Aproximación

Importamos los módulos `matplotlib.pyplot`, `numpy` y las funciones de `numpy` para polinomios.

```
import numpy as np
import matplotlib.pyplot as plt
import numpy.polynomial.polynomial as pol
```

Precisamos el formato en que queremos imprimir

```
np.set_printoptions(precision = 2)   # solo dos decimales
np.set_printoptions(suppress = True) # no usar notación exponencial
```

Aproximación polinómica discreta

Supongamos que buscamos un polinomio de grado 2 que aproxime a la función $f(x) = \text{sen}(x)$ de la cual disponemos valores para 5 puntos

$$x_1 = 0, x_2 = 0.5, x_3 = 1, x_4 = 1.5, x_5 = 2$$

El polinomio sería

$$P_2(x) = a_0 + a_1 x + a_2 x^2$$

Y tendremos tres incógnitas a_0, a_1, a_2,

Las ecuaciones serían

$$\begin{aligned} P_2(x_1) = y_1 \qquad & a_0 + a_1 x_1 + a_2 x_1^2 = y_1 \\ P_2(x_2) = y_2 \qquad & a_0 + a_1 x_2 + a_2 x_2^2 = y_2 \\ P_2(x_3) = y_3 \qquad & a_0 + a_1 x_3 + a_2 x_3^2 = y_3 \\ P_2(x_4) = y_4 \qquad & a_0 + a_1 x_4 + a_2 x_4^2 = y_4 \\ P_2(x_5) = y_5 \qquad & a_0 + a_1 x_5 + a_2 x_5^2 = y_5 \end{aligned}$$

Y el sistema a resolver es:

$$\begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ 1 & x_4 & x_4^2 \\ 1 & x_5 & x_5^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} \qquad (1)$$

es decir

$$Vp = y \qquad (1)$$

Tenemos más ecuaciones que incógnitas y el sistema no tiene solución. Pero si multiplicamos los dos miembros por la traspuesta de la matriz de coeficientes:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 & x_5 \\ x_1^2 & x_2^2 & x_3^2 & x_4^2 & x_5^2 \end{pmatrix} \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ 1 & x_4 & x_4^2 \\ 1 & x_5 & x_5^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 & x_5 \\ x_1^2 & x_2^2 & x_3^2 & x_4^2 & x_5^2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix}$$

Y ahora tenemos un sistema determinado de 3 ecuaciones con 3 incógnitas:

$$\begin{pmatrix} n & \sum_{k=1}^{n} x_k & \sum_{k=1}^{n} x_k^2 \\ \sum_{k=1}^{n} x_k & \sum_{k=1}^{n} x_k^2 & \sum_{k=1}^{n} x_k^3 \\ \sum_{k=1}^{n} x_k^2 & \sum_{k=1}^{n} x_k^3 & \sum_{k=1}^{n} x_k^4 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^{n} y_k \\ \sum_{k=1}^{n} x_k y_k \\ \sum_{k=1}^{n} x_k^2 y_k \end{pmatrix} \qquad (2)$$

es decir tenemos un sistema

$$Cp = d \qquad (2)$$

que del que tenemos como datos C y d y queremos calcularemos los coeficientes del polinomio de aproximación p.

Ejercicio 1

Escribir una función **`aprox1(f,g,a,b,n)`** que calcule y dibuje el polinomio de aproximación de grado **`g`** para los **`n`** puntos equiespaciados de la función **`f`** en el intervalo **`[a,b]`** (tomando los extremos del intervalo como nodos). La función no devolverá nada: hará los cálculos necesarios para calcular el polinomio de aproximación y dibujará el polinomio de aproximación y los puntos a aproximar. Para ello seguir los pasos siguientes

1. Construir los nodos **`x`** con la función **`np.linspace`** y obtener **`y = f(x)`**
2. Construir la matriz de coeficientes C y la matriz de términos independientes d del sistema (2).
   A. Construir la matriz V del sistema (1) y luego obtener **`C = np.dot(V.T,V)`** y **`d = np.dot(V.T,y)`** , donde **`V.T`** es la matriz tranpuesta de V o
   B. Construir directamente C usando el la orden python `np.sum` que suma los elementos de un vector y teniendo en cuenta que, por ejemplo, `x**2` es un vector cuyos elementos son los de `x` elevados al cuadrado.
3. Resolver el sistema $Cp = d$ usando la orden **`p = np.linalg.solve(C,d)`** .
4. Obtener 50 puntos en el intervalo $[a, b]$ y los correspondientes valores del polinomio **`p`** usando la orden **`pol.polyval`** .
5. Dibujar, en el intervalor $[a, b]$:

   - Los nodos.
   - El polinomio de aproximación obtenido.

Utilizar esta función para obtener y dibujar:

- El polinomio de aproximación de grado $g = 2$ para la función

$$f_1(x) = \text{sen}(x)$$

utilizando $n = 5$ puntos equiespaciados en el intervalo $[a, b] = [0, 2]$.

