# Scipy questions

*August 2017*

1. Find all the zeroes of the polynomial $x^9 - 3x^6 + x^2 + 1$.

```
from scipy import roots
print(roots([1, 0, 0, -3, 0, 0, 0, 1, 0, 1]))
```

```
## [-0.75107001+1.23681826j -0.75107001-1.23681826j  1.36843381+0.j
##  -0.88577070+0.j          1.00000000+0.j         -0.30876102+0.74799231j
##  -0.30876102-0.74799231j  0.31849948+0.70729763j  0.31849948-0.70729763j]
```

2. Find a solution to this system of transcendental equations $xy = 2, \log x \log y = -\log 2$

```
from scipy.optimize import fsolve
from scipy import log
def func2(x):
    out = [x[0] * x[1] - 2, log(x[0]) * log(x[1]) + log(2)]
    return out

x = fsolve(func2,[4, 0.5])
print(x)
```

```
## [ 3.48470406  0.57393683]
```

3. Perform a linear least squares fit on the data set {(1, 2), (2, 3.5), (3, 6.5), (4, 7.8), (5, 11)}.

```
from scipy.optimize import leastsq
from numpy import random, array
def residuals(p, y, t):
    err = y - func(t, p)
    return err

def func(t, p):
    return p[0] * t + p[1]

t = array([1, 2, 3, 4, 5])
y = array([2, 3.5, 6.5, 7.8, 11])

p0 = [2, 0]
plsq = leastsq(residuals, p0, args = (y, t))
print(plsq)
```

```
## (array([ 2.23, -0.53]), 3)
```

4. The following data were collected for the harmonic oscillations of a vertically suspended spring-mass system. The time measurements were

```
[0.0,  0.1,  0.2,  0.3,  0.4,  0.5,  0.6, 0.7,  0.8,  0.9,  1.0,
1.1,  1.2,  1.3,  1.4,  1.5,  1.6,  1.7,  1.8,  1.9,  2.0]
```

and the corresponding displacements from equilibirum were

```
[1.78,  1.43, 1.12, 0.79,  0.30, -0.14, -0.62, -0.99, -1.30, -1.64, -1.91,
-1.97, -2.00, -1.89, -1.74,-1.44, -1.14, -0.78, -0.30, 0.12, 0.51]
```

The attached mass is 1. Find the spring constant k.

```
from pylab import *
from scipy.optimize import leastsq
from numpy import random

def residuals(p, y, t):
    err = y - func(t, p)
    return err

def func(t, p):
    return p[0] * cos(p[1] * t + p[2])

t = array([0.0, 0.1, 0.2, 0.3, 0.4, 0.5,
           0.6, 0.7, 0.8, 0.9, 1.0, 1.1,
           1.2, 1.3, 1.4, 1.5, 1.6, 1.7,
           1.8, 1.9, 2.0])

y = array([1.78, 1.43, 1.12, 0.79, 0.30,
           -0.14, -0.62, -0.99, -1.30, -1.64,
           -1.91, -1.97, -2.00, -1.89, -1.74,
           -1.44, -1.14, -0.78, -0.30,  0.12, 0.51])

p0 = [1, 2, pi/3]
plsq = leastsq(residuals, p0, args = (y, t))
print("k = ", plsq[0][1] ** 2)
```

```
## k =  5.00094267724
```

5. Write a program that will generate a normally distributed data sample of 100 numbers , with pdf mean $\overline{x} = 3$ and standard deviation $\sigma = 1$. Print out the mean, variance, and standard deviation of this data sample.

```
from scipy import std, mean, var, stats
data = stats.norm.rvs(loc=3,size=100)
print("Data sample: ")
print(data)
print("Standard deviation:", std(data))
print("Mean:", mean(data))
print("Variance:", var(data))
```

```
## Data sample:
## [ 2.72270244  2.58367356  0.6085725   2.60342002  1.93475291  3.33074614
##   4.33157223  0.52410416  0.57153292  2.30949038  1.8800606   1.28459692
##   2.50241489  4.67800456  2.2840444   3.28132084  3.99441152  3.81442547
##   2.90012861  4.13575849  1.64227761  2.80091461  4.55761288  4.66542676
##   0.80652099  2.49569026  2.52123886  3.10498849  1.36911996  3.21190663
##   2.63762856  3.2288875   3.72574778  3.16025179  2.51530597  2.07377444
##   3.40428085  2.07190312  2.40534356  4.45668445  0.82519571  3.77340293
##   3.15835159  3.67459803  3.60829649  3.7835559   3.67843417  2.0086701
##   0.62018755  5.13324369  4.10540814  2.90051543  3.50230233  3.00672109
##   3.00914764  1.45748638  3.23118589  5.1755227   3.51119352  3.88037983
##   4.03773407  0.88917409  2.9740995   0.76904     2.53910232  3.84930415
##   3.67502164  4.77086281  3.4959671   2.0425555   4.4603983   4.67022034
##   1.43171967  4.31721621  2.98763323  2.24026719  3.4813142   2.33481027
##   3.04005856  2.23201902  3.70142913  4.94845733  3.73049962  1.84466053
##   1.85512551  3.74018877  3.8266719   3.74706394  3.44202809  4.19847984
```

```
##   2.3503677   2.32782199  2.29863681  4.23106693  2.11620449  1.85151331
##   1.90883067  4.16629303  2.84023647  1.76822458]
## Standard deviation: 1.13052334346
## Mean: 2.94307356554
## Variance: 1.27808303011
```

