Introducción a la **Computación Científica** con **Python**
Clase 5: **SciPy**

Diego Passarella

Víctor Viana

EDUCACION PERMANENTE
Universidad de la República

SEDE TACUAREMBÓ
CENUR NORESTE
UNIVERSIDAD DE LA REPÚBLICA URUGUAY

Contenido

- Apenas una mirada por SciPy
- Interpolación y ajuste de funciones
- Derivadas
- Integrales

El ecosistema SciPy

Vamos a utilizar como principal fuente de información, la página del proyecto:

https://www.scipy.org/about.html

Se emplea el término "ecosistema" porque refiere a una colección de distintos softwares empleados en la computación científica en Python.

El núcleo de ese ecosistema está compuesto por: Python, Numpy, SciPy, Matplotlib. Sobre ellos se pueden agregar: pandas, scikit-learn, ..., ∞

El ecosistema SciPy

Una muy breve mirada a la librería SciPy puede comenzar por: https://docs.scipy.org/doc/scipy/reference/tutorial/index.html

SciPy está organizado por "subpaquetes", los cuales cubren áreas específicas de la computación científica.

Cada subpaquete debe ser importado de forma separada.

El ecosistema SciPy

Organización de SciPy

#Subpackage	Description
#cluster	Clustering algorithms
#constants	Physical and mathematical constants
#fftpack	Fast Fourier Transform routines
#integrate	Integration and ordinary differential equation solvers
#interpolate	Interpolation and smoothing splines
#io	Input and Output
#linalg	Linear algebra
#ndimage	N-dimensional image processing
#odr	Orthogonal distance regression
#optimize	Optimization and root-finding routines
#signal	Signal processing
#sparse	Sparse matrices and associated routines
#spatial	Spatial data structures and algorithms
#special	Special functions
#stats	Statistical distributions and functions

```
from scipy import linalg, optimize
```

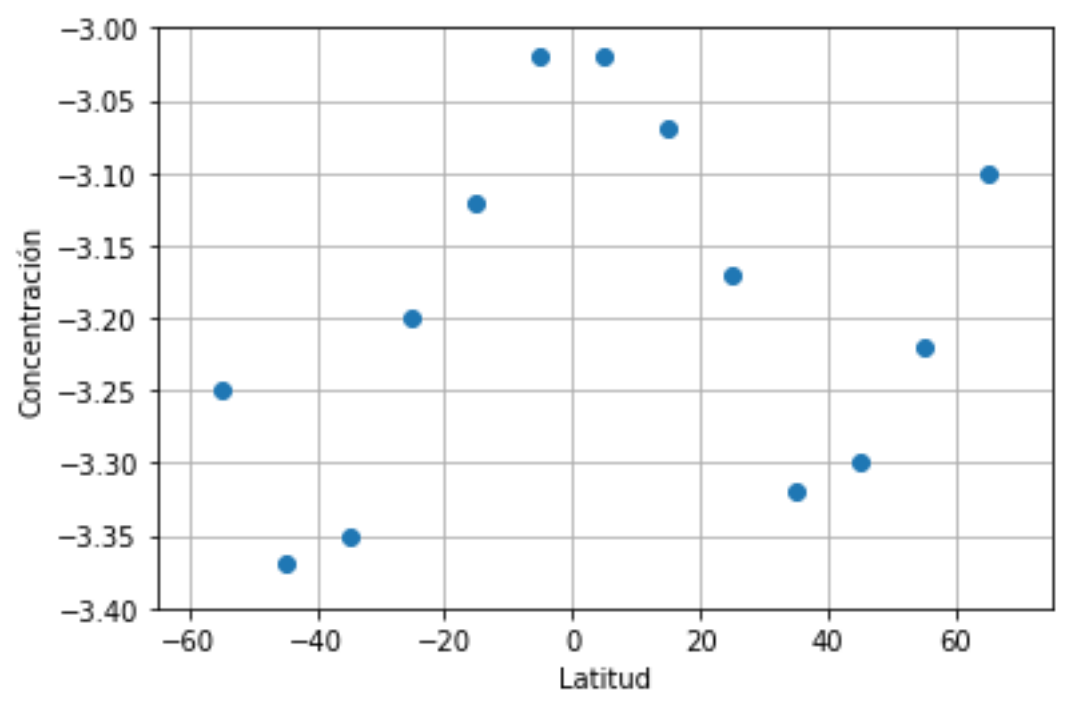
Ajuste de funciones

El caso de disponer de datos experimentales y no conocer la función que los describe.

