curve_fit

April 28, 2021

[1]:
```
import numpy as np
import scipy.optimize as opt
import matplotlib.pyplot as plt
```

0.1 Using scipy

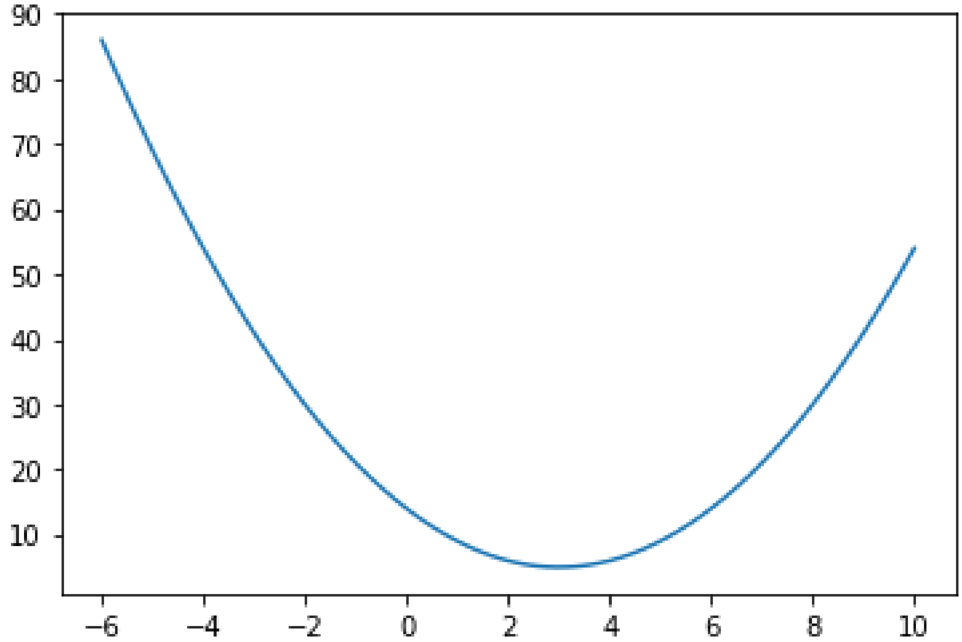
0.1.1 scipy.optimize.minimize

0.1.2 Example

[2]:
```
#Create function to be minimized
def parabola(x):
    return (x-3)**2 + 5

xplot = np.linspace(-6,10,1000)
yplot = parabola(xplot)
plt.plot(xplot,yplot)
```

[2]:
```
[<matplotlib.lines.Line2D at 0x7faa8d7f0290>]
```



[3]:
```
result = opt.minimize(parabola,[0])
print(result)
```

```
      fun: 5.000000000000001
 hess_inv: array([[0.5]])
      jac: array([5.96046448e-08])
  message: 'Optimization terminated successfully.'
     nfev: 9
      nit: 2
     njev: 3
   status: 0
  success: True
        x: array([3.00000003])
```

[4]:
```
print("The function is minmized at x={}, f(x)={}".format(result.x[0],result.fun))
```

```
The function is minmized at x=3.0000000283269603, f(x)=5.000000000000001
```

[ ]:
```
```

[ ]:
```
```

[ ]:
```
```

0.2 Using the minimizer for curve fitting

[5]:
```
!cat datasets/current_vs_voltage.dat
```

```
voltage current
0.0 0.0
0.5 0.024622583018826805
1.0 0.030137467072485186
1.5 0.073597904594552
2.0 0.08889264504403813
2.5 0.12399147905759039
3.0 0.1441837430664787
3.5 0.11710445518081906
4.0 0.23149945668221925
4.5 0.2099161164814378
5.0 0.2804608109013857
5.5 0.29488924919434784
6.0 0.374167630589526
6.5 0.3153011630443429
7.0 0.2799094388098678
7.5 0.37675055900833854
8.0 0.3633889099166405
8.5 0.4965626299571659
9.0 0.4366777740972727
9.5 0.5557754992763733
10.0 0.36119424660252475
10.5 0.5787745657432957
11.0 0.5244313652567868
11.5 0.5353749218066816
12.0 0.5242314047198181
```

