.

Junto con las bibliotecas usuales se agrega la funcionalidad `curve_fit` de `scipy.optimize` que permitirá sistematizar los ajustes.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from scipy.optimize import curve_fit
```

3.1. Modelo y observaciones

A la hora de efectuar un análisis por regresión (ajuste) tenés que tener en claro qué objetivo te lleva a realizarlo.

Suponé que tenés un conjunto de n pares de valores experimentales (X_n, Y_n) tales que, por ejemplo, $X := L$ represente la longitud e $Y := T$ el periodo de un péndulo. También suponé que existe cierta relación funcional $Y(X)$, que puede ser conocida o no.

Si la relación funcional no es conocida, un objetivo posible es intentar estudiar qué *modelo* $Y(X)$ permite explicar la relación entre los resultados $Y_n(X_n)$ obtenidos. Esto se realiza, por ejemplo, proponiendo un modelo o varios que permitan explicar tus resultados.

Ahora pensá que el modelo ya fue propuesto y es conocido, aunque también depende de uno o varios *parámetros* desconocidos. Por ejemplo, suponé que tu modelo depende según cierta relación $Y(X; \alpha)$; donde tu parámetro $\alpha := g$ puede ser la aceleración de la gravedad (supongamos constante). Tu objetivo ahora es *estimar* el valor de α de tu modelo que mejor explique tus resultados experimentales.

Con este último objetivo en mente, la propuesta es que puedas realizar la estimación del parámetro de un modelo conocido mediante ajustes. Suponé entonces que vas a estimar la aceleración de la gravedad g, a partir de un modelo conocido que relaciona el periodo T de un péndulo con su longitud L, y que dicho modelo puede aproximarse como

$$T(L; g) = 2\pi\sqrt{\frac{L}{g}}\ .$$

Para empezar, tendrías que cargar tus datos experimentales (L_n, T_n), junto con sus correspondientes incertezas $(\Delta L_n, \Delta T_n)$. Una forma de hacerlo es cargando tus datos en `arrays` con nombres `L`, `dL`, `T` y `dT` como se ejemplifica en el siguiente bloque.

```
1 L = np.array([
```

```
L = np.array([
  2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 3.0,
  3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, 4.0
  ])

dL = np.array([
  0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
  0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1
  ])

T = np.array([
  2.82, 2.90, 2.91, 3.10, 3.12, 3.22, 3.25, 3.29, 3.32, 3.47, 3.53,
  3.54, 3.64, 3.71, 3.61, 3.73, 3.83, 3.87, 3.85, 3.91, 4.07
  ])

dT = np.array([
  0.01, 0.01, 0.06, 0.06, 0.01, 0.05, 0.01, 0.01, 0.04, 0.05, 0.06,
  0.01, 0.05, 0.06, 0.09, 0.02, 0.02, 0.01, 0.06, 0.05, 0.05
  ])
```

3.2. Regresión lineal

La forma más sencilla en la que podés encarar tu objetivo es mediante un *ajuste o regresión lineal*.

Notemos que la ecuación aproximada del periodo del péndulo, tal como fue escrita, no es lineal. Para salvar esta situación fijate que podés reescribir la misma como

$$T(X;\alpha) = \overbrace{\frac{2\pi}{\sqrt{g}}}^{\alpha} \underbrace{\sqrt{L}}_{X} = \alpha X \ ,$$

de modo que tu problema se reduce a estudiar la relación lineal entre T y X para poder estimar la pendiente α y, por lo tanto, estimar g. Asimismo, corresponderá propagar la incerteza de X como

$$\Delta X = \sqrt{\left(\frac{\partial X}{\partial L}\Delta L\right)^2} = \frac{\Delta L}{2\sqrt{L}}.$$

Queda como tarea aparte revisar que, aplicando $\log()$ a ambos miembros de la ecuación original del periodo del péndulo, es posible también linealizar en forma alternativa la relación. La única diferencia es que, en ese caso, ambas variables T y L cambiarán.

Bajo estas consideraciones, podés computar las nuevas variables X_n y ΔX_n con el siguiente bloque.

```
X = np.sqrt(L)
dX = dL/2/np.sqrt(L)
```

Habiendo organizado el conjunto de mediciones experimentales según (T_n, X_n) con sus respectivos errores, resta solo realizar la regresión lineal correspondiente a $T = \alpha X$ para estimar el valor de α.

Primero podés definir en tu código una función lineal `TX` que sólo tenga pendiente `a`; es decir, la ordenada al origen la consideramos, por hipótesis de nuestro modelo, vale cero.

Luego, llamando a la función `curve_fit` de `scipy.optimize` que realizará el ajuste por *cuadrados mínimos*, vas a poder estimar el parámetro `a` (α) y su incerteza estadística `da` ($\Delta\alpha$).

A continuación podés ejecutar esta regresión lineal y verificar los valores obtenidos para α y $\Delta\alpha$.

```
def TX(X, a):
  return a*X

popt, pcov = curve_fit(TX, X, T)

a = popt[0]
da = np.sqrt(pcov[0,0])

print(a)
print(da)
```

Haciendo uso de tu estimación de α vas a poder representar gráficamente la curva $T(X;\alpha)$ correspondiente a ese ajuste. De este modo, vas a poder contrastar este ajuste con los valores empíricos $T_n(X_n)$ y sus correspondientes incertezas.

Siguiendo esta estrategia, en el bloque a continuación podés graficar $(X_n \pm \Delta X_n, T_n \pm \Delta T_n)$ en contraste con $T(X;\alpha)$.

```
x_min = 1.3
x_max = 2.1

axis = np.linspace(x_min, x_max, 100)

plt.scatter(X, T)
plt.errorbar(X, T, xerr=dX, yerr=dT, linestyle="None")
plt.plot(axis, TX(axis, *popt), 'r-')
plt.axis([x_min, x_max, TX(x_min, *popt), TX(x_max, *popt)])
plt.title('Estimación de $\\alpha$ por regresión lineal')
plt.xlabel('Raíz cuadrada de la longitud del péndulo (m$^{1/2}$)')
plt.ylabel('Periodo de oscilación (s)')
plt.show()
```

Por último, es fundamental que no olvides el objetivo primero que te impulsó realizar esta regresión: estimar el valor de g. Para ello, solo basta invertir la relación $g(\alpha)$ y propagar su incerteza de modo tal que

$$g = \frac{4\pi^2}{\alpha^2} \ , \quad \Delta g = \frac{8\pi^2}{\alpha^3}\Delta\alpha \ ,$$

para que, finalmente, puedas resolver la estimación deseada de $g \pm \Delta g$ con el siguiente bloque.

```
g = 4*np.pi**2/popt[0]**2
dg = 8*np.pi**2*da/a**3

print(g)
print(dg)
```