Si tengo "n" pares de puntos (x_i, y_i), puedo construir un único polinomio que pase por todos y cada uno de esos puntos.

Ese polinomio será de orden n-1.

Ajuste de funciones



un ejemplo de datos experimentales extraído del Cap. 3 de "Scientific Computing with Matlab and Octave". A. Quarteroni, F. Saleri.

Ajuste de funciones

```
p = np.polyfit(x,y,n)
```

Me devuelve un array con los n+1 coeficientes del polinomio de ajuste, ordenados de forma

```
p[0]*x**n + p[1]*x**(n-1) + ... + p[n]*x + p[n+1]
```

Se debe cumplir que x e y posean por lo menos n+1 elementos.

Si poseen más elementos, el ajuste que realiza es por mínimos cuadrados.

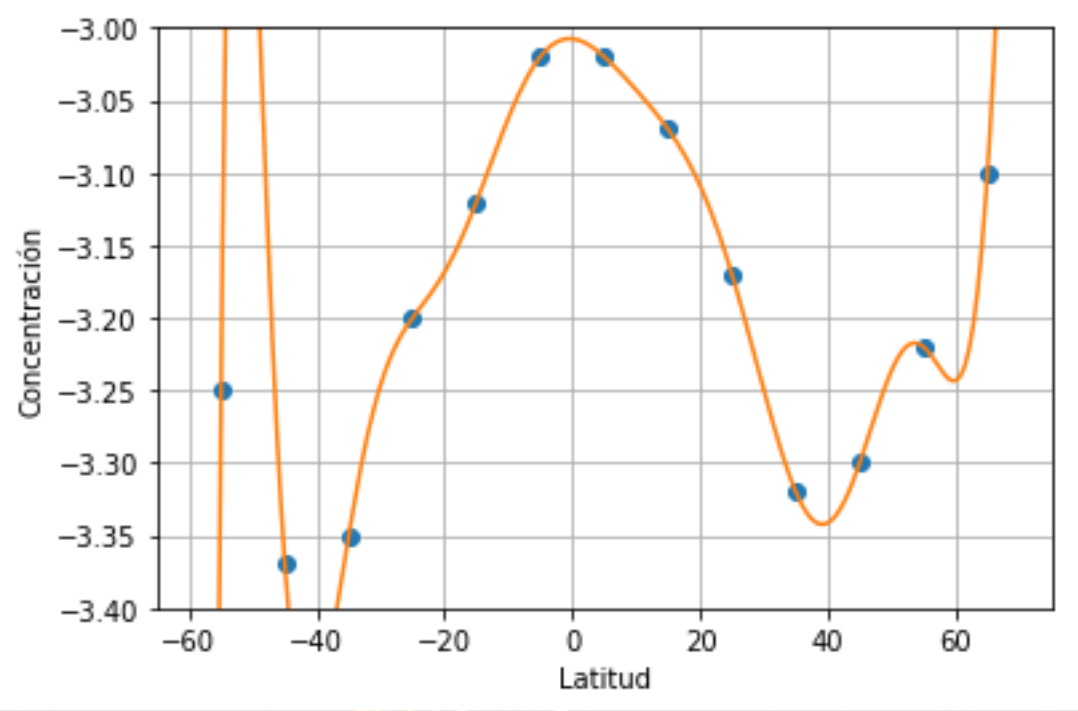
Ajuste de funciones

Dado un array p, se puede evaluar el polinomio en distintos puntos del dominio

```
p = np.polyfit(x,y,n)

y2 = np.polyval(p,x2)
```