- El polinomio de aproximación de grado $g = 4$ para la función

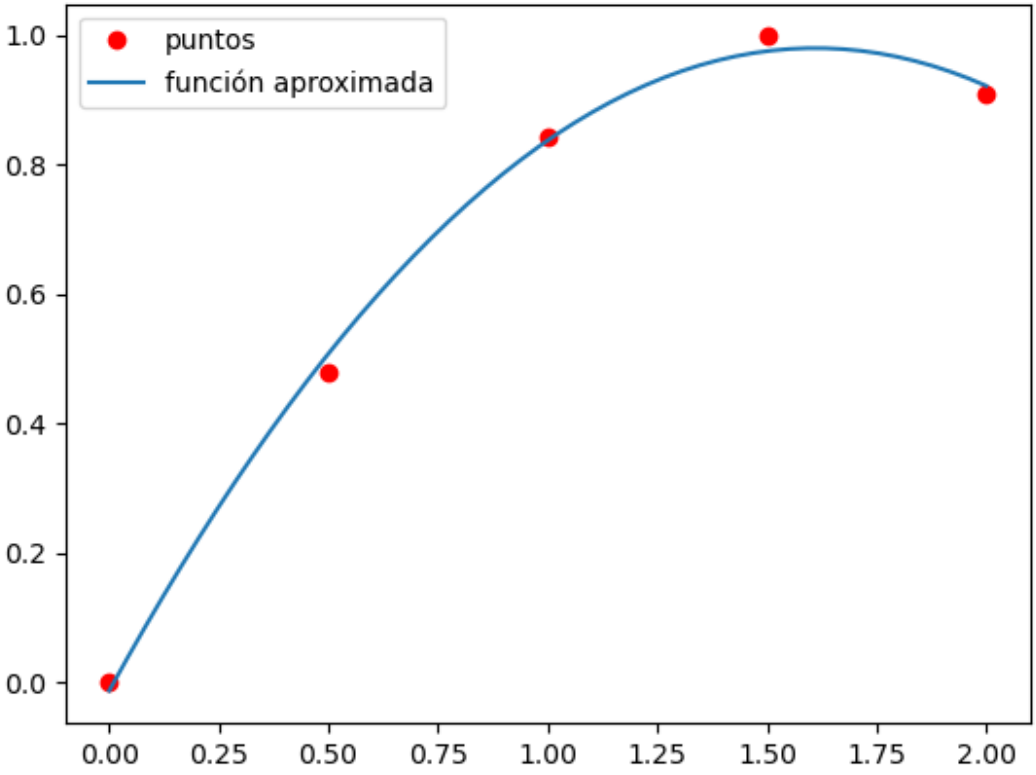
$$f_2(x) = \cos(\arctan(x)) - \log(x + 5)$$

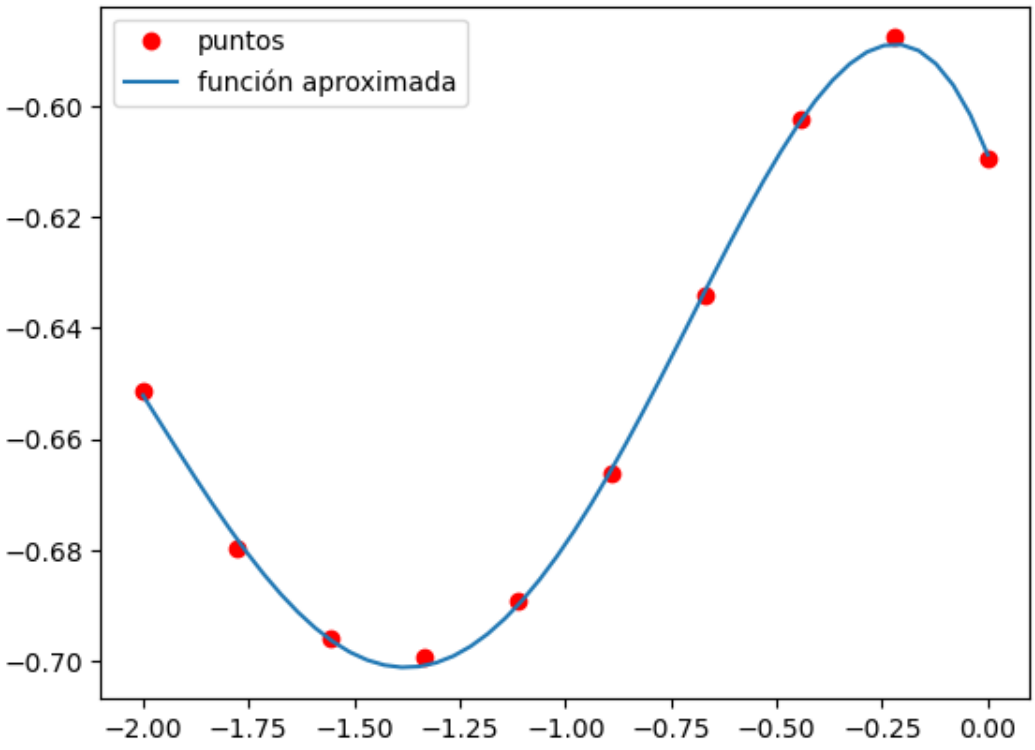
utilizando $n = 10$ puntos equiespaciados en el intervalo $[a, b] = [-2, 0]$.

Nota: Lista de funciones matemáticas elementales en numpy (https://www.pythonprogramming.in/numpy-elementary-mathematical-functions.html).

```
%run Ejercicio1.py
```

```
 Matriz del sistema

[[ 5.    5.    7.5 ]
 [ 5.    7.5  12.5 ]
 [ 7.5  12.5  22.12]]


 Término independiente

[3.23 4.4  6.84]


 Solución del sistema

[-0.01  1.23 -0.38]
```





Podemos obtener los coeficientes del polinomio de aproximación con la orden `polyfit` utilizando como argumento de entrada el grado del polinomio de aproximación.

```
x = np.linspace(0,2,5)
y = np.sin(x)

p = pol.polyfit(x,y,2)

print(p)
```

```
[-0.01  1.23 -0.38]
```

Utilizando el dataset Auto MPG (http://archive.ics.uci.edu/ml/datasets/Auto+MPG) de UC Irvine Machine Learning Repository (http://archive.ics.uci.edu/ml/) vamos a estimar la potencia (*horsepower*) necesaria para un coche de un peso dado (*weight*) y las millas por galón (*mpg*) que sería capaz de recorrer. Los modelos de coche que se manejan en esta base de datos son del año 1970 al 1982. La base de datos también está en kaggle (https://www.kaggle.com/uciml/autompg-dataset?select=auto-mpg.csv).

