دوره جامع پایتون:

بخش محاسبات عددی

جلسه هفدهم

دکتر ذبیح اله ذبیحی

کتابخانه Scipy

Pip install scipy

---

import scipy

print(scipy.__version__)

- **constants**: physical constants and conversion factors
- **cluster**: hierarchical clustering, vector quantization, K-means
- **fft**: Discrete Fourier Transform algorithms
- **fftpack**: Legacy interface for Discrete Fourier Transforms
- **integrate**: numerical integration routines
- **interpolate**: interpolation tools
- **io**: data input and output
- **lib**: Python wrappers to external libraries
- **linalg**: linear algebra routines
- **misc**: miscellaneous utilities (e.g. image reading/writing)
- **ndimage**: various functions for multi-dimensional image processing
- **optimize**: optimization algorithms including linear programming
- **signal**: signal processing tools
- **sparse**: sparse matrix and related algorithms
- **spatial**: KD-trees, nearest neighbors, distance functions
- **special**: special functions
- **stats**: statistical functions
- **weave**: tool for writing C/C++ code as Python multiline strings

ثابت ها

- لیست ثوابت

```
from scipy import constants
print(dir(constants))
print(constants.pi)
```

```
print(constants.yotta)    #1e+24
print(constants.zetta)    #1e+21
print(constants.exa)      #1e+18
print(constants.peta)     #1000000000000000.0
print(constants.tera)     #1000000000000.0
print(constants.giga)     #1000000000.0
print(constants.mega)     #1000000.0
print(constants.kilo)     #1000.0
print(constants.hecto)    #100.0
print(constants.deka)     #10.0
print(constants.deci)     #0.1
print(constants.centi)    #0.01
print(constants.milli)    #0.001
print(constants.micro)    #1e-06
print(constants.nano)     #1e-09
print(constants.pico)     #1e-12
print(constants.femto)    #1e-15
print(constants.atto)     #1e-18
print(constants.zepto)    #1e-21
```

```
print(constants.atomic_mass) #1.66053904e-27
print(constants.m_u)         #1.66053904e-27
print(constants.u)           #1.66053904e-27

print(constants.minute)      #60.0
print(constants.hour)        #3600.0
print(constants.day)         #86400.0
print(constants.week)        #604800.0
print(constants.year)        #31536000.0

print(constants.kmh)           #0.2777777777777778
print(constants.mph)           #0.4470399999999994
print(constants.mach)          #340.5
print(constants.speed_of_sound) #340.5
print(constants.knot)          #0.5144444444444445
```

ریشه معادلات

F(x)=0

root(fun,x0)

```
from scipy.optimize import root
from math import cos

def f(x):
  return x + cos(x)

myroot = root(f, 0)
print(myroot.x)
```

چند معادله چندمجهول

x+3y+5z=10
2x+5y+z=8
2x+3y+8z=3

```
from scipy import linalg
import numpy as np
a = np.array([[1, 3, 5], [2, 5, 1], [2, 3, 8]])
b = np.array([10, 8, 3])
x = linalg.solve(a, b)
print (x)
```

دترمینان

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = 1 * 4 - 2 * 3 = 6$$

```
from scipy import linalg
import numpy as np
A = np.array([[1,2],[3,4]])
x = linalg.det(A)
print (x)
```

ویژه مقادیر و ویژه بردار

```
from scipy import linalg
import numpy as np
A = np.array([[1,2],[3,4]])
a, b = linalg.eig(A)
print(a)
print (b)
```

درون یابی (interpolation) و برون یابی (extrapolation)

- فرض کنید جدول تابعی موجود باشد (یعنی مقادیر داده ها در بازه مشخصی را داشته باشیم)

x	x_1	x_2	x_n
f(x)	$f(x_1)$	$f(x_2)$ ...	$f(x_n)$

- اگر بخوایم مقدار تابع را در نقاطی داخل بازه $[x_1,x_n]$ تخمین بزنیم این عمل را درون یابی گویند.
- اگر بخوایم مقدار تابع را در نقاطی خارج بازه $[x_1,x_n]$ تخمین بزنیم این عمل را برون یابی گویند.

- در برون یابی فرض برین است که رفتار تابع خارج از بازه مشابه داخل بازه است. اما امکان دارد این مطلب همیشه معتبر نباشد.

درون یابی

- تکنیک عمومی درون یابی، برازش یک چند جمله ای بر تعدادی نقطه می باشد که نقطه x که مقدار تابع باید در آن حساب شود، را در میان دارد. این چند جمله ای، تقریبی برای تابع است و برای f(x) بکار می رود.

