6. Ajustes no-lineales

En esta guía vas a encontrar algunas herramientas básicas para el análsis de de datos a partir de ajustes no lineales.

Se utilizarán principalmente las siguientes bibliotecas:

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

6.1. Oscilación armónica simple

En la guía anterior vimos el ejemplo de *oscilador armónico simple*, cuya frecuencia y amplitud es constante. Sin embargo, las mediciones realizadas en laboratorio no se corresponden a la perfección con este modelo. La incorporación al modelo de oscilador de un término de fricción (como una fuerza proporcional a la velocidad) permite complejizar el análisis y dar cuenta de parte de las fenomenología que no se podíá explicar. Llamaremos a esto *oscilador armónico amortiguado*.

Puede verificarse que, en general, un sistema oscilatorio se ve amortiguado debido a la interacción viscosa con su medio. Este tipo de oscilación armónica amortiguada se puede modelar como

$$m\ddot{x} = -kx - b\dot{x},$$

donde b es coeficiente que mide el amortiguamiento debido a la viscosidad.

Para un caso subamortiguado, tal que $b^2 < 4km$, puede verificarse que la posición a todo tiempo es

$$x(t) = Ae^{-\lambda t}\cos(\omega t + \varphi)$$

donde $\lambda \equiv b/2m$ y la frecuencia

$$\omega \equiv \sqrt{\omega_0^2 + \lambda^2}$$

depende de la frecuencia natural $\omega_0 = \sqrt{\frac{k}{m}}$ del sistema sin amortiguamiento; es decir, para $b \equiv 0$.

Al igual que fuera realizado antes, la aceleración se puede representar como

$$a(t) := \ddot{x}(t) = Be^{-\lambda t}\cos(\omega t + \varphi),$$

siendo B una amplitud que corresponda a la aceleración.

Partiendo de la serie de datos adquiridos previmente, intentá realizar un ajuste no lineal sobre los datos para inferir los parámetros relevantes que dominan la dinámica del sistema según el modelo propuesto. Tené en cuenta que ahora tu modelo va a depender de un parámetro adicional λ, el cual debés contemplar en la definición de tu función.

No te olvides que la frecunecia angular ω, la frecuencia f y el periodo T cumplen la relación:

$$f \equiv \frac{\omega}{2\pi} = \frac{1}{T}$$

.

Para empezar con el análisis de tus datos, considerá que vas a estudiar $a(t)$: la aceleración del sistema como función del tiempo. Para ello, adquiriste una serie de n datos (t_n, a_n). A continuación generaremos datos de ejemplos, sólo para continuar con el posterior análisis.