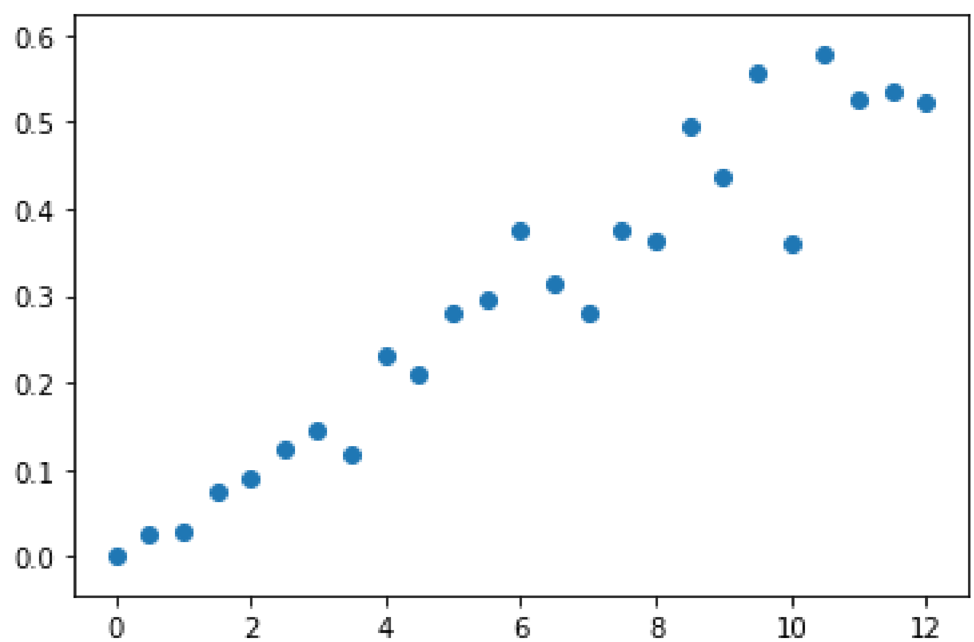
[6]:
```
data = np.genfromtxt("datasets/current_vs_voltage.dat",skip_header=1)
print(data)
```

```
[[ 0.          0.        ]
 [ 0.5         0.02462258]
 [ 1.          0.03013747]
 [ 1.5         0.0735979 ]
 [ 2.          0.08889265]
 [ 2.5         0.12399148]
 [ 3.          0.14418374]
 [ 3.5         0.11710446]
 [ 4.          0.23149946]
 [ 4.5         0.20991612]
 [ 5.          0.28046081]
 [ 5.5         0.29488925]
 [ 6.          0.37416763]
```

```
 [ 6.5         0.31530116]
 [ 7.          0.27990944]
 [ 7.5         0.37675056]
 [ 8.          0.36338891]
 [ 8.5         0.49656263]
 [ 9.          0.43667777]
 [ 9.5         0.5557755 ]
 [10.          0.36119425]
 [10.5         0.57877457]
 [11.          0.52443137]
 [11.5         0.53537492]
 [12.          0.5242314 ]]
```

[7]:
```
voltage = data[:,0]
current = data[:,1]
plt.scatter(voltage,current)
```

[7]:
```
<matplotlib.collections.PathCollection at 0x7faa8d95a510>
```



0.2.1 model:

$I = V/R$

[8]:
```
def residuals(R):
    s = 0
```

```python
    for i in range(voltage.size):
        imeas = current[i]
        ipred = voltage[i] / R
        s += (imeas - ipred)**2
    return s
```

[9]:
```python
result = opt.minimize(residuals,[10])
print("The residuals are minmized at R={}, S(R)={}".format(result.x[0],result.
 ↪fun))
```

```
The residuals are minmized at R=20.51906342539265, S(R)=0.05828806629427458
```

[ ]:
```python

```

[ ]:
```python

```

0.2.2 An easier (and better) way

scipy.optimize.curve_fit

[10]:
```python
def current_func(V,R):
    return V / R
```

[11]:
```python
popt,pcov = opt.curve_fit(current_func,voltage,current,absolute_sigma=0)
```

[12]:
```python
print("The best-fit value for R is: {} ohms".format(popt[0]))
```

```
The best-fit value for R is: 20.519067055737914 ohms
```

What is pcov?