Utilizando la librería pandas, leemos el fichero de datos `cars.csv` y lo almacenamos como un `dataframe` , que es el equivalente a una hoja de Excel: los datos están organizados en una matriz con una muestra por fila y una variable por columna. Los datos no tienen por qué ser homogéneos. El `dataframe` puede contener variables de distintos tipos (categóricos, ordinales, numéricos continuos y discretos,...)

```
import pandas as pd

df = pd.read_csv('http://www.unioviedo.es/compnum/laboratorios_py/new/cars.csv',se
p=',')
```

Otra forma de leer el fichero sería descargárselo al directorio de trabajo desde cars (http://www.unioviedo.es/compnum/laboratorios_py/new/cars.csv) y leerlo con

```
df = pd.read_csv('cars.csv',sep=',') # Lee datos separados por comas
```

Vamos a echar un vistazo a los datos.

```
df
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	mode yea
0	18.0	8	307.0	130	3504	12.0	7
1	15.0	8	350.0	165	3693	11.5	7
2	18.0	8	318.0	150	3436	11.0	7
3	16.0	8	304.0	150	3433	12.0	7
4	17.0	8	302.0	140	3449	10.5	7
...	...	...	...	...	...	...	.
387	27.0	4	140.0	86	2790	15.6	8
388	44.0	4	97.0	52	2130	24.6	8
389	32.0	4	135.0	84	2295	11.6	8
390	28.0	4	120.0	79	2625	18.6	8
391	31.0	4	119.0	82	2720	19.4	8

392 rows × 9 columns

Para extraer la columna *horsepower*

```
x = df['horsepower']
```

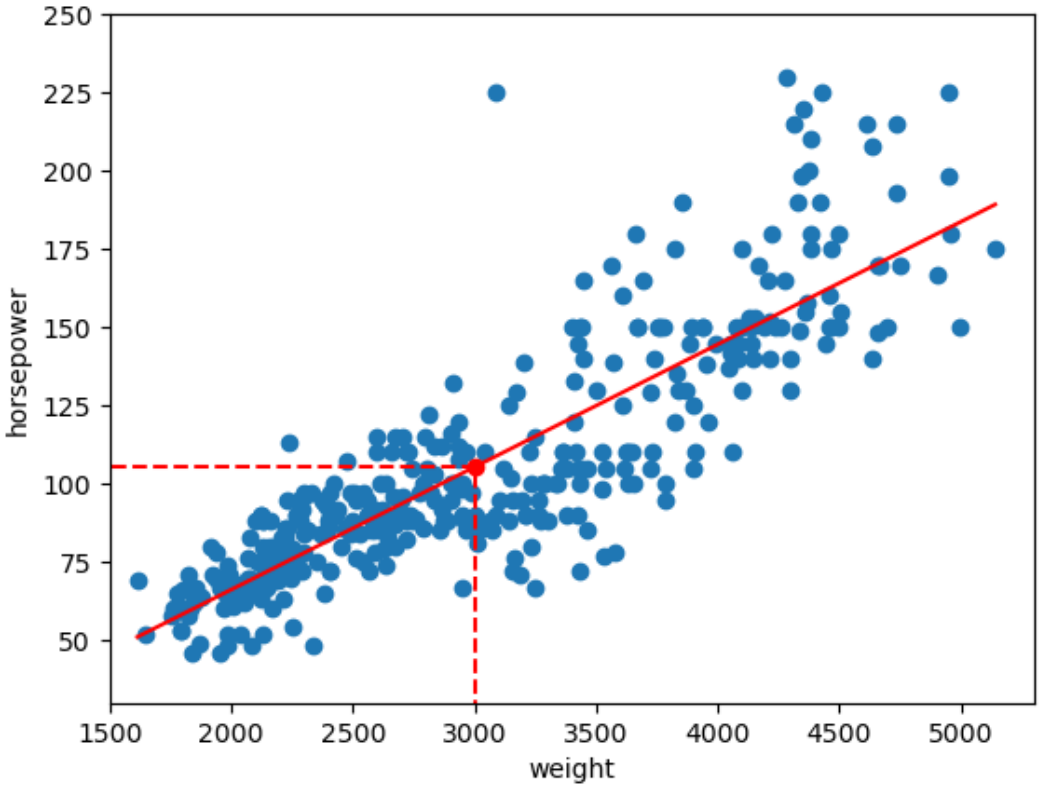
Ejercicio 2

Utilizando el fichero cars (http://www.unioviedo.es/compnum/laboratorios_py/new/cars.csv) que contine datos de Auto MPG (http://archive.ics.uci.edu/ml/datasets/Auto+MPG) de UC Irvine Machine Learning Repository (http://archive.ics.uci.edu/ml/), estimar la potencia (*horsepower*) necesaria para un coche de un peso dado (*weight*) y las millas por galón (*mpg*) que sería capaz de recorrer en ciudad. Para ello

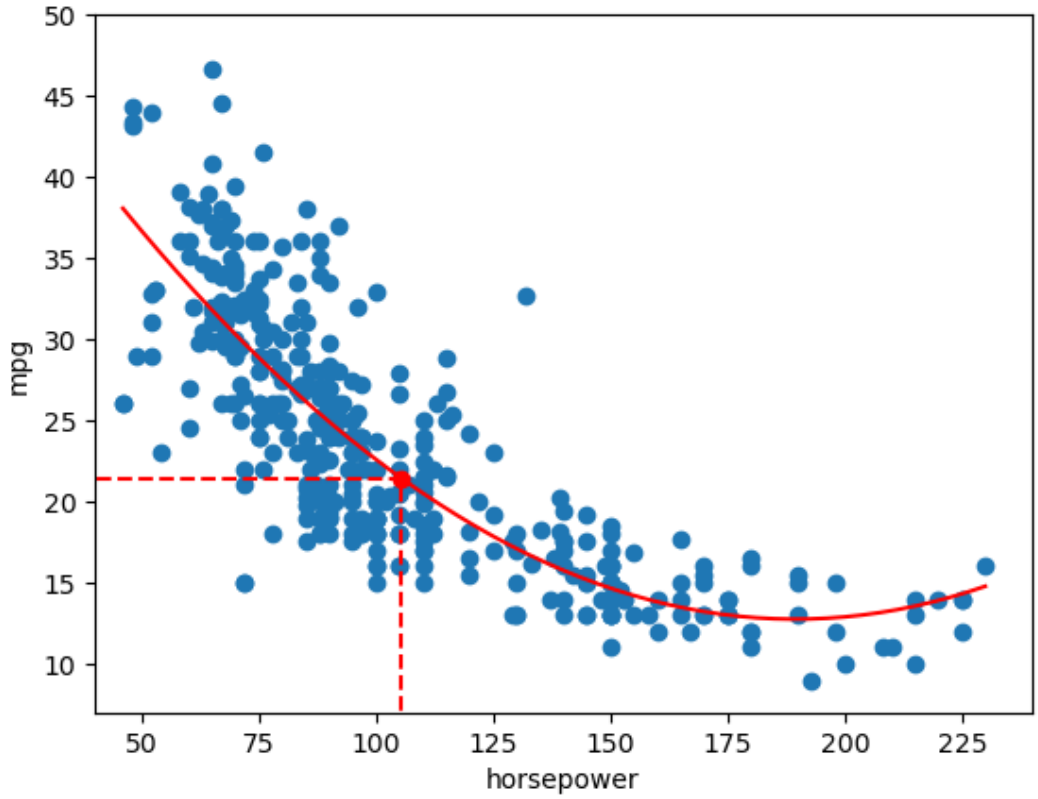
1. Extraer las variables *weight* y *horsepower*.
2. Utilizando `pol.polyfit` calcular el polinomio de grado 1 que aproxima estos valores y tomando como variable x *weight* y como variable y *horsepower*.
3. Utilizando `pol.polyval` estimar la potencia (*horsepower*) de un coche que pesa 3000 libras.
4. Dibujar los puntos, el polinomio de aproximación y el punto sobre la recta para un peso de 3000 libras.
5. Extraer la variable *mpg* y tomando como variable x *horsepower* y como variable y *mpg* calcular el polinomio de grado 2 que aproxima estos puntos.
6. Con la potencia estimada en el apartado para el coche de 3000 libras, estimar las millas por galón que podría recorrer este coche en ciudad utilizando el polinomio de aproximación
7. Dibujar los datos, el polinomio de aproximación respectivo y el punto sobre el polinomio de grado 3 con el que estimamos las millas por galón en ciudad.

