Scuola universitaria professionale
della Svizzera italiana
SUPSI

Scientific Computing with Python

Kamini Garg
kamini.garg@supsi.ch

University of Applied Sciences and Arts of Southern Switzerland (SUPSI)

SciPy

- SciPy is a library of algorithms and mathematical tools built to work with NumPy arrays.
  - ➡ statistics - scipy.stats
  - ➡ optimization - scipy.optimize
  - ➡ signal processing - scipy.signal
  - ➡ etc….

SciPy Packages

`scipy.cluster`	Vector quantization / Kmeans
`scipy.constants`	Physical and mathematical constants
`scipy.fftpack`	Fourier transform
`scipy.integrate`	Integration routines
`scipy.interpolate`	Interpolation
`scipy.io`	Data input and output
`scipy.linalg`	Linear algebra routines
`scipy.ndimage`	n-dimensional image package
`scipy.odr`	Orthogonal distance regression
`scipy.optimize`	Optimization
`scipy.signal`	Signal processing
`scipy.sparse`	Sparse matrices
`scipy.spatial`	Spatial data structures and algorithms
`scipy.special`	Any special mathematical functions
`scipy.stats`	Statistics

SciPy Packages

- Scipy packages greatly depend on numpy

```
>>> import numpy as np
```

```
>>> from scipy import stats #import statistics
module
```

- The main scipy namespace mostly contains functions that are really numpy functions e.q. **scipy.cos is similar np.cos.**
- There's usually no reason to use import scipy in your code for numpy like methods.

Getting help

- You can go to official documentation website to know more about the packages http://docs.scipy.org/
- In addition to the help(), there is a scipy.info() function that outputs things.

Optimization

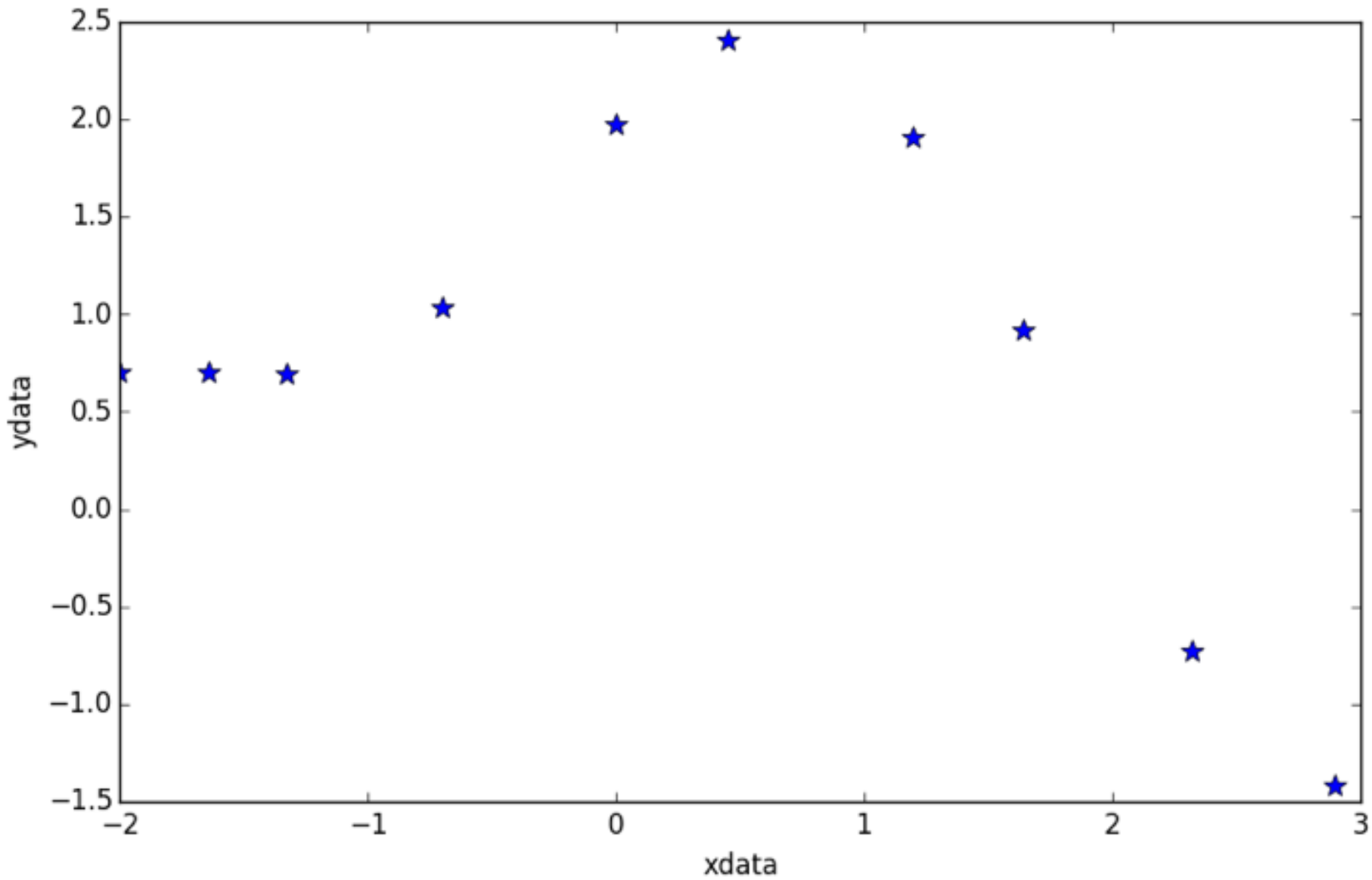
- The optimization package in SciPy allows us to solve minimization problems easily and quickly.
- Performing linear regression
- Finding a function’s minimum and maximum values
- Determining the root of a function
- Finding where two function will intersect