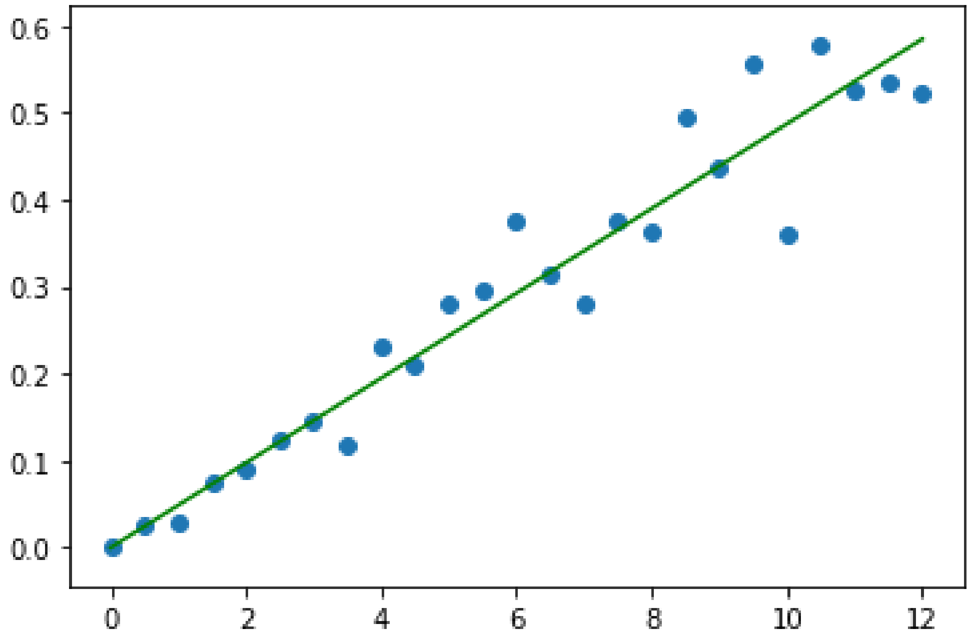
pcov is the *covariance matrix*

[ ]:
```python

```

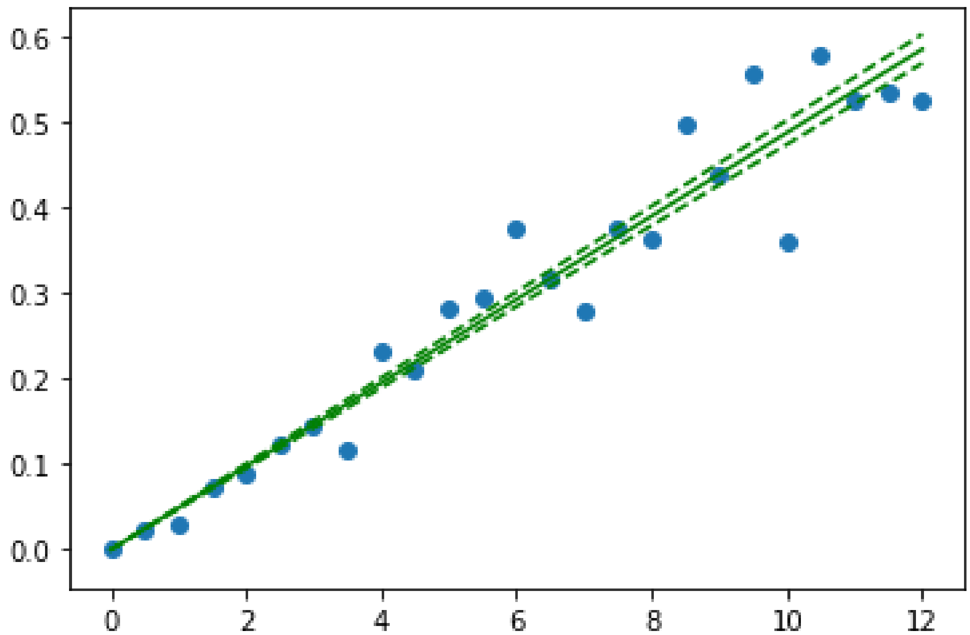
[13]:
```python
deltaR = np.sqrt( np.diag(pcov) )
vplot = np.linspace(voltage.min(),voltage.max(),1000)
ymean = current_func(vplot,popt[0])
plt.scatter(voltage,current)
plt.plot(vplot,ymean,c='green')
```