```
%run Ejercicio2.py
```

```
105 caballos
```




```
21  mpg
```



Aproximación polinómica continua

Si en lugar de conocer solo algunos puntos de la función conocemos la función completa en un intervalo podemos hacer el ajuste contínuo. En este caso, los sumatorios se convierten en integrales:

$$\begin{pmatrix} \int_a^b 1dx & \int_a^b xdx & \int_a^b x^2dx \\ \int_a^b xdx & \int_a^b x^2dx & \int_a^b x^3dx \\ \int_a^b x^2dx & \int_a^b x^3dx & \int_a^b x^4dx \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} \int_a^b f(x)dx \\ \int_a^b xf(x)dx \\ \int_a^b x^2f(x)dx \end{pmatrix} \qquad (3)$$

Ejercicio 3

Escribir una función **`aprox2(f,g,a,b)`** que calcule y dibuje el polinomio de aproximación de grado **`g`** para la función **`f`** en el intervalo **`[a,b]`** . Para ello seguir los pasos siguientes

1. Construir la matriz de coeficientes y la matriz de términos independientes del sistema (3). Utilizar la función **`scipy.integrate.quad`** . Por ejemplo, para calcular el valor de la integral

$$I = \int_a^b f(x)dx$$

```
from scipy.integrate import quad
Ia = quad(f,a,b)[0]
```

   donde **`f`** es una función lambda. Para obtener las funciones integrando, puedes definir una función lambda a partir de otra. Por ejemplo