چند جمله ای خط راست

$$f(x)=a_1+a_2x$$

$$f(x_1)=a_1+a_2x_1$$

$$f(x_2)=a_1+a_2x_2$$

$$\Rightarrow \begin{cases} a_2=\dfrac{[f(x_2)-f(x_1)]}{x_2-x_1} \\ a_1=f(x_1)-x_2a_2 \end{cases}$$

مثال

x	0.1	0.2	0.3	0.4
f(x)=coshx	1.005	1.020	1.045	1.081

f(x=0.16)=?

مقدار دقیق تابع:

f(x=0.16)= cosh(0.16)=1.0128

$$a_2 = \frac{1.52 - 1.05}{0.2 - 0.1} = \frac{0.15}{0.1} = 0.15$$

$$a_1 = 1.005 - 0.15 \times 0.1 = 0.99$$

$$\Rightarrow \hat{y} = f(x) = 0.99 + 0.15x$$

$$\hat{y}(0.16) = 0.99 + 0.15(0.16) = 1.014$$

$$\text{خطای نسبی} = \left| \frac{\text{مقدار دقیق} - \text{مقدار محاسبه شده}}{\text{مقدار دقیق}} \right|$$

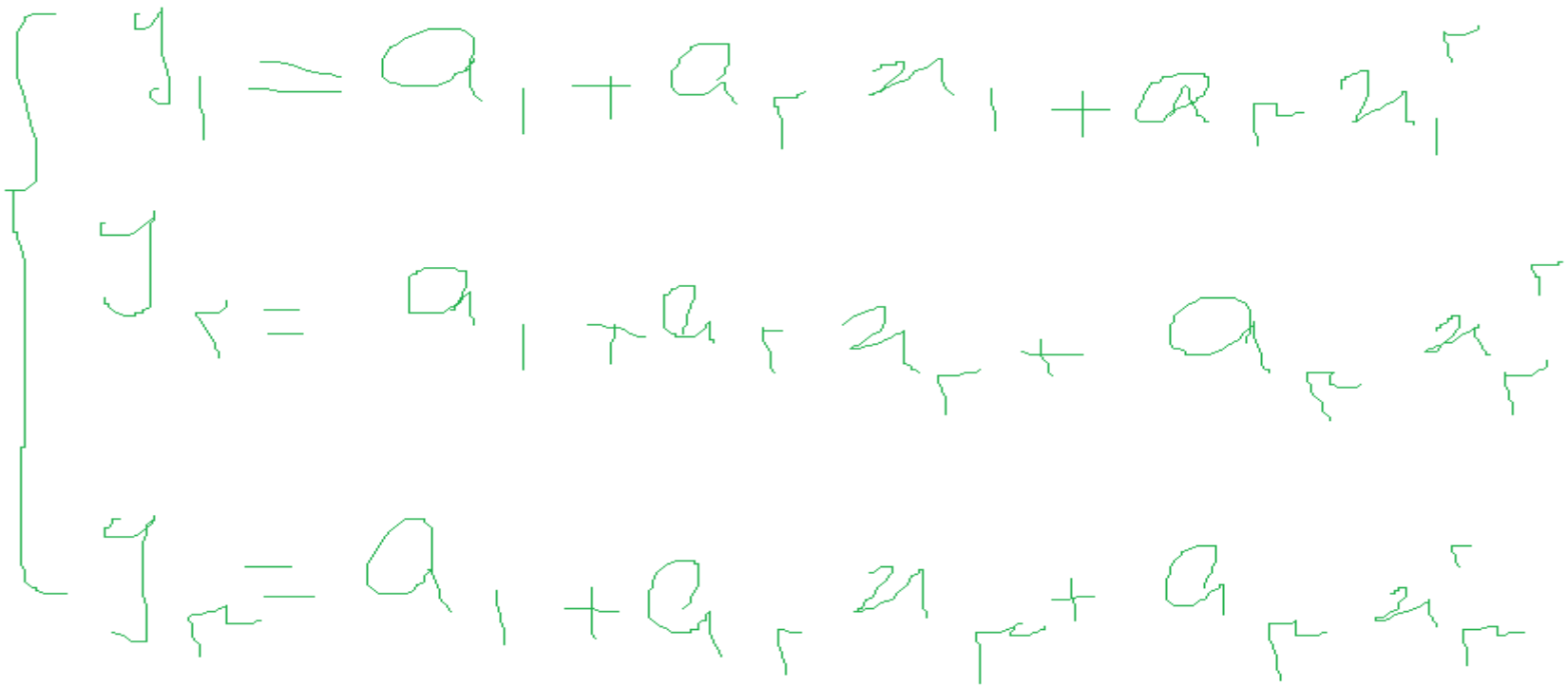
$$= \left| \frac{۱,۰۲۸ - ۱,۰۱۴}{۱,۰۲۸} \right|$$