[13]:
```
[<matplotlib.lines.Line2D at 0x7faa8d95abd0>]
```



[14]:
```
deltaR = np.sqrt( np.diag(pcov) )
vplot = np.linspace(voltage.min(),voltage.max(),1000)
ymean = current_func(vplot,popt[0])
yupper = current_func(vplot,popt[0]-deltaR)
ylower = current_func(vplot,popt[0]+deltaR)
plt.scatter(voltage,current)
plt.plot(vplot,ymean,c='green')
plt.plot(vplot,ylower,ls='--',c='green')
plt.plot(vplot,yupper,ls='--',c='green')
```

[14]: [<matplotlib.lines.Line2D at 0x7faa8dad3790>]



[ ]:

```

```

[ ]:

```

```

0.2.3 Hubble's Constant

[19]:

```
!cat datasets/hubble.dat
```

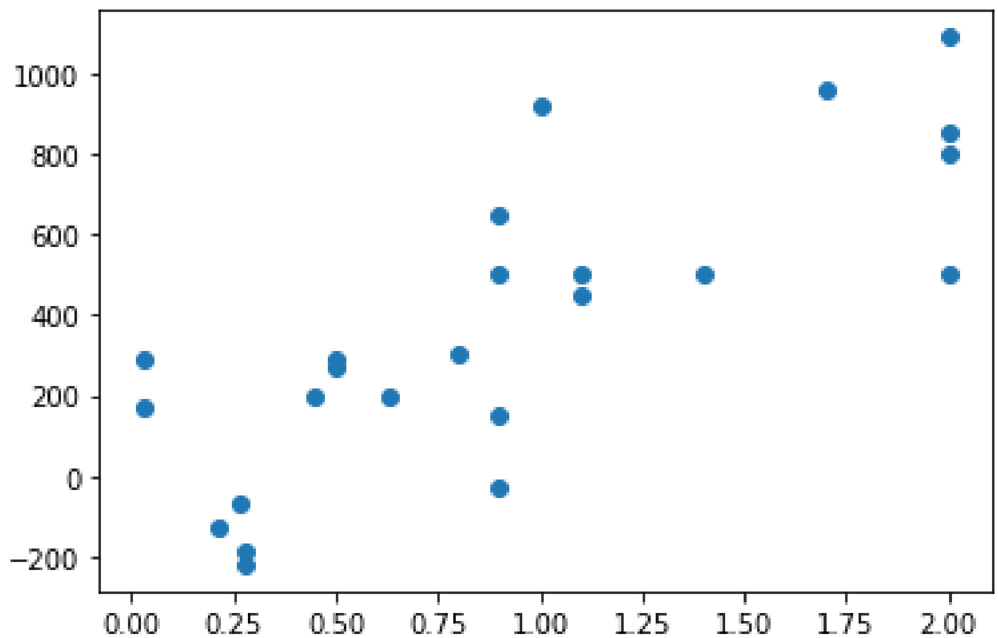
```
R (Mpc)  v (km/sec)
0.032    170
0.03     290
0.214    -130
0.263    -70
0.275    -185
0.275    -220
0.45     200
0.5      290
0.5      270
0.63     200
0.8      300
0.9      -30
0.9      650
0.9      150
0.9      500
1        920
```

```
1.1      450
1.1      500
1.4      500
1.7      960
2        500
2        850
2        800
2        1090
```

[20]:
```python
hubble_data = np.genfromtxt("datasets/hubble.dat",skip_header=1)
dist = hubble_data[:,0]
vel = hubble_data[:,1]
```