```
g = lambda x: x * f(x)
```

2. Resolver el sistema usando la orden **`np.linalg.solve`** y escribir los coeficientes del polinomio solución de mayor a menor grado.
3. Obtener los valores del polinomio cuyos coeficientes hemos obtenido usando la orden **`pol.polyval`** en el intervalo $[a,b]$.
4. Dibujar, en el intervalor $[a,b]$:
   - La función.
   - El polinomio de aproximación obtenido.

Utilizar esta función para obtener y dibujar:

- El polinomio de aproximación de grado $g = 2$ para la función

$$f_1(x) = \text{sen}(x)$$

en el intervalo $[a,b] = [0,2]$.

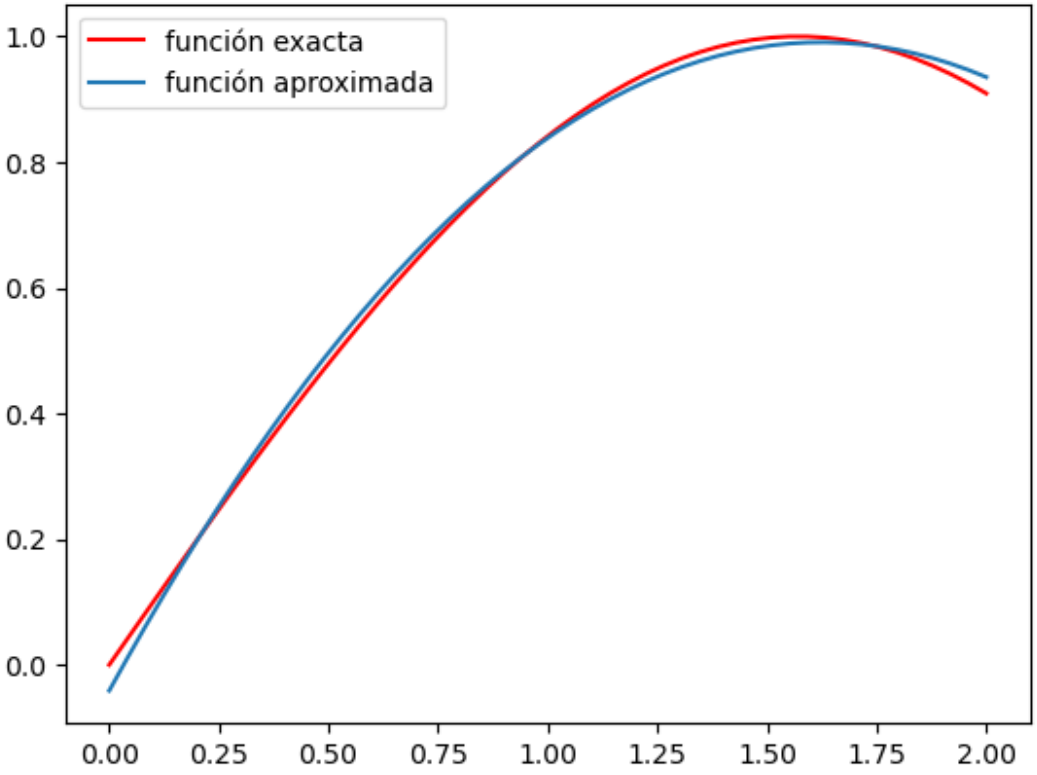
- El polinomio de aproximación de grado $g = 4$ para la función

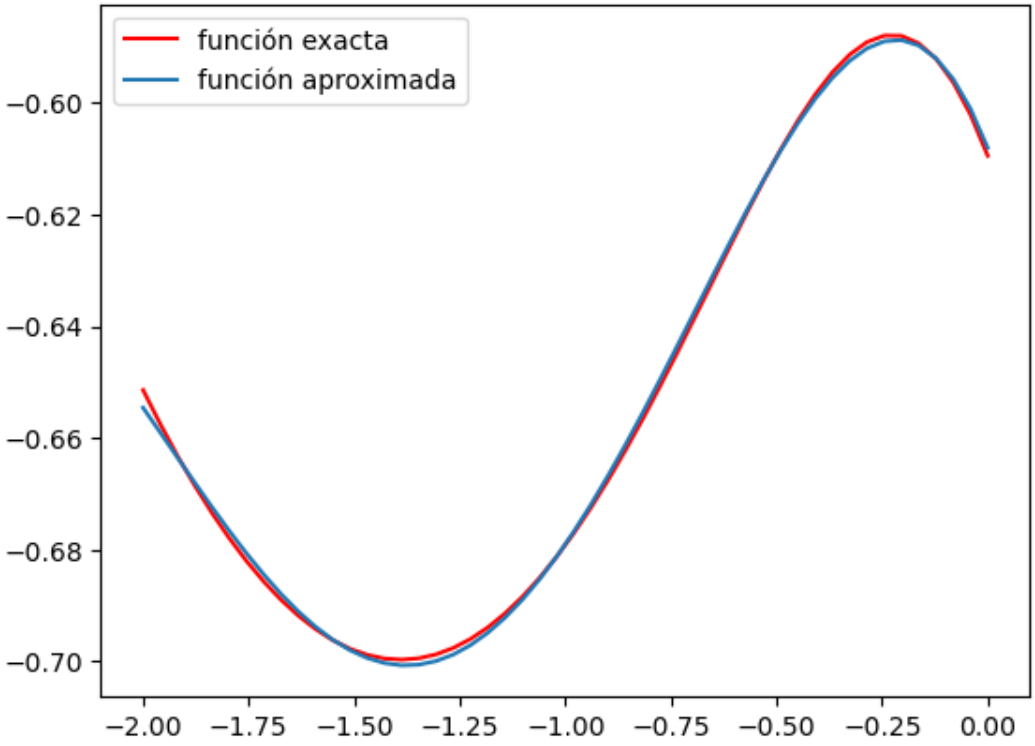
$$f_2(x) = \cos(\arctan(x)) - \log(x+5)$$

en el intervalo $[a,b] = [-2,0]$.