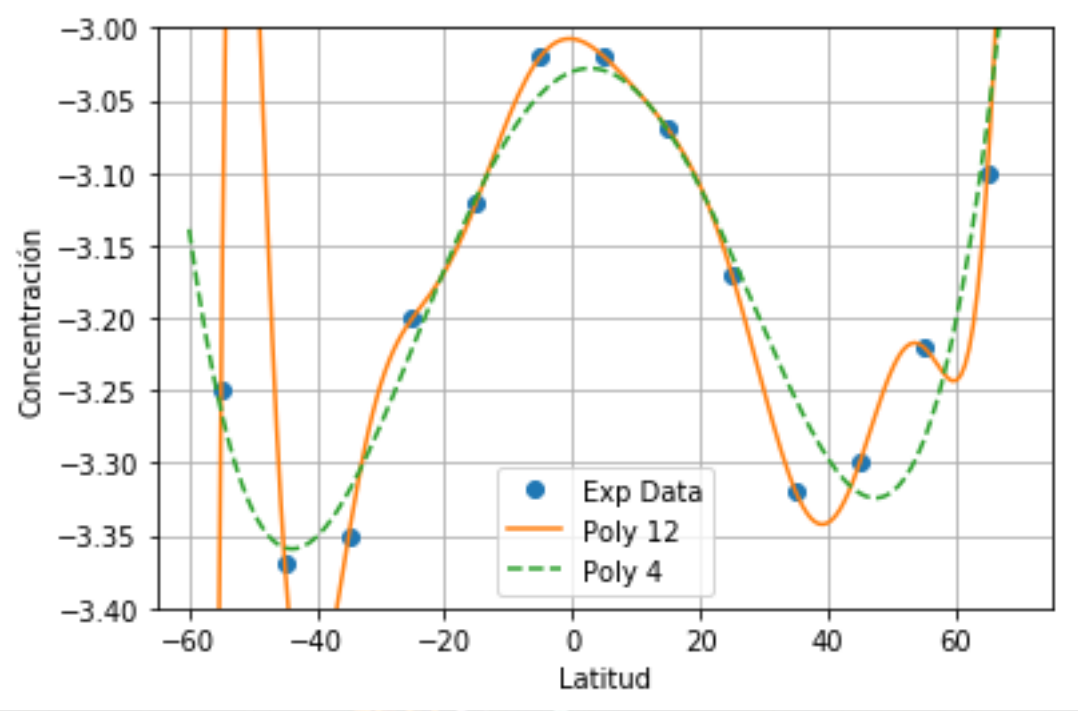
Algunos resultados:

Ajuste de funciones



Polinomio de orden 12 que pasa exactamente por cada uno de los datos experimentales...

Ajuste de funciones



Polinomio de orden 4 que ajusta por mínimos cuadrados los datos experimentales

Ajuste de funciones

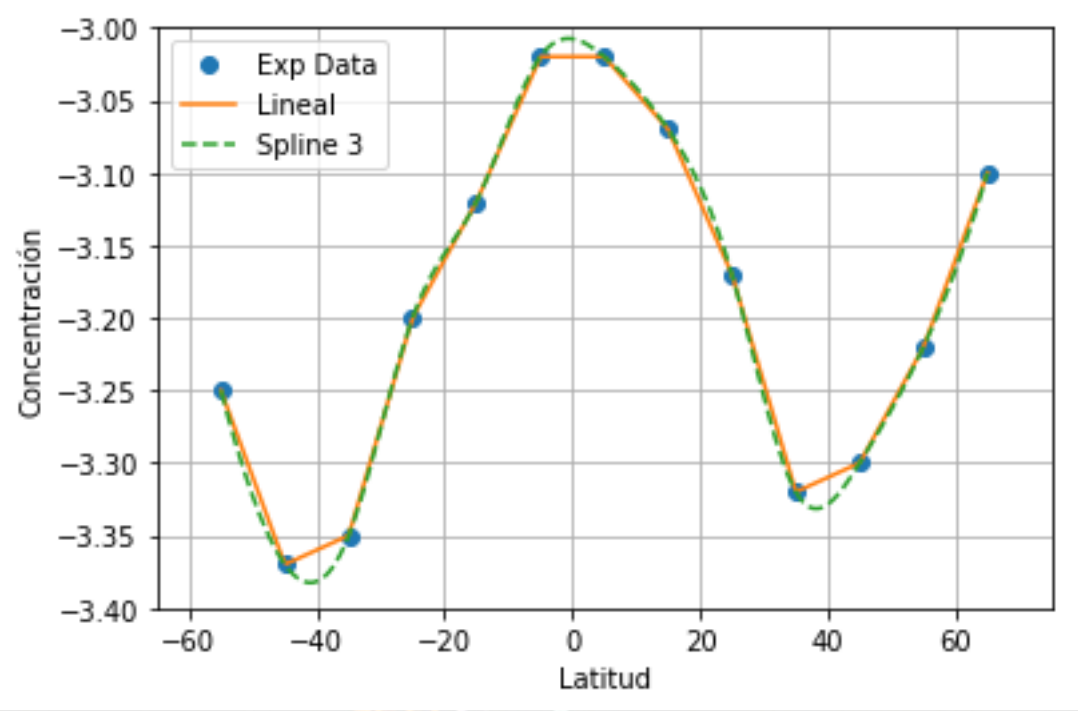
Si no se conoce la funcionalidad a priori de los resultados experimentales, emplear polinomios (y más aún de alto orden) no es conveniente.