Data Modeling and Fitting

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

# Lets say we have following data

xdata = np.array([-2,-1.64,-1.33,-0.7,0,0.45,1.2,1.64,2.32,2.9])

ydata =
np.array([0.699369,0.700462,0.695354,1.03905,1.97389,2.41143,1.91091,0.919576,
-0.730975,-1.42001])

# Function to fit using nonlinear least squares
F(p1,p2,x) = p1 cos(p2x)+p2 sin(p1x)

# Goal
1.The fit parameters
2.Sum of squared residuals (error)
```

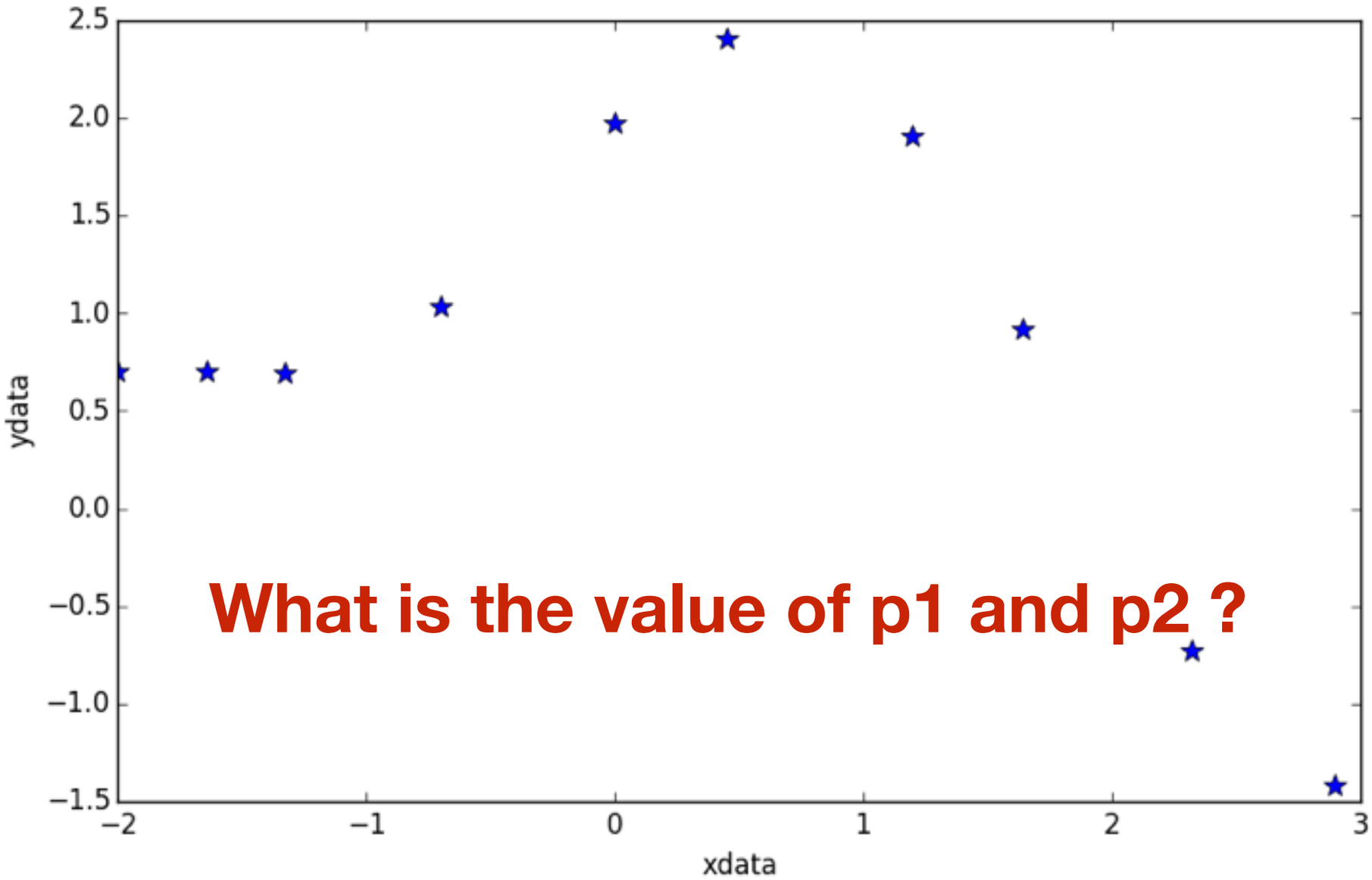
Data Modeling and Fitting

```
# Lets plot the data first
```



Data Modeling and Fitting

```
# Lets plot the data first
```



What is the value of p1 and p2 ?

Data Modeling and Fitting