```
%run Ejercicio3.py
```

```
 Matriz del sistema

[[2.   2.   2.67]
 [2.   2.67 4.  ]
 [2.67 4.   6.4 ]]


 Término independiente

[1.42 1.74 2.47]


 Coeficientes del polinomio:

[-0.04  1.27 -0.39]
```





Ejercicios propuestos

Aproximación con polinomios ortogonales

Si suponemos que hemos estado usando el producto escalar de funciones

$$\langle f(x), g(x)\rangle = \int_{-1}^{1} f(x)g(x)dx$$

y la base de polinomios $\{P_0, P_1, P_2\} = \{1, x, x^2\}$ podemos reescribir el sistema como

$$\begin{pmatrix} \langle P_0, P_0\rangle & \langle P_0, P_1\rangle & \langle P_0, P_2\rangle \\ \langle P_1, P_0\rangle & \langle P_1, P_1\rangle & \langle P_1, P_2\rangle \\ \langle P_2, P_0\rangle & \langle P_2, P_1\rangle & \langle P_2, P_2\rangle \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} \langle P_0, f(x)\rangle \\ \langle P_1, f(x)\rangle \\ \langle P_2, f(x)\rangle \end{pmatrix}$$

Podemos mejorar el método anterior si conseguimos que la matriz de coeficientes sea diagonal. Lo conseguiríamos si tuviéramos una base de polinomios ortogonales $\{L_0, L_1, L_2\}$. Entonces se tendría

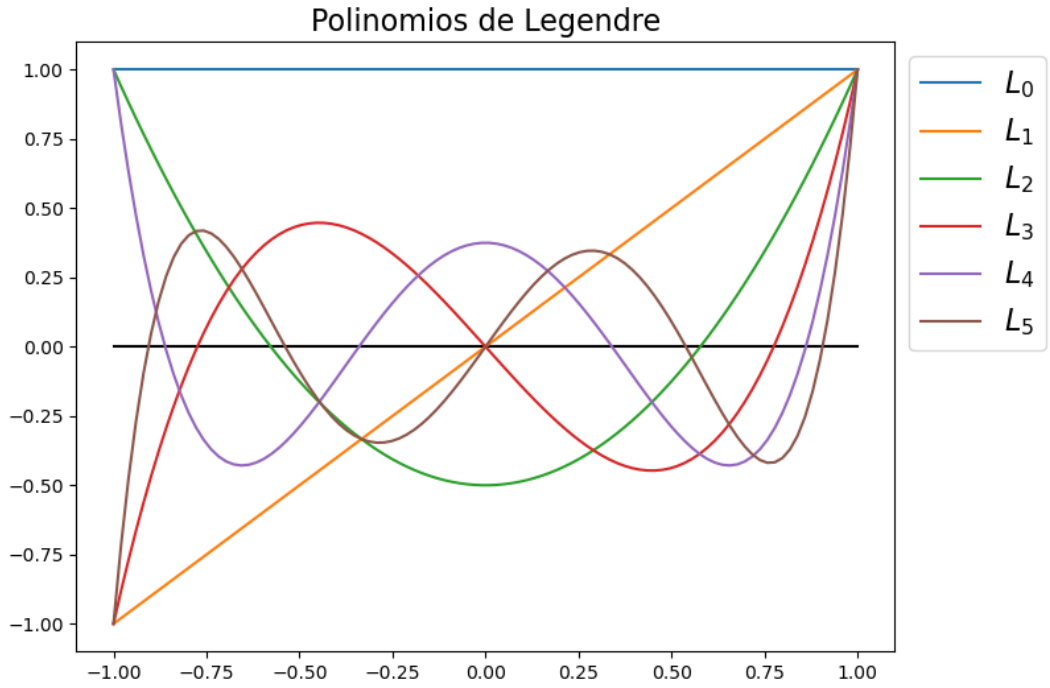
$$\begin{pmatrix} \langle L_0, L_0\rangle & 0 & 0 \\ 0 & \langle L_1, L_1\rangle & 0 \\ 0 & 0 & \langle L_2, L_2\rangle \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} \langle L_0, f(x)\rangle \\ \langle L_1, f(x)\rangle \\ \langle L_2, f(x)\rangle \end{pmatrix}$$

Y ahora tenemos que

$$a_0 = \frac{\langle L_0, f(x)\rangle}{\langle L_0, L_0\rangle} \quad a_1 = \frac{\langle L_1, f(x)\rangle}{\langle L_1, L_1\rangle} \quad a_2 = \frac{\langle L_2, f(x)\rangle}{\langle L_2, L_2\rangle} \qquad (3)$$

Estos polinomios existen y son los *Polinomios de Legendre*. Podemos obtener los polinomios de Legendre del módulo `scipy.special`

```
from scipy.special import eval_legendre

xp = np.linspace(-1,1,100)