Splines:

```
from scipy.interpolate import interp1d

y_lin = interp1d(x_exp,y_exp)

y_sp3 = interp1d(x_exp,y_exp, kind = 'cubic')
```

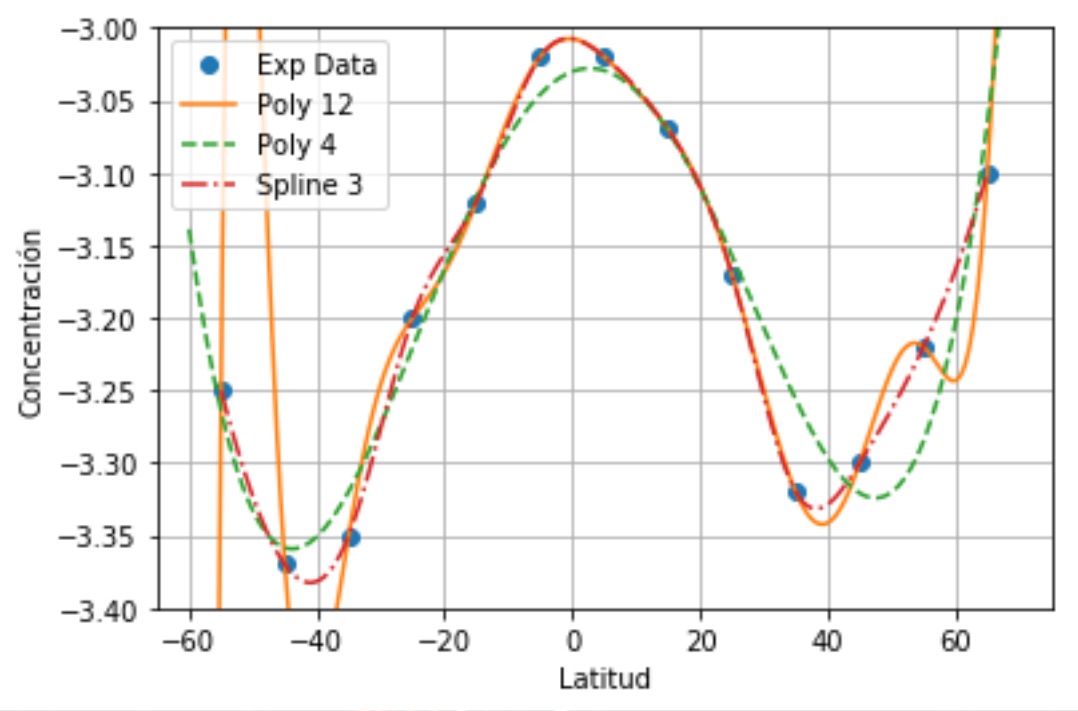
Notar que `y_lin` y `y_sp3` son funciones que pueden ser llamadas por un dado argumento

Ajuste de funciones



Splines lineales y cúbicos de los datos experimentales.

Ajuste de funciones



Comparación entre splines y polinomios.