$$= \frac{۰/۰۱۴}{۱,۰۲۸} \simeq ۰,۰۱۳۶$$

$$\text{خطای مطلق} = |\text{مقدار دقیق} - \text{مقدار محاسبه شده}|$$

$$= ۰,۰۰۵۱۲$$

چند جمله ای درجه دوم



چند جمله ای بالا مثلثی (روش نیوتن-گریگوری)

$$y = b_1 + b_2(x - x_1) + b_3(x - x_1)(x - x_2)$$

$$\begin{cases} y_1 = b_1 \\ y_2 = b_1 + b_2(x_2 - x_1) \\ y_3 = b_1 + b_2(x_3 - x_1) + b_3(x_3 - x_1)(x_3 - x_2) \end{cases}$$

$$b_1 = y_1$$

$$b_2 = \frac{y_2 - y_1}{x_2 - x_1}$$

$$b_3 = \frac{[y_3 - y_1] - \left[\frac{(y_2 - y_1)(x_3 - x_1)}{x_2 - x_1}\right]}{(x_3 - x_1)(x_3 - x_2)}$$

$$= \frac{y_3 - y_1}{(x_3 - x_1)(x_3 - x_2)} - \frac{y_2 - y_1}{(x_2 - x_1)(x_3 - x_1)}$$

مثال:

x	0.1	0.2	0.3	0.4
f(x)=coshx	1.005	1.020	1.045	1.081

f(x=0.16)=?

مقدار دقیق تابع:

f(x=0.16)= cosh(0.16)=1.0128

$$b_1 = ۱٫۰۰۵$$

$$b_2 = ۰٫۰۱۵ / ۰٫۱ = ۰٫۱۵$$

$$b_3 = (۰٫۰۲۵ - ۰٫۰۱۵) / ۰٫۰۲ = ۰٫۵$$

$$\text{Cosh}\,(۰٫۱۶) = ۱٫۰۰۵ + ۰٫۱۵(۰٫۰۶) + ۰٫۵(۰٫۰۶)(-۰٫۰۴) = ۱٫۰۱۲۸$$

چند جمله ای لاگرانژ

$$f(x) = c_1(x - x_2)(x - x_3) + c_2(x - x_1)(x - x_3) + c_3(x - x_1)(x - x_2)$$

$$f(x_1) = c_1(x_1 - x_2)(x_1 - x_3)$$

$$f(x_2) = c_2(x_2 - x_1)(x_2 - x_3)$$

$$f(x_3) = c_3(x_3 - x_1)(x_3 - x_2)$$

$$c_1 = \frac{f(x_1)}{(x_1 - x_2)(x_1 - x_3)}$$

$$c_2 = \frac{f(x_2)}{(x_2 - x_1)(x_2 - x_3)}$$

$$c_3 = \frac{f(x_3)}{(x_3 - x_1)(x_3 - x_2)}$$

$$\frac{f(x_1)(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} + \frac{f(x_2)(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} + \frac{f(x_3)(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)}$$

$$f(x) = \sum_{i=1}^{3} f(x_i) \prod_{\substack{j \neq i \\ j=1}}^{3} \frac{(x - x_j)}{(x_i - x_j)}$$

- چند جمله ای لاگرانژ را می توان به شکل زیر برای مرتبه n ام تعمیم داد:

$$f(x)\sum_{i=1}^{n+1} f(x_i)\prod_{\substack{j=1\\ j\neq i}}^{n+1}\frac{(x-x_j)}{(x_i-x_j)}$$

مثال:

$$f(x=4)=?$$

x	1.5	3	6
F(x)	-0.25	2	20

$$f(x)=\frac{-۰٫۲۵(x-۳)(x-۶)}{(۱٫۵-۳)(۱٫۵-۶)}+\frac{۲(x-۱٫۵)(x-۶)}{(۳-۱٫۵)(۳-۶)}$$

$$+\frac{۲۰(x-۱٫۵)(x-۳)}{(۶-۱٫۵)(۶-۳)}$$

$$f(۴)=\frac{-۰٫۲۵(۱)(-۲)}{(-۱٫۵)(-۴٫۵)}+\frac{۲(+۲٫۵)(-۲)}{(۱٫۵)(-۳)}+\frac{۲۰(۲٫۵)(۱)}{(۴٫۵)۳}$$