6. Write a program that will generate a normally distributed data sample of 10000 numbers with pdf mean 0 and standard deviation 1. Print out what percentage of these numbers falls within 1, 2 and 3 standard deviations from the mean.

```
from scipy import std, mean, var, stats
data = stats.norm.rvs(loc=0,scale=1,size=10000)
count1 = 0
count2 = 0
count3 = 0
for number in data:
    if abs(number) <= 1:
        count1 += 1
    elif abs(number) <= 2:
        count2 += 1
    elif abs(number) <= 3:
        count3 += 1
print(count1/100,
      "% of the data fall within 1 standard deviation of the mean.")
print((count1 + count2)/100,
      "% of the data fall within 2 standard deviations of the mean.")
print((count1 + count2 + count3)/100,
      "% of the data fall within 3 standard deviations of the mean.")
```

```
## 68.17 % of the data fall within 1 standard deviation of the mean.
## 95.46 % of the data fall within 2 standard deviations of the mean.
## 99.72 % of the data fall within 3 standard deviations of the mean.
```

7. The position of a certain particle was measured at times

   `[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]`

   and the measurements were

   `[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]`

   Write a program that writes this data to a file called data.txt. The data should be formatted into two columns, the first being labelled 'time' and the second being labelled 'position'. Then write a program that will read the data from the file, store it in an array, and produce a position-time graph for the motion.

```
from pylab import plot, loadtxt
from rmdplot import savefig
t_array = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
x_array = [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
outfile = open('data.txt', 'w')
outfile.write('%10s %10s' % ('time', 'position'))
for i in range(0, len(t_array)-1):
    outfile.write('\n' + '%10.1f %10d' % (t_array[i], x_array[i]))
outfile.close()

data_array = loadtxt('data.txt', skiprows=1)
times = data_array[:, 0]
```

```
positions = data_array[:, 1]
plot(times, positions)
savefig("figure/scipy1.pdf","Question 7")
```

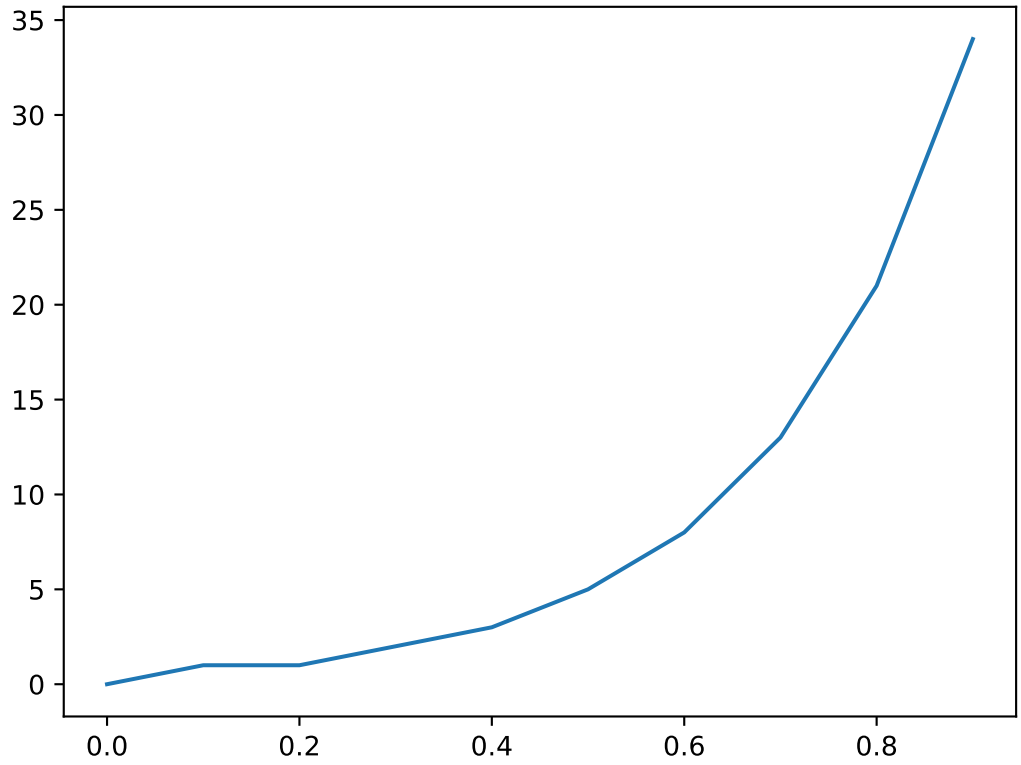


Figure 1: Question 7