Ajuste de funciones

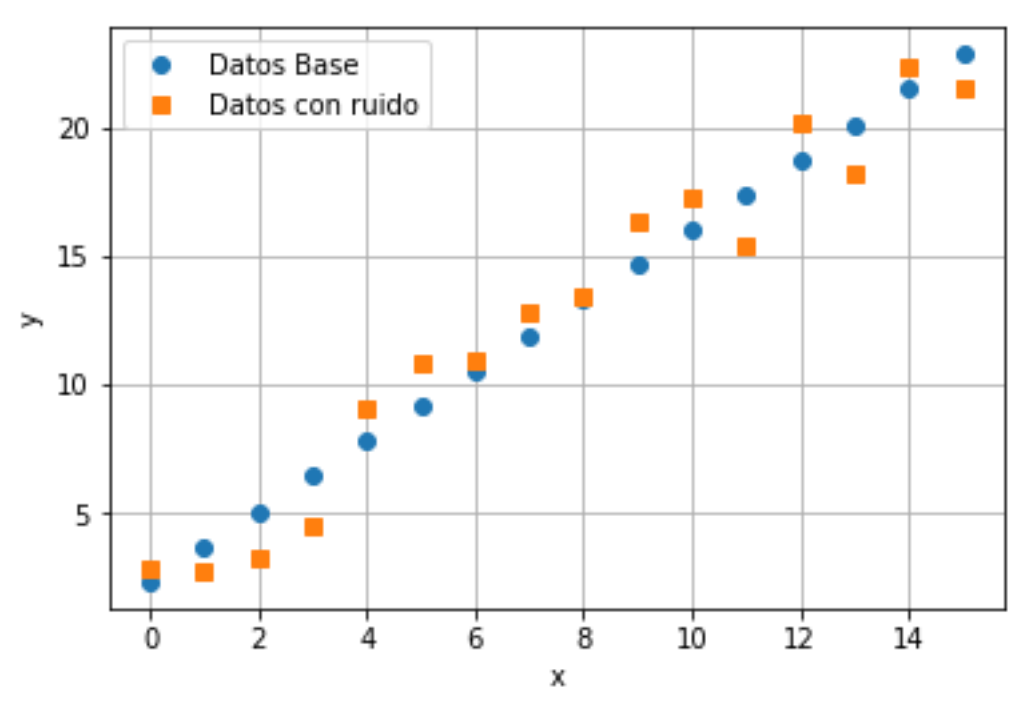
El ajuste lineal es uno de los más utilizados en el análisis de datos. El método polyfit no arroja información sobre el coeficiente R^2 de ajuste.

Para ello se debe instalar la librería scikit-learn (si uno no quiere programarse una función que calule R^2)

```
from sklearn.metrics import r2_score

r2 = r2_score(y2, y2_Fit)
```

Donde se compara el y2 (medido) vs. el y2_fit (entregado por el ajuste

Ajuste de funciones



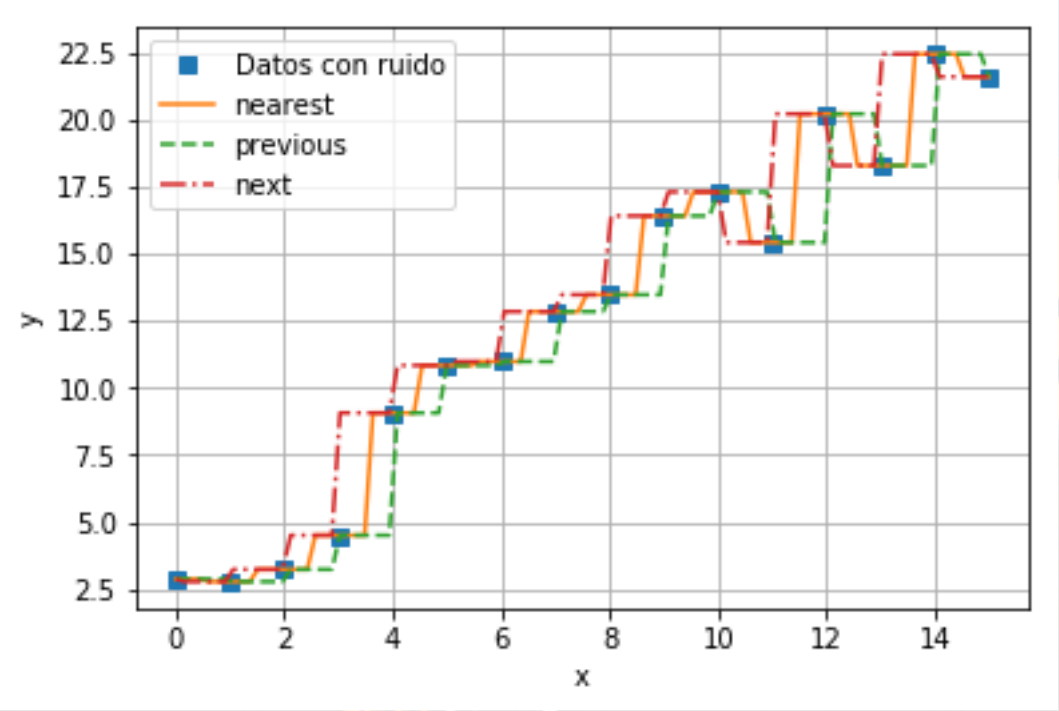
Ver el ejemplo en el script

Ajuste de funciones

Algunas funcionalidades más de interp1d

```
f1 = interp1d(x, y2, kind='nearest')

f2 = interp1d(x, y2, kind='previous')

f3 = interp1d(x, y2, kind='next')
```

Ajuste de funciones



Continuamos desde Spyder...