$$=۰٫۰۷۴+۲٫۲۲۲+۳٫۷۰۳$$

$$=۵٫۹۹۹=۶ \quad \text{(گرد شده)}$$

روش نیوتن گریگوری

$$f(x) = b_1 + b_2(x - x_1) + b_3(x - x_1)(x - x_2)$$

جداول تفاضل

$f(x_1)$	$\Delta_d f_1 = b_2 = \dfrac{f(x_2) - f(x_1)}{x_2 - x_1}$	
$f(x_2)$		$\Delta_d^2 f_1 = b_3 = (\Delta_d f_2 - \Delta_d f_1)/(x_3 - x_1)$
$f(x_3)$	$\Delta_d f_2 = \dfrac{f(x_3) - f(x_2)}{x_3 - x_2}$	

$$f(x) = f(x_1) + \Delta_d f_1(x - x_1) + \Delta_d^2 f_1(x - x_1)(x - x_2)$$

• در حالت کلی برای برازش یک چند جمله مرتبه n-1 بر n نقطه مندرج در یک جدول تابعی:

$$f(x) = f(x_1) + \Delta_d f_1(x - x_1) + \Delta_d^2 f_1(x - x_1)(x - x_2)$$
$$+ \Delta_d^3 f_1(x - x_1)(x - x_2)(x - x_3) + \ldots$$
$$+ \Delta_d^{n-1} f_1(x - x_1)(x - x_2)\ldots(x - x_{n-1})$$

که در آن

$$\Delta_d^n f_1 = (\Delta_d^{n-1} f_n - \Delta_d^{n-1} f_{n-1}) / (x_n - x_1)$$

مثال:

$$f(x=2.5)=?$$

x	-3	-1	0	3	5
F(x)	-30	-22	-12	330	3458

x	$f(x)$	$\Delta_d f$	$\Delta_d^2 f$	$\Delta_d^3 f$	$\Delta_d^4 f$
-۳	-۳۰				
		۴			
-۱	-۲۲		۲		
		۱۰		۴	
۰	-۱۲		۲۶		
		۱۱۴		۴۴	۵
۳	۳۳۰		۲۹۰		
		۱۵۶۴			
۵	۳۴۵۸				

بنابراین چندجمله‌ای عبارت است از :

$$f(x) = -۳۰ + ۴(x+۳) + ۲(x+۳)(x+۱) + ۴(x+۳)(x+۱)x$$

$$+ ۵(x+۳)(x+۱)(x-۳)$$

- اگر توابع در بازه های مساوی h جدول بندی شوند یعنی

$$(x_2 - x_1) = (x_3 - x_2) = \ldots = (x_n - x_{n-1}) = h$$

آنگاه

$$f(x) = b_1 + b_2(x - x_1) + b_3(x - x_1)(x - x_2)\ldots + b_n(x - x_1)\ldots(x - x_n)$$

$$= f_1 + \frac{\Delta f_1}{h}(x - x_1) + \frac{\Delta^2 f_1}{2!h^2}(x - x_1)(x - x_2) + \frac{\Delta^3 f_1}{3!h^3}(x - x_1)(x - x_2)(x - x_3) + \ldots + \frac{\Delta^n f_1}{n!h^n}(x - x_1)(x - x_2)\ldots(x - x_n)$$

x_1	f_1				
		$\Delta f_1 = f_2 - f_1$			
$x_1 + h$	f_2		$\Delta^2 f_1 = \Delta f_2 - \Delta f_1$		
		$\Delta f_2 = f_3 - f_2$		$\Delta^3 f_1 = \Delta^2 f_2 - \Delta^2 f_1$	
$x_1 + 2h$	f_3		$\Delta^2 f_2 = \Delta f_3 - \Delta f_2$		$\Delta^4 f_1 = \Delta^3 f_2 - \Delta^3 f_1$
		$\Delta f_3 = f_4 - f_3$		$\Delta^3 f_2 = \Delta^2 f_3 - \Delta^2 f_2$	
$x_1 + 3h$	f_4		$\Delta^2 f_3 = \Delta f_4 - \Delta f_3$		
		$\Delta f_4 = f_5 - f_4$			
⋮	⋮				
$x_1 + nh$	f_{n+1}				

$$x = x_1 + uh$$

$$x - x_1 = hu$$
$$x - x_2 = x_1 + hu - x_1 - h = h(u-1)$$
$$x - x_3 = x - (x_2 + h) = h(u-1) - h = h(u-2)$$
$$\vdots$$
$$x - x_n = h(u - \overline{n-1})$$

$$f(x_1 + uh) = f_1 + \Delta f_1 u + \frac{\Delta^2 f_1}{2!} u(u-1) + \frac{\Delta^3 f_1}{3!} u(u-1)(u-2)$$

$$+ \ldots + \frac{\Delta^n f_1}{n!} u(u-1)(u-2)\ldots(u - \overline{n-1})$$

- که u عددی بین 0 و 1 است. چند جمله ای اسلاید قبل به فرمول درون یابی رو به جلو نیوتن-گریگوری معروف است.