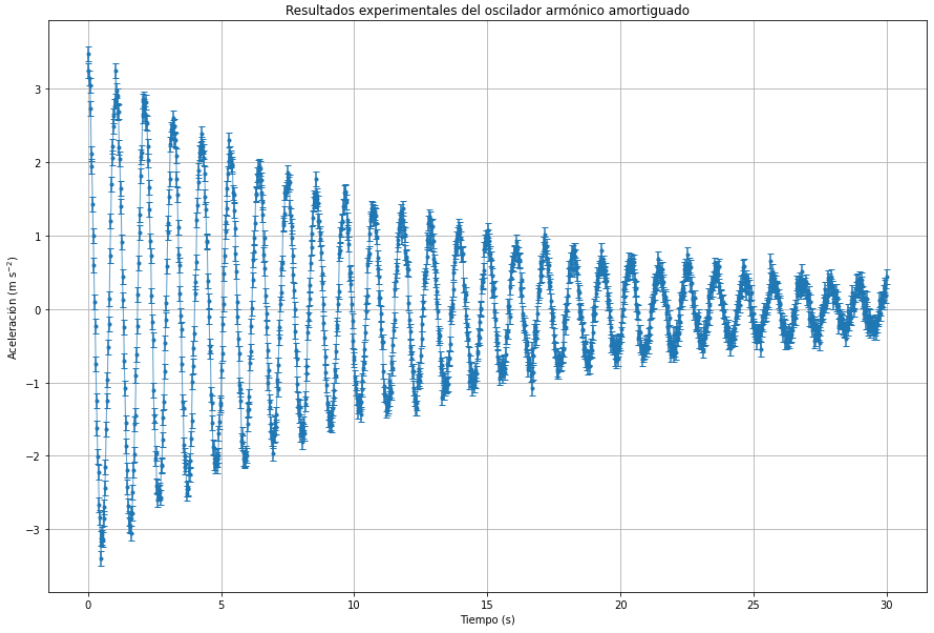
```
numero_de_datos_adquiridos= 1234

# Parámetros inventados para fabricar los datos de ejemplo
omega_0 = 5.853 # Hz
lamb    = 0.083 # Hz
phi     = 0.12  # Rad
A_0     = 3.32  # Expresado en m/s^2

# vector de tiempos
t = np.linspace(0,30,numero_de_datos_adquiridos)

# aceleraciones "medidas"
```

```
a = A_0 * np.exp(-lamb * t ) * np.cos( omega_0*t + phi )

# Sumamos ruido gaussiano a los datos
np.random.seed(0)
a += np.random.normal(size=numero_de_datos_adquiridos)/10

# Intervalos de incerteza asociados a las mediciones de aceleración (ej: error del instrumento)
err = np.ones(numero_de_datos_adquiridos)*0.1
```

Además, si correspondiese, no olvides registrar también las incertezas de tus mediciones.

Hecho esto, podés graficar $a_n(t_n)$ para así previsualizar tus datos y verificar que obtenés el comportamiento esperado del sistema.

```
plt.figure(figsize=(15, 10))
plt.plot(t, a, '-', lw=1 , alpha=0.7 , color='C0')
plt.errorbar(t, a, yerr=err, fmt='.', capsize=3  , color='C0')

plt.title('Resultados experimentales del oscilador armónico amortiguado')
plt.xlabel('Tiempo (s)')
plt.ylabel('Aceleración (m s$^{-2}$)')
plt.grid()
plt.show()
```



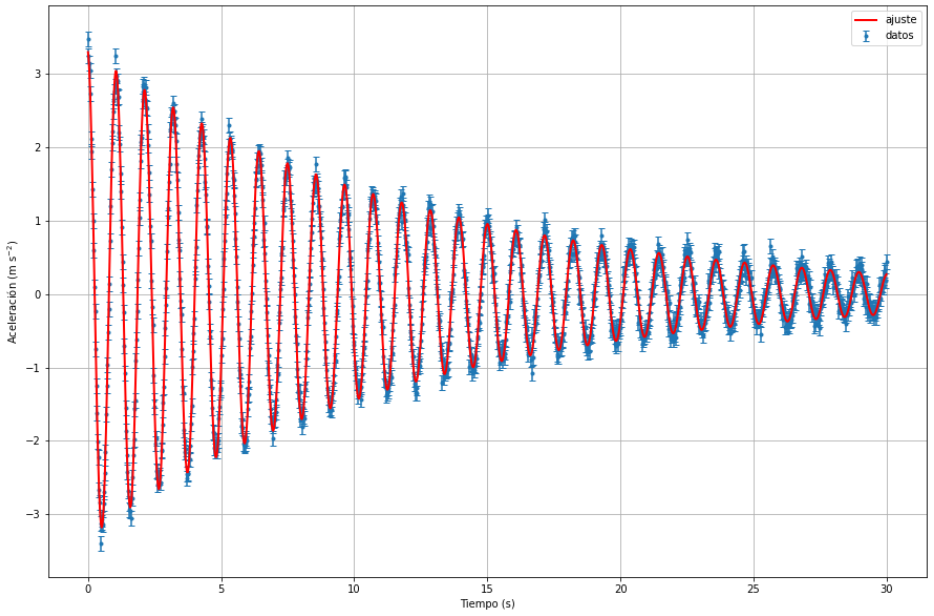

```
# Esto tiene formato de código
```

En la guía pasada vimos una serie de ejemplos para extraer de los datos los tiempos de cruce por cero (y de allí el periodo) y los picos de cada oscilación (y de allí la amplitud en cada uno de esos tiempos). Esa información nos permite tener una estimación de los parámetros del modelo.