Derivadas

Se pueden tomar dos caminos:

- Si se conoce la función f(x), se puede importar la función derivative, que realiza una derivada centrada de f, alrededor de un punto x0.
- Si solamente se dispone de un conjunto de puntos (x,y), se puede plantear un esquema de derivadas centradas, basado en:

$$y'[i] = \frac{y[i+1] - y[i-1]}{x[i+1] - x[i-1]}$$

El cual es válido para los puntos interiores del vector x.

Derivadas

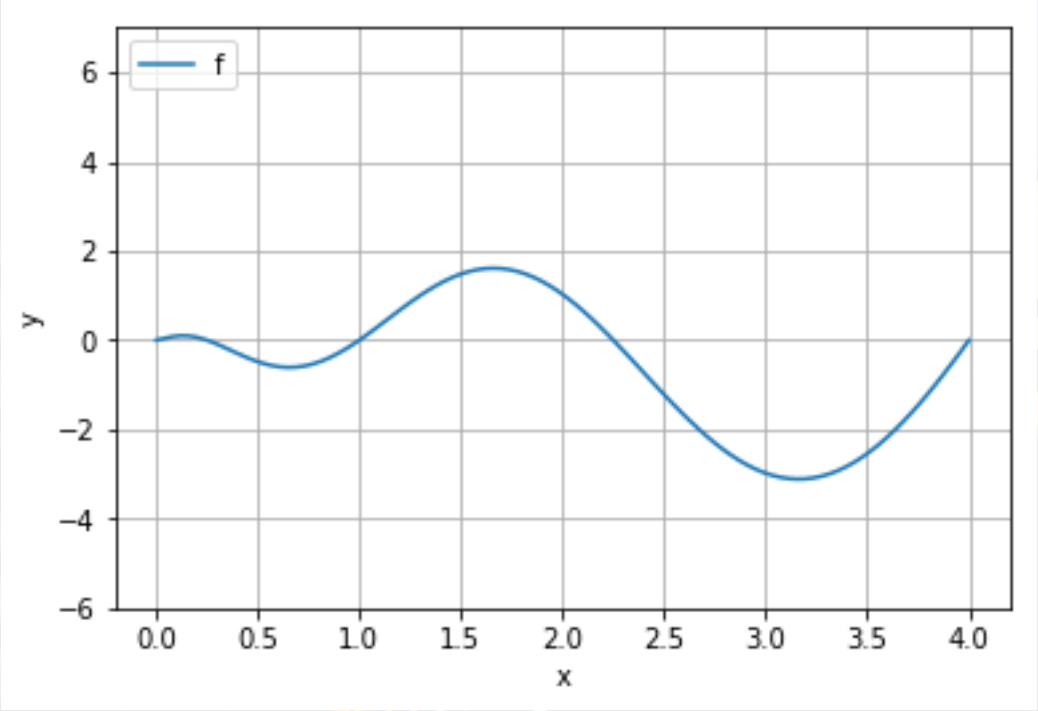
```
from scipy.misc import derivative

dfun = derivative(fun,x0,dx=1e-6)
```

Se pasa la función `fun` (previamente definida), el punto x0 donde se desea evaluar la derivada y el delta que se toma a cada lado de x0 para aproximar a la derivada centrada.

Notar que `fun` toma como argumento a un escalar y la derivación es numérica (no simbólica).