مثال

x	0.1	0.2	0.3	0.4
f(x)=coshx	1.005	1.020	1.045	1.081

f(x=0.16)=?

مقدار دقیق تابع:

f(x=0.16)= cosh(0.16)=1.0128

x	$f(x)$			
۰٫۱	۱٫۰۰۵			
		۰٫۰۱۵		
۰٫۲	۱٫۰۲۰		۰٫۰۱۰	
		۰٫۰۲۵		۰٫۰۰۱
۰٫۳	۱٫۰۴۵		۰٫۰۱۱	
		۰٫۰۳۶		
۰٫۴	۰٫۰۸۱			

$$f(۰٫۱۶) = f(۰٫۱ + ۰٫۰۶) = f[۰٫۱ + (۰٫۶)(۰٫۱)]$$

$$f(۰٫۱۶) = ۱٫۰۰۵ + ۰٫۶(۰٫۰۱۵) + \frac{۰٫۰۱۰}{۲} ۰٫۶(۰٫۶ - ۱) + \frac{۰٫۰۰۱}{۶}$$
$$۰٫۶(۰٫۶ - ۱)(۰٫۶ - ۲)$$
$$= ۱٫۰۰۵ + ۰٫۰۰۹۰ - ۰٫۰۰۱۲ + ۰٫۰۰۰۰۵۶ \approx ۱٫۰۱۲۸$$

خطای انباشته ناشی از خطای گردن کردن e

+ e				
	-۲e			
- e		۴e		
	۲e		-۸e	
+ e		-۴e		۱۶ e
	-۲e		۸e	
- e		۴e		-۱۶ e
	۲e		-۸e	
+ e		-۴e		
	-۲e			
- e				

خطای انباشته ناشی از یک خطای اولیه e

۰				
	۰			
۰		۰		
	۰		e	
۰		e		-۴e
	e		-۳e	
e		-۲e		۶e
	-e		۳e	
۰		e		-۴e
	۰		-e	
۰		۰		
	۰			
۰				

۱۰e
-۱۰e

جدول تفاضل رو به جلو

x_1	f_1		
		Δf_1	
$x_1 + h$	f_2		$\Delta^2 f_1$
		Δf_2	
$x_1 + 2h$	f_3		$\Delta^2 f_2$
		Δf_3	
$x_1 + 3h$	f_4		$\Delta^2 f_3$
		Δf_4	
$x_1 + 4h$	f_5		$\Delta^2 f_4$
		Δf_5	
$x_1 + 5h$	f_6		

جدول تفاضل رو به عقب

x_1	f_1			
		∇f_2		
$x_1 + h$	f_2		$\nabla^2 f_3$	
		∇f_3		$\nabla^3 f_4$
$x_1 + 2h$	f_3		$\nabla^2 f_4$	
		∇f_4		$\nabla^3 f_5$
$x_1 + 3h$	f_4		$\nabla^2 f_5$	
		∇f_5		$\nabla^3 f_6$
$x_1 + 4h$	f_5		$\nabla^2 f_6$	
		∇f_6		
$x_1 + 5h$	f_6			

روش کمترین مربعات یا رگرسیون

- برازش کردن تابعی بر مجموعه ای از داده ها که دارای خطا هستند.

رگرسیون خطی

$$y = a_1 x + a_0$$

$$S = \sum_i (y_i - \bar{y}_i)^2 = \sum_i [y_i - a_1 x_i - a_0]^2$$

$$\min S = \min\left(\sum_i [y_i - a_1 x_i - a_0]^2\right)^2$$

$$\frac{\partial S}{\partial a_0} = \sum_i 2(y_i - a_1 x_i - a_0)(-1) = 0$$

$$\rightarrow \sum y_i - a_1 \sum x_i - n a_0 = 0$$

$$\frac{\partial S}{\partial a_0} = \sum_i 2(y_i - a_1 x_i - a_0)(-x_i) = 0$$

$$\rightarrow \sum x_i y_i + a_1 \sum x_i^2 + a_0 \sum x_i = 0$$

$$na_0 + \left(\sum x_i\right) a_1 = \sum y_i$$

$$\left(\sum x_i\right) a_0 + \left(\sum x_i^2\right) a_1 = \sum x_i y_i$$

$$a_0 = \frac{\sum y_i \sum x_i^2 - \sum x_i \sum x_i y_i}{n \sum x_i^2 - (\sum x_i)^2}$$

$$a_1 = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum x_i^2 - (\sum x_i)^2}$$