[21]:
```python
plt.scatter(dist,vel)
```

[21]: `<matplotlib.collections.PathCollection at 0x7faa8e994ad0>`



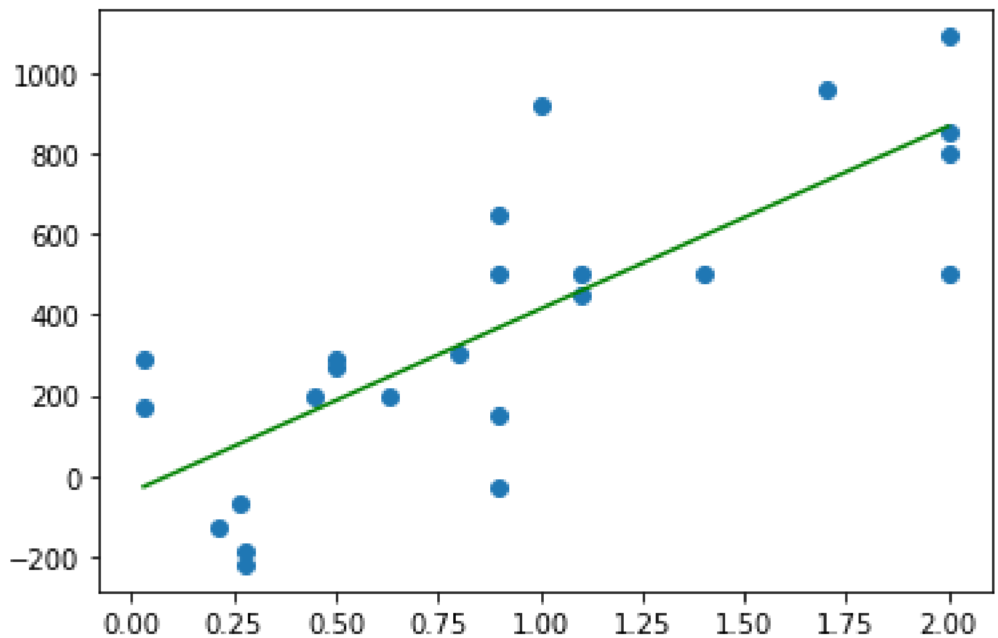
[28]:
```python
def linear_model(x,m,b):
    return m * x + b
```

[35]:
```python
popt,pcov = opt.curve_fit(linear_model,dist,vel,p0=[70,0])
uncert = np.sqrt(np.diag(pcov))
print("The best fit values are {:.2f}+-{:.2f}, {:.2f}+-{:.2f}".
 ↪format(popt[0],uncert[0],popt[1],uncert[1]))
```

```
The best fit values are 453.86+-75.25, -40.44+-83.45
```

[37]:
```
xplot = np.linspace(dist.min(),dist.max(),100)
ymean = linear_model(xplot,popt[0],popt[1])
plt.scatter(dist,vel)
plt.plot(xplot,ymean,c='green')
```