plt.figure(figsize=(8,6))
plt.plot(xp,0*xp,'k-')
for i in range(6):
    plt.plot(xp,eval_legendre(i,xp),label=r'$L_'+str(i)+'$')
plt.legend(bbox_to_anchor=(1, 1), loc='upper left', ncol=1,fontsize=16)
plt.title('Polinomios de Legendre',fontsize=16)
plt.show()
```



Si resolvemos el problema anterior usando los polinomios ortogonales, obtenemos el mismo resultado. Y el polinomio de aproximación será

$$P_2(x) = a_0\,L_0(x) + a_1\,L_1(x) + a_2\,L_2(x) \qquad (4)$$

Ejercicio 4

Escribir una programa que calcule el polinomio de aproximación de grado dos para la función $f_1(x) = \cos(x)$ en el intervalo $[-1, 1]$. Para ello:

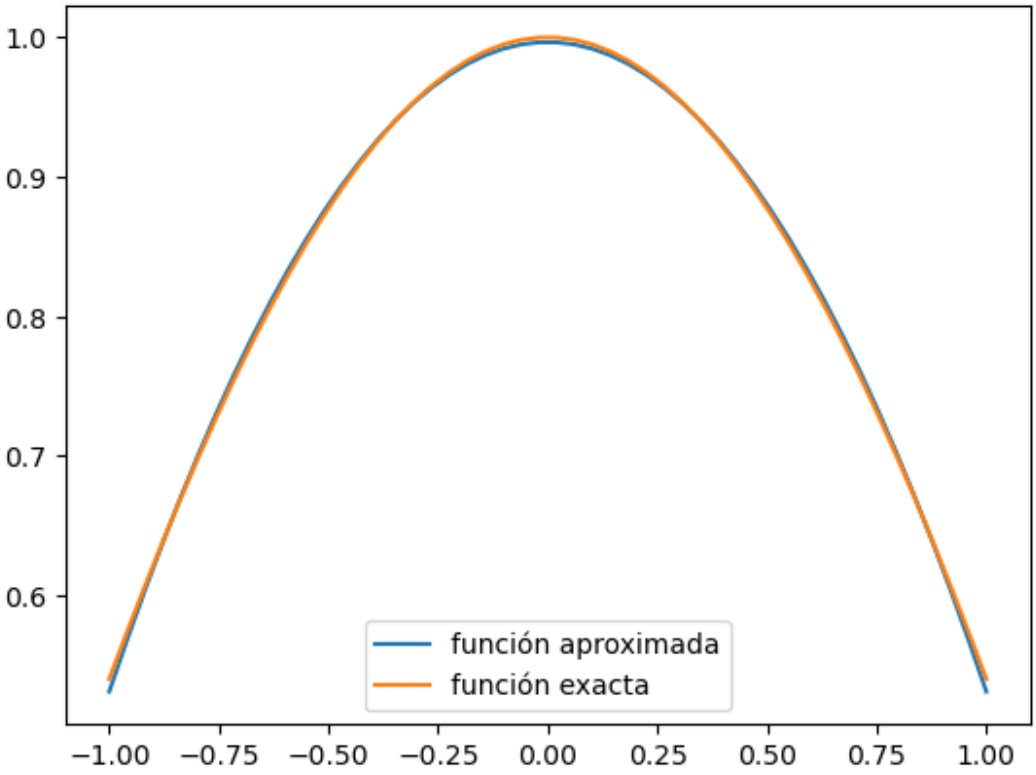
- Calcular los coeficientes (3). Utilizar la función **`scipy.integrate.quad`** y los polinomios de Legendre de **`scipy.special`**.
- Obtener los valores del polinomio (4) en el intervalo $[-1, 1]$.
- Dibujar, en el intervalor $[-1, 1]$:
  - La función.
  - El polinomio de aproximación obtenido.
- Calcular el error relativo

$$E_r = \frac{\|\cos(x) - P(x)\|}{\|\cos(x)\|}$$

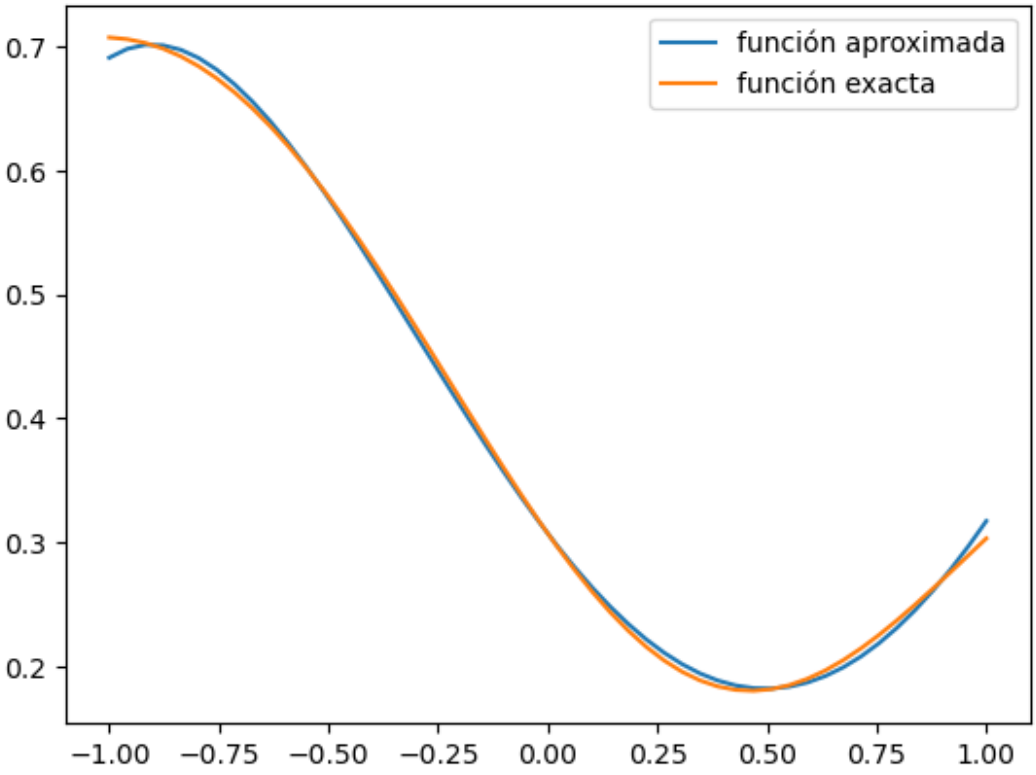
  usando la función **`np.linalg.norm`**.

Calcular también la aproximación polinómica de grado 4 de $f_2(x) = \cos(\arctan(x)) - e^{-x^2}\log(x+2)$. Dar el error relativo y dibujar la función y la aproximación polinómica como en el ejemplo anterior.