متغیر مستقل	متغیر وابسته			
x	y	x^2	xy	
۱	۲	۱	۲	
۲	۵	۴	۱۰	
۴	۷	۱۶	۲۸	
۵	۱۰	۲۵	۵۰	
۶	۱۲	۳۶	۷۲	
۸	۱۵	۶۴	۱۲۰	
۹	۱۹	۸۱	۱۷۱	
۳۵	۷۰	۲۲۷	۴۵۳	جمع‌ها

$$a_0 = \frac{۷۰ \times ۲۲۷ - ۳۵ \times ۴۵۳}{۷ \times ۲۲۷ - (۳۵)^2} = \frac{۳۵}{۳۶۴} = \frac{۵}{۵۲} = ۰٫۰۹۶$$

$$a_1 = \frac{۷ \times ۴۵۳ - ۳۵ \times ۷۰}{۷ \times ۲۲۷ - (۳۵)^2} = \frac{۷۲۱}{۳۶۴} = \frac{۱۰۳}{۵۲} = ۱٫۹۸$$

$$y = ۱٫۹۸x + ۰٫۰۹۶$$

رگرسیون نمایی

$$y = ae^{-bx}$$

با تبدیل زیر داریم

$$z = \ln y = \ln ae^{-bx} = \ln a + (-bx)$$

با مقایسه با معادله خطی

$$a_0 = \ln a$$
$$a_1 = -b$$

بنابراین

$$z = a_0 + a_1 x$$

$$na_0 + \left(\sum x_i\right) a_1 = \sum \ln y_i$$

$$\left(\sum x_i\right) a_0 + \left(\sum x_i^2\right) a_1 = \sum x_i \ln y_i$$

$$a_0 = \frac{\sum \ln y_i \sum x_i^2 - \sum x_i \sum x_i \ln y_i}{n \sum x_i^2 - (\sum x_i)^2}$$

$$a_1 = \frac{n \sum x_i \ln y_i - \sum x_i \sum \ln y_i}{n \sum x_i^2 - (\sum x_i)^2}$$