`Revisar la función de derivada centrada en el script`

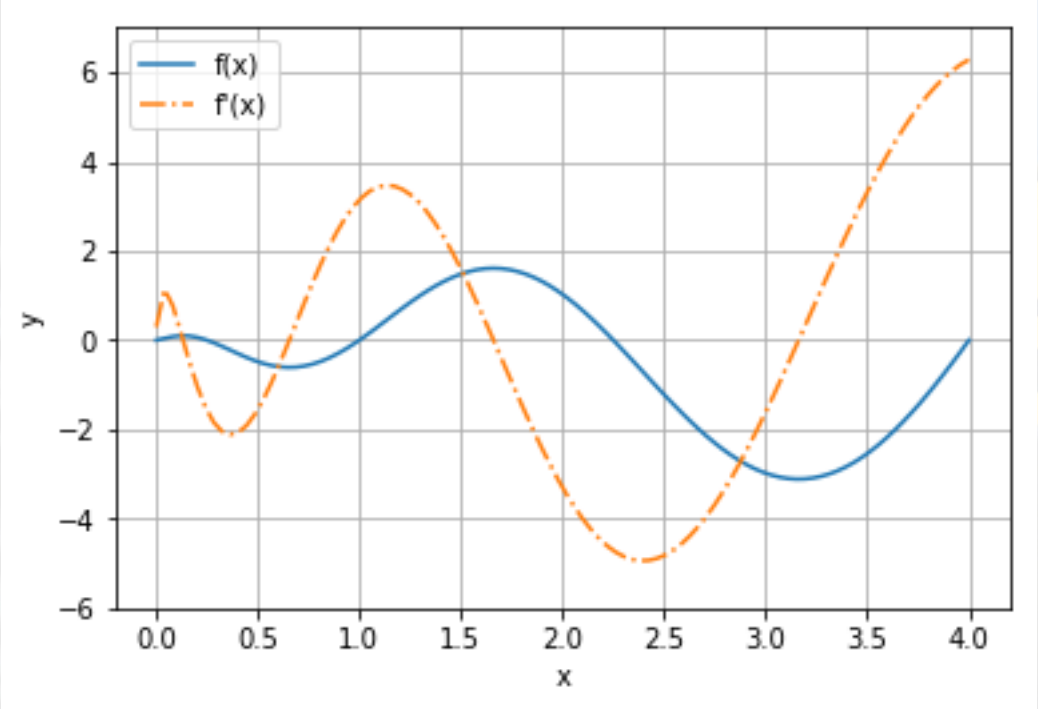
Derivadas



Ejemplo:

$f(x) = \text{sen}(2\pi\sqrt{x})x$

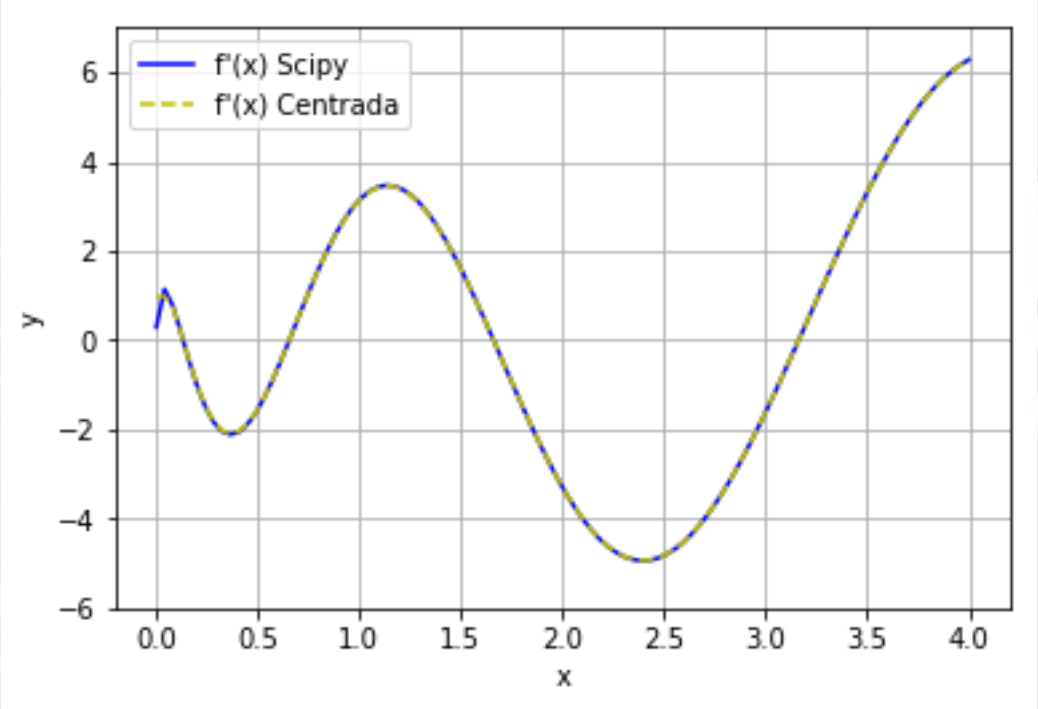
Derivadas



Ejemplo:
f(x) = sen(2π√x)x

f'(x) calculada con la función "derivative"

Derivadas



Ejemplo:
$f(x) = \text{sen}(2\pi\sqrt{x})x$

Comparación de derivadas obtenidas con "derivative" de SciPy y la función de derivada centrada del script.