```
%run Ejercicio4.py
```




```
Er de f1 =  0.0038870502951536636
```




```
Er de f2 =  0.012674424229446089
```

Aproximación con funciones trigonométricas: Fourier

Si f es una función periódica de periodo T la aproximación mediante funciones trigonométricas resulta especialmente adeduada. Utilizaríamos el producto escalar

$$\langle f(x), g(x)\rangle = \int_{\lambda}^{\lambda+T} f(x)g(x)dx$$

donde el intervalo de integración es un intervalo que abarca un periodo completo.

La base de funciones trigonométricas

$$\left\{1, \cos\left(\frac{2\pi x}{T}\right), \text{sen}\left(\frac{2\pi x}{T}\right), \cos\left(2\frac{2\pi x}{T}\right), \text{sen}\left(2\frac{2\pi x}{T}\right), \ldots, \cos\left(n\frac{2\pi x}{T}\right), \text{sen}\left(n\frac{2\pi x}{T}\right.\right.$$

que es una base ortogonal para el producto escalar dado. Entonces, podemos escribir el sistema $Ax = b$ donde

$$A = \begin{pmatrix} \int_{\lambda}^{\lambda+T} 1^2 dx & 0 & 0 & \cdots & 0 \\ 0 & \int_{\lambda}^{\lambda+T} \cos^2\left(\frac{2\pi x}{T}\right)dx & 0 & \cdots & 0 \\ 0 & 0 & \int_{\lambda}^{\lambda+T} \text{sen}^2\left(\frac{2\pi x}{T}\right)dx & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & \int_{\lambda}^{\lambda+T} \cos^2\left(n\frac{2\pi x}{T}\right)dx \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

$$= \begin{pmatrix} \int_{\lambda}^{\lambda+T} f(x)dx \\ \int_{\lambda}^{\lambda+T} f(x)\cos\left(\frac{2\pi x}{T}\right)dx \\ \int_{\lambda}^{\lambda+T} f(x)\text{sen}\left(\frac{2\pi x}{T}\right)dx \\ \cdots \\ \int_{\lambda}^{\lambda+T} f(x)\cos\left(n\frac{2\pi x}{T}\right)dx \\ \int_{\lambda}^{\lambda+T} f(x)\text{sen}\left(n\frac{2\pi x}{T}\right)dx \end{pmatrix}$$

Y la función que aproximaría a f sería de la forma:

$$F(x) = \frac{a_0}{2} + a_1\cos\left(\frac{2\pi x}{T}\right) + b_1\text{sen}\left(\frac{2\pi x}{T}\right) + a_2\cos\left(2\frac{2\pi x}{T}\right) + b_2\text{sen}\left(2\frac{2\pi x}{T}\right) + \cdots$$

Y ahora tenemos que

$$\frac{a_0}{2}=\frac{\int_\lambda^{\lambda+T} f(x)dx}{\int_\lambda^{\lambda+T} 1^2dx} \quad a_1=\frac{\int_\lambda^{\lambda+T} f(x)\cos\left(\frac{2\pi x}{T}\right)dx}{\int_\lambda^{\lambda+T} \cos^2\left(\frac{2\pi x}{T}\right)dx} \quad b_1=\frac{\int_\lambda^{\lambda+T} f(x)\text{sen}\left(\frac{2\pi x}{T}\right)dx}{\int_\lambda^{\lambda+T} \text{sen}^2\left(\frac{2\pi x}{T}\right)dx},\ldots,$$

$$a_n=\frac{\int_\lambda^{\lambda+T} f(x)\cos\left(n\frac{2\pi x}{T}\right)dx}{\int_\lambda^{\lambda+T} \cos^2\left(n\frac{2\pi x}{T}\right)dx} \quad b_n=\frac{\int_\lambda^{\lambda+T} f(x)\text{sen}\left(n\frac{2\pi x}{T}\right)dx}{\int_\lambda^{\lambda+T} \text{sen}^2\left(n\frac{2\pi x}{T}\right)dx}$$

Como se tiene que

$$\int_\lambda^{\lambda+T} 1^2dx=T, \quad \int_\lambda^{\lambda+T} \cos^2\left(k\frac{2\pi x}{T}\right)dx=\frac{T}{2}, \quad \int_\lambda^{\lambda+T} \text{sen}^2\left(k\frac{2\pi x}{T}\right)dx=\frac{T}{2}$$

para cualquier k entero positivo, los coeficientes de las funciones de la base serán

$$a_0=\frac{2}{T}\int_\lambda^{\lambda+T} f(x)dx, \quad a_1=\frac{2}{T}\int_\lambda^{\lambda+T} f(x)\cos\left(\frac{2\pi x}{T}\right)dx, \quad b_1=\frac{2}{T}\int_\lambda^{\lambda+T} f(x)\text{sen}\left(\cdot$$

$$b_n=\frac{2}{T}\int_\lambda^{\lambda+T} f(x)\text{sen}\left(n\frac{2\pi x}{T}\right)dx$$

Vamos a dibujar la serie de Fourier desde el término 1 hasta el término 5 para la función $f(x)=x$ en el intervalo $[0,3]$ de periodo $T=3$.

Ejercicio 5

- Dibujar la serie de Fourier de orden 5 para la función $f(x)=x$ en el intervalo $[0,3]$ de periodo $T=3$.
- Dibujar la serie de Fourier de orden 6 asociada a la función $f(x)=(x-\pi)^2$ en el intervalo $[0,2\pi]$ de periodo $T=2\pi$.

```
%run Ejercicio5.py
```