بنابراین

$$a = e^{a_0}$$
$$b = -a_1$$

درون یابی

درون یابی تک متغیره

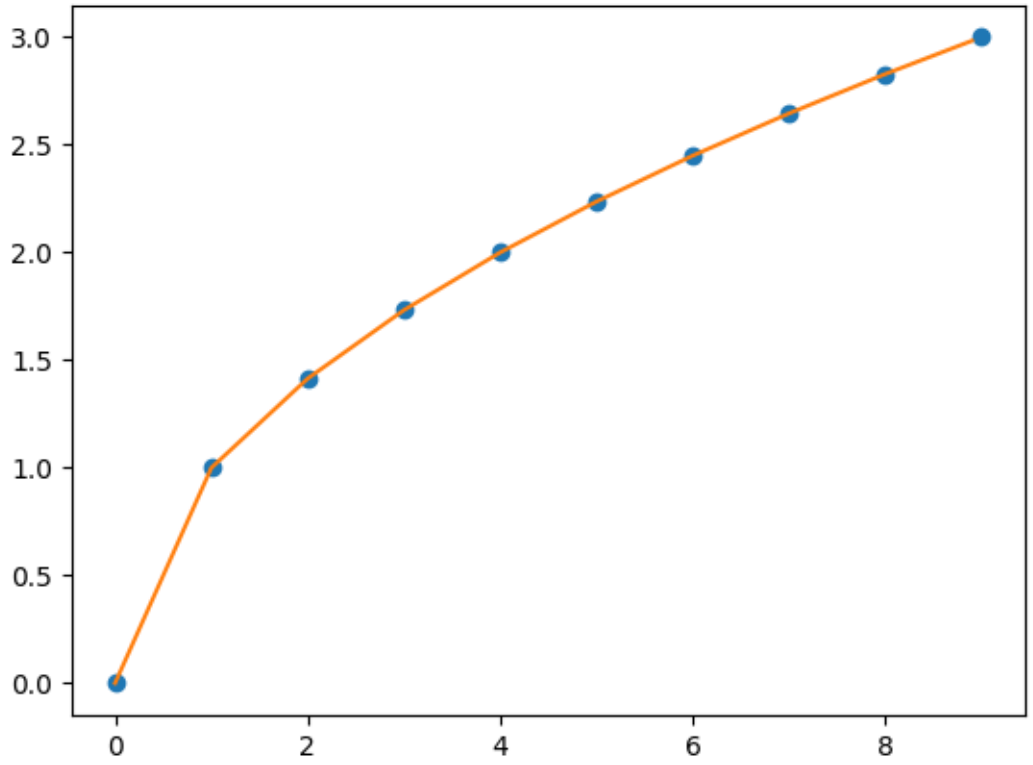
- interp1d(x, y, kind=‘method‘)
- Specifies the kind of interpolation as a string or as an integer specifying the order of the spline interpolator to use. The string has to be one of ‘linear’, ‘nearest’, ‘nearest-up’, ‘zero’, ‘slinear’, ‘quadratic’, ‘cubic’, ‘previous’, or ‘next’. ‘zero’, ‘slinear’, ‘quadratic’ and ‘cubic’ refer to a spline interpolation of zeroth, first, second or third order; ‘previous’ and ‘next’ simply return the previous or next value of the point; ‘nearest-up’ and ‘nearest’ differ when interpolating half-integers (e.g. 0.5, 1.5) in that ‘nearest-up’ rounds up and ‘nearest’ rounds down. Default is ‘linear’.

```
from scipy.interpolate import interp1d
import numpy as np

xs = np.array([0,1,2,3,4,5,6,7,8,9])
ys = np.array([1,3,5,7,9,11,13,15,17,19])

interp_func = interp1d(xs, ys)

newarr = interp_func(np.arange(2.1, 3, 0.1))

print(newarr)
```

```
from scipy.interpolate import UnivariateSpline
import numpy as np

xs = np.array([0,1,2,3,4,5,6,7,8,9])
ys = np.array([1,2.84,5.90,10.14,16.24,25.04,36.72,50.65,65.98,82.41])

interp_func = UnivariateSpline(xs, ys)

newarr = interp_func(np.arange(2.1, 3, 0.1))

print(newarr)
```

```
from scipy.interpolate import Rbf
import numpy as np
xs = np.array([0,1,2,3,4,5,6,7,8,9])
ys= np.array([1,2.84,5.90,10.14,16.24,25.04,36.72,50.65,65.98,82.41])

interp_func = Rbf(xs, ys)

newarr = interp_func(np.arange(2.1, 3, 0.1))

print(newarr)
```

```
import matplotlib.pyplot as plt
from scipy import interpolate
import numpy as np
x = np.array([0,1,2,3,4,5,6,7,8,9])
y = np.array([0,1,1.41,1.73,2,2.23,2.44,2

f = interpolate.interp1d(x, y)
xnew = np.arange(0, 9, 0.1)
ynew = f(xnew)
plt.plot(x, y, 'o', xnew, ynew, '-')
plt.show()
```



فیت کردن تابع روی دیتا

scipy.optimize.curve_fit(f, xdata, ydata)