```
from scipy.optimize import curve_fit


# define the function
def func(x, p1,p2):
    return p1*np.cos(p2*x) + p2*np.sin(p1*x)

# guess parameters Use a starting guess
p1=1
p2=0.2

# fit parameters
popt, pcov = curve_fit(func, xdata, ydata, p0=(1.0,0.2))

print(popt) # this variable stores parameter values

>>> [ 1.88185099  0.70022986]
```

Data Modeling and Fitting

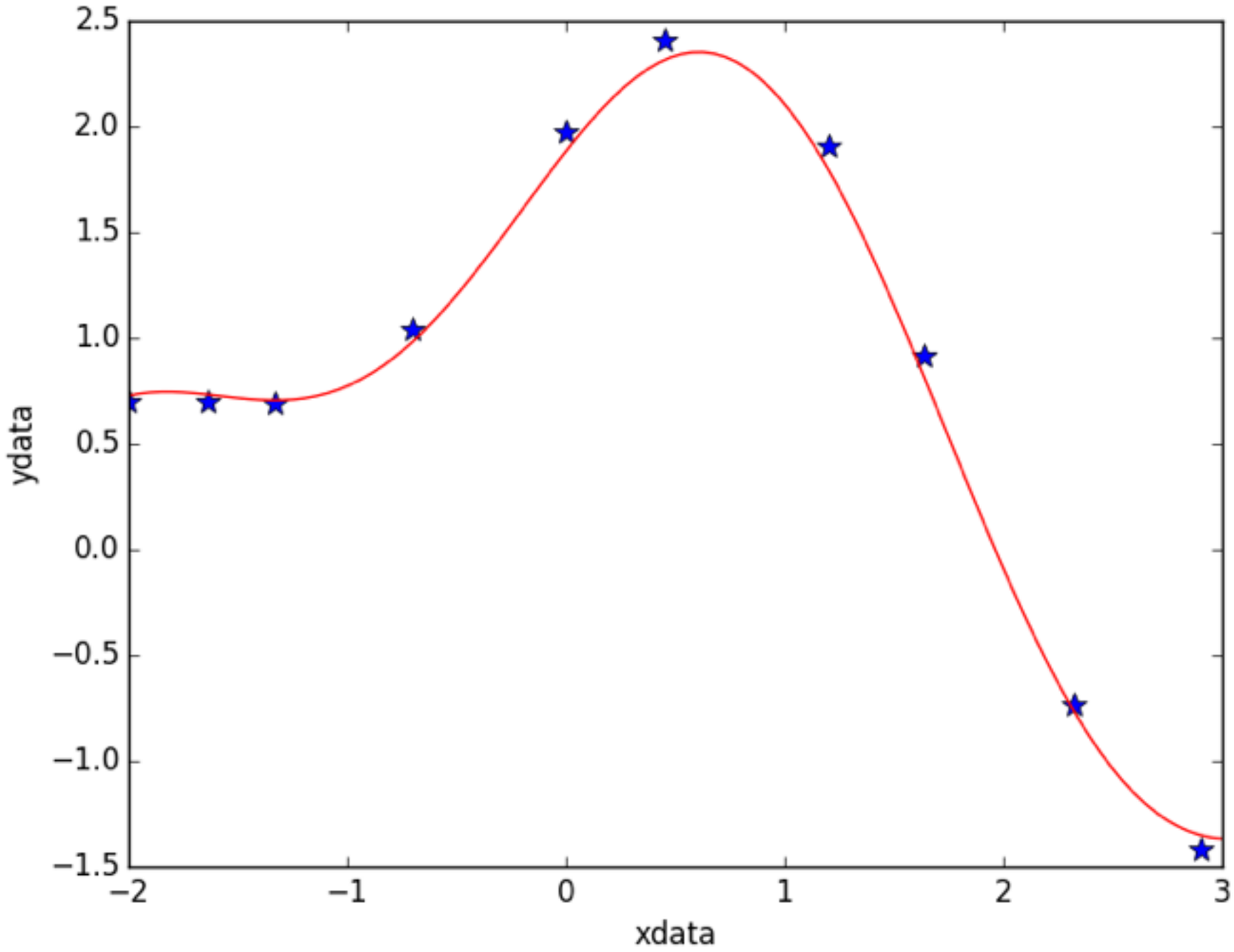
```
from scipy.optimize import curve_fit


# define the function
def func(x, p1,p2):
    return p1*np.cos(p2*x) + p2*np.sin(p1*x)

# guess parameters Use a starting guess
p1=1
p2=0.2

# fit parameters
popt, pcov = curve_fit(func, xdata, ydata, p0=(1.0,0.2))

print(popt) # this variable stores parameter values

>>> [ 1.88185099  0.70022986] # The fit parameters
```

Data Modeling and Fitting

```
# Now find the Sum of squared residuals or error
p1 = popt[0]
p2 = popt[1]
residuals = ydata - func(xdata,p1,p2)
res = sum(residuals**2)
print(res)
>>> 0.0538126964188
```