Se puede realizar un análisis del modelo completo realizando un ajuste no lineal a los datos medidos. Como se vio en clase, los ajustes no lineales dependen fuertemente de hacer una elección correcta de los parámetros iniciales.

```

# Función que describe el modelo.
```

```
def modelo(t,A,w,lamb,phi):
  return A*np.exp(-lamb*t)* np.cos(w*t+phi)

# parámetros iniciales:    A_0 , lambda , omega , phi
parametros_iniciales   = [  5  ,    1   ,   1   , 0   ]

popt, pcov = curve_fit( modelo , t, a, p0 = parametros_iniciales , sigma=err )

A    , w    , lamb    , phi      = popt
A_err, w_err, lamb_err, phi_err  = np.sqrt(  np.diag(pcov) )

a_modelo = modelo( t , A  , w  , lamb  , phi)

plt.figure(figsize=(15, 10))

# plt.plot( t, a, '.', label='datos')
plt.errorbar(t, a, yerr=err, fmt='.', capsize=3 , label='datos' )

plt.plot( t, a_modelo , 'r-' , linewidth=2, label='ajuste' , zorder=3 )

# zorder es para indicar que curva se dibuja sobre otra

plt.xlabel('Tiempo (s)')
plt.ylabel('Aceleración (m s$^{-2}$)')
plt.legend()

plt.grid()
plt.show()

print(f'Parámetro A   : ({round(A,2)   } ± {round(A_err,2)   }) m/s^2')
print(f'Parámetro w   : ({round(w,4)   } ± {round(w_err,4)   }) Hz')
print(f'Parámetro lamb: ({round(lamb,4)} ± {round(lamb_err,4)}) s^(-1)')
print(f'Parámetro phi : ({round(phi,3) } ± {round(phi_err,3) }) ')


```




```
Parámetro A   : (-3.32 ± 0.01) m/s^2
Parámetro w   : (-5.853 ± 0.0005) Hz
Parámetro lamb: (0.083 ± 0.0005) s^(-1)
Parámetro phi : (3.022 ± 0.004)
```

Un concepto importante a tener en cuenta con los ajustes no lineales es que hay una fuerte dependencia de los parámetros iniciales. El algoritmo de optimización que permite hallar los parámetros busca el mínimo de los cuadrados de los residuos. Si el modelo es lineal, existe un sólo mínimo. Pero en los modelos no lineales pueden existir mínimos locales que no son el resultado que esperamos obtener. Acá un ejemplo de cómo pueden resultar mal las cosas si se eligen MAL los parámetros iniciales:

```
# Función que describe el modelo.
def modelo(t,A,w,lamb,phi):
  return A*np.exp(-lamb*t)* np.cos(w*t+phi)

# parámetros iniciales:    A_0 , lambda , omega , phi
# La frecuencia es muy lejana a la de los datos
parametros_iniciales   = [  0.1  ,   1   ,   50   , 0   ]

popt, pcov = curve_fit( modelo , t, a, p0 = parametros_iniciales )

A    , w    , lamb    , phi      = popt
A_err, w_err, lamb_err, phi_err  = np.sqrt(  np.diag(pcov) )

a_modelo = modelo( t , A  , w  , lamb  , phi)

plt.figure(figsize=(15, 10))

plt.plot( t, a, '.', label='datos')
plt.plot( t, a_modelo , 'r-' , label='ajuste')
plt.xlabel('Tiempo (s)')
plt.ylabel('Aceleración (m s$^{-2}$)')
plt.legend()

plt.grid()
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/scipy/optimize/minpack.py:808: OptimizeWarning: Covariance of the parameters could not
  category=OptimizeWarning)
```