روش کمترین مربعات غیر خطی

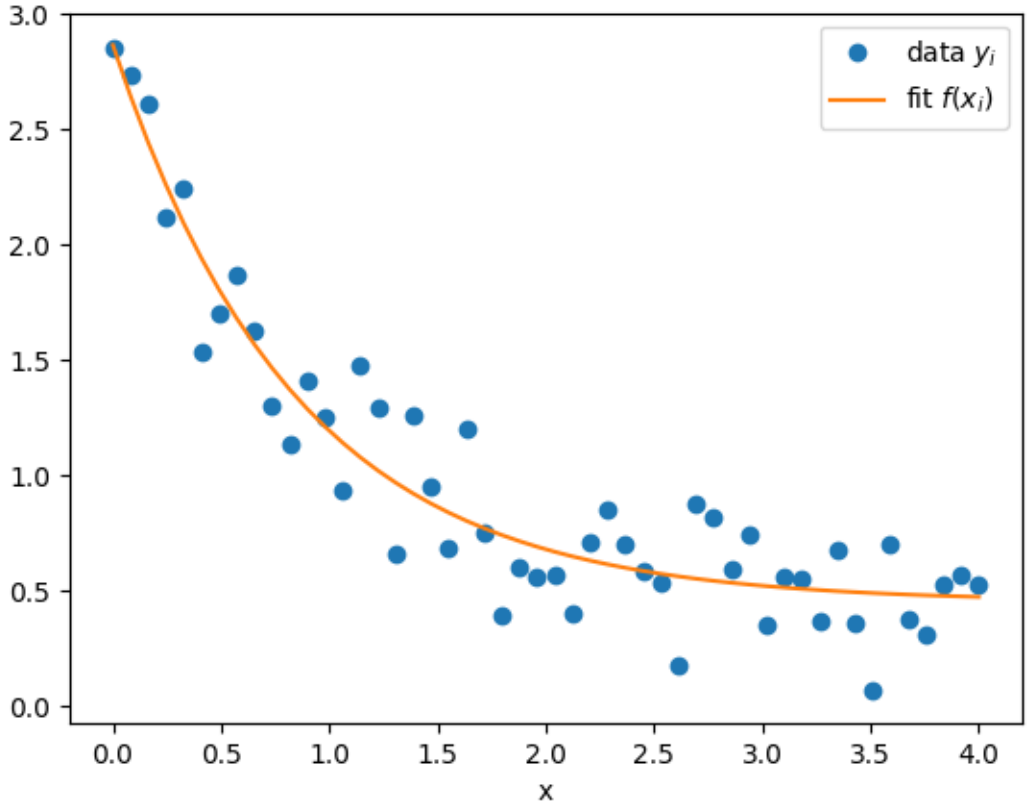
popt: Optimal values for the parameters

pcov:The estimated covariance of popt

مثال

$$y = ae^{-bx} + c$$

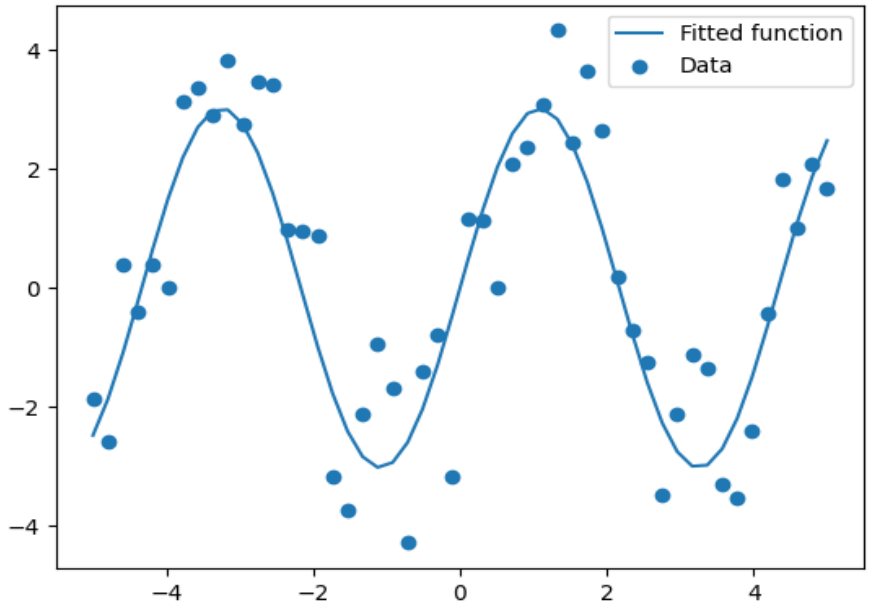
```
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

def f(x, a, b, c):
    return a * np.exp(- b * x) + c
x = np.linspace(0, 4, 50)
y = f(x, a=2.5, b=1.3, c=0.5)
yi = y + 0.2 * np.random.normal(size=len(x))

popt, pcov = curve_fit(f, x, yi)
a, b, c = popt
print(a, b, c)
yfitted = f(x, popt[0], popt[1], popt[2]) # or f(x,*pop)
plt.plot(x, yi, 'o', label='data ')
plt.plot(x, yfitted, '-', label='fit ')
plt.xlabel('x')
plt.legend()
plt.show()
```



مثال

$y=a \sin(bx)$

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import optimize
x_data = np.linspace(-5, 5, num=50)
y_data = 2.9 * np.sin(1.5 * x_data) + np.random.normal(size=50)
def f(x, a, b):
    return a * np.sin(b * x)
params, params_covariance = optimize.curve_fit(f, x_data, y_data)
print(params)
plt.scatter(x_data, y_data, label='Data')
plt.plot(x_data, f(x_data, params[0], params[1]),label='Fitted function')
plt.legend()
plt.show()
```



تمرین

- برازش هذلولی

کدی بنویسید که تابع زیر را بتوانید روی مجموعه ای از داده ها برازش کنید

$$y = \frac{1}{a + bx}$$

- برازش یک تابع مثلثاتی

کدی بنویسید که تابع زیر را بتوانید روی مجموعه ای از داده ها برازش کنید

$$y = Asin(\omega x + \varphi)$$

- کدی بنویسید که خم زیر را بتوانید روی مجموعه ای از داده ها برازش کنید

$$y = ax^b + c$$