Integrales

De forma análoga al caso de derivación, se pueden seguir dos caminos:

- Métodos de integración conociendo la función objetivo y los límites de integración.
- Integral de un conjunto de datos que se dispone (función desconocida)

Integrales

La función `quad` se utiliza de forma general para la integración de funciones de una variable, dado sus intervalos (leer el help de integrate para más opciones de integración).

Se le pasa la función `fun` ya definida y el límite inferior (`a`) y superior (`b`) de integración.

Devuelve el valor de la integral definida `IntQuad` y un estimador del error cometido `err`.

```
from scipy.integrate import quad

(IntQuad,err) = quad(fun,a,b)
```

Integrales

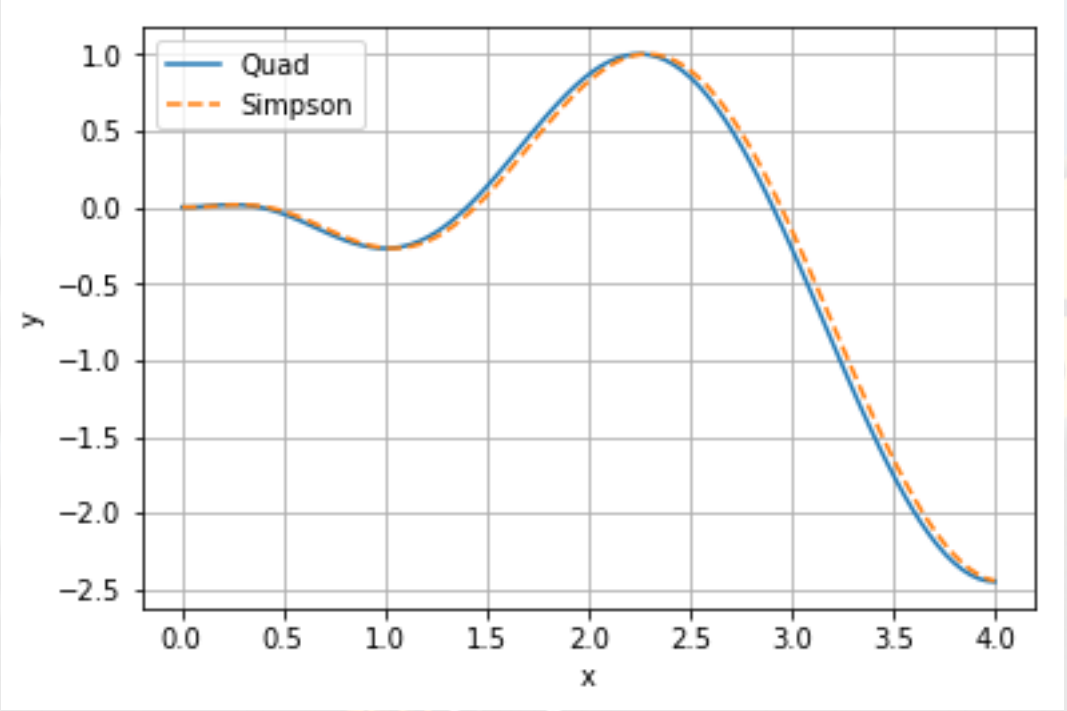
La función `simps` realiza la integral por el método de Simpson dados un par de vectores x e y

```
from scipy.integrate import simps

IntSimp = simps(x,y)
```

Dados (x0,y0), (x1,y1) y (x2,y2) (equiespaciados):

$$\int_{x_0}^{x_2} f(x)\,dx \simeq \frac{x_2 - x_0}{6}\left[y_0 + 4\,y_1 + y_2\right]$$

Derivadas



Ejemplo:
$f(x) = \text{sen}(2\pi\sqrt{x})x$

Compración de las integrales de f(x) usando el método Quad y Simpson.