[37]:
```
[<matplotlib.lines.Line2D at 0x7faa8eea3a90>]
```



[ ]:
```

```

[ ]:
```

```

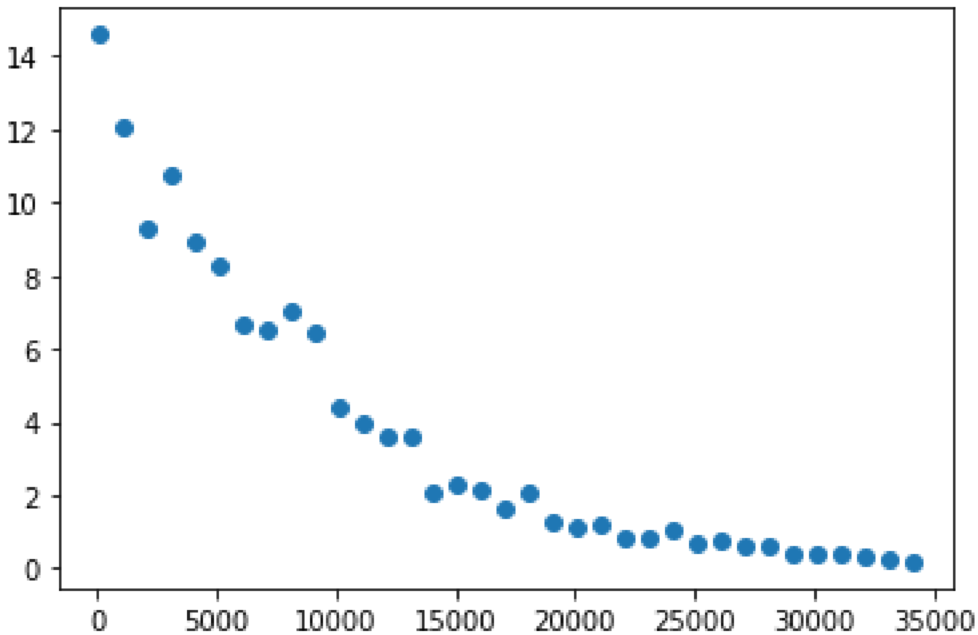
[ ]:
```

```

[64]:
```
data = np.genfromtxt("datasets/radioactive_decay.dat",skip_header=1)
time = data[:,0]
mass = data[:,1]
```

[65]:
```
plt.scatter(time,mass)
```

[65]:
```
<matplotlib.collections.PathCollection at 0x7faa90596fd0>
```



[52]:
```
def func_exp(t,A,tau):
    return A * np.exp(-t/tau)
```

[69]:
```
popt,pcov = opt.curve_fit(func_exp,time,mass,p0=[5,1000])
uncert = np.sqrt(np.diag(pcov))
```

[70]:
```
print("The best fit values are {:.2f}+-{:.2f}, {:.2f}+-{:.2f}".
 ↪format(popt[0],uncert[0],popt[1],uncert[1]))
```

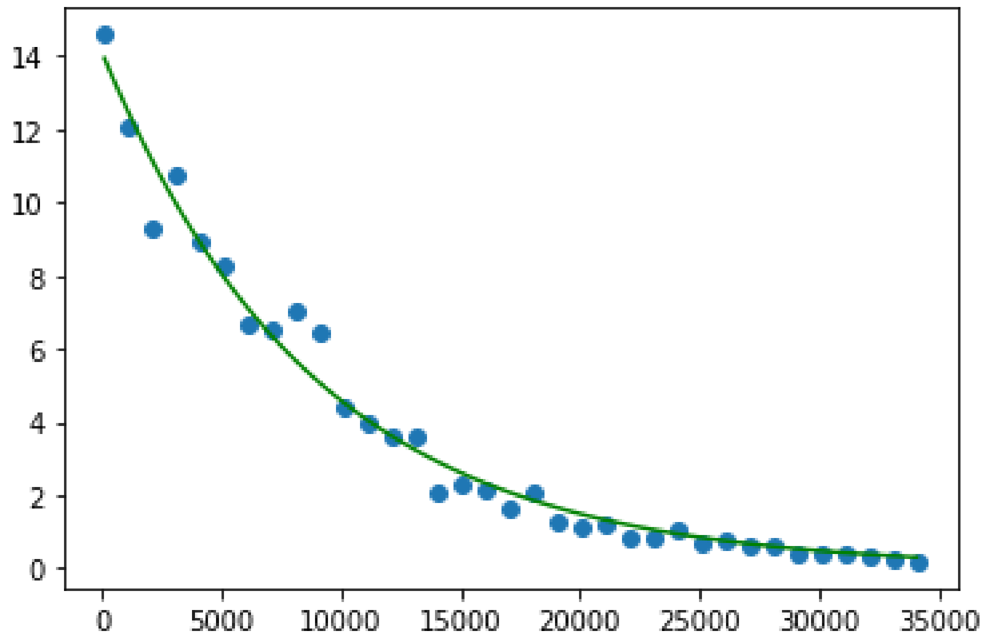
```
The best fit values are 14.09+-0.34, 8918.81+-323.72
```

[77]:
```
print("The half life is {:.2f}+-{:.2f}".format(popt[1]*np.log(2),uncert[1]*np.
 ↪log(2)))
print("The true value is: 5730")
```

```
The half life is 6182.05+-224.39
The true value is: 5730
```

[68]:
```
xplot = np.linspace(time.min(),time.max(),100)
yplot = func_exp(xplot,popt[0],popt[1])
plt.scatter(time,mass)
plt.plot(xplot,yplot,c='green')
```

[68]: `[<matplotlib.lines.Line2D at 0x7faa90708750>]`


14
12
10
8
6
4
2
0
0
5000
10000
15000
20000
25000
30000
35000

```
[ ]:
```