```
# plot the curve along fitted data for more points

curvex = np.linspace(-2,3,100) # create more data points in between
survey = func(curvex,p1,p2)
```

Data Modeling and Fitting



Assignment 8(a) (20 mins)

Taking this a step further, we can do a least-squares fit to a Gaussian profile, a non-linear function:

$$a * \exp\left(\frac{-(x-\mu)^2}{2\,\sigma^2}\right),$$

where a is a scalar, μ is the mean, and σ is the standard deviation.

Fit the above model by using 10 data points x between 1 to 10 and

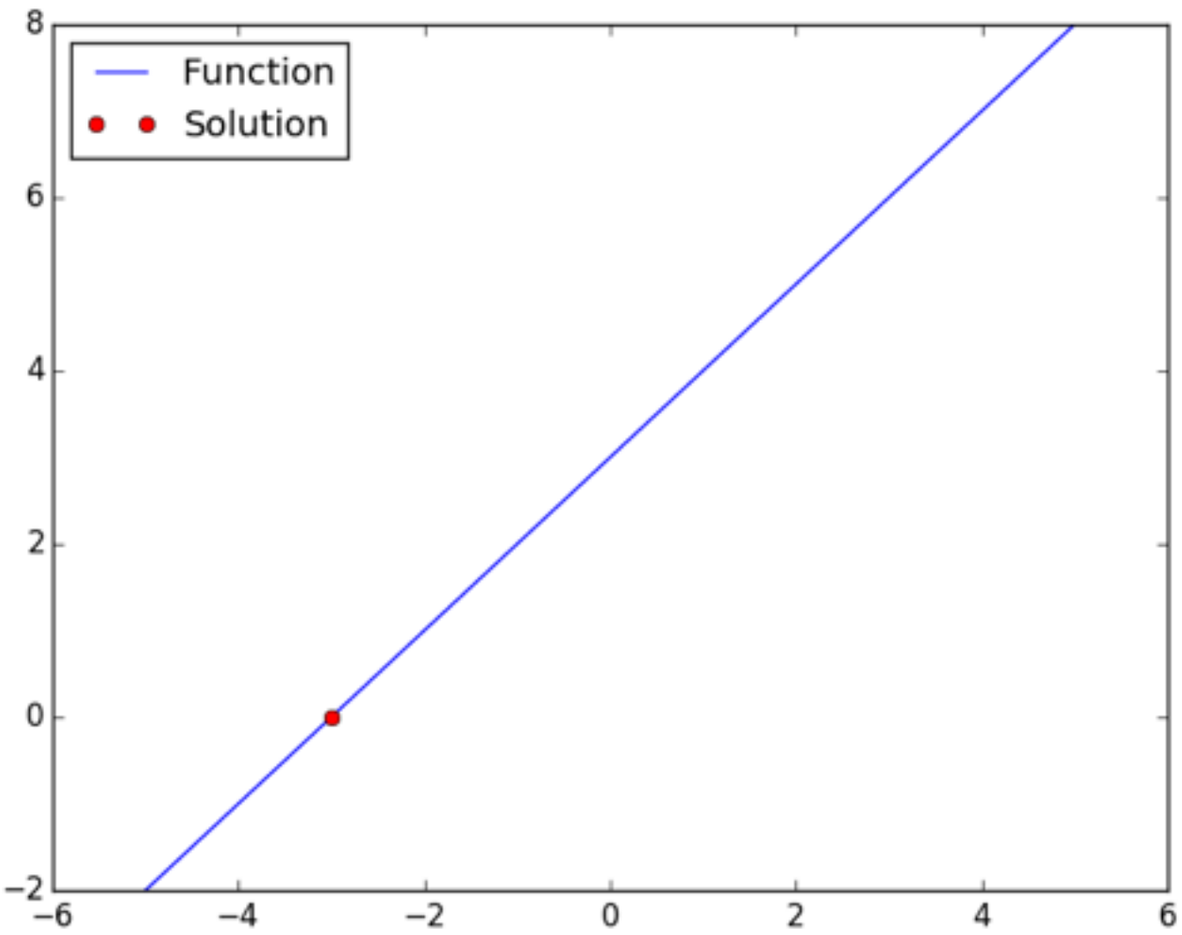
y = [-0.00983774 0.34473621 -0.00974954 0.69951548 0.5170367 1.23443728 1.0023327 0.44091547 0.1839785 0.16744486]

- Find parameters a, b, c (you can say mu as ‘b’ and sigma as ‘c’) **(2pt)**
- Find the *Sum of squared residuals* **(1pt)**
- Visualise your fit similar to previous model fit for 100 data points. **(2pt)**

1. File name : **as8a_yoursurname_name.py**
2. Upload your solution at Moodle under A08 folder after completion.

Solution to the functions

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

line = lambda x:x+3

solution = fsolve(line,-1) # starting
from x = -1
print(solution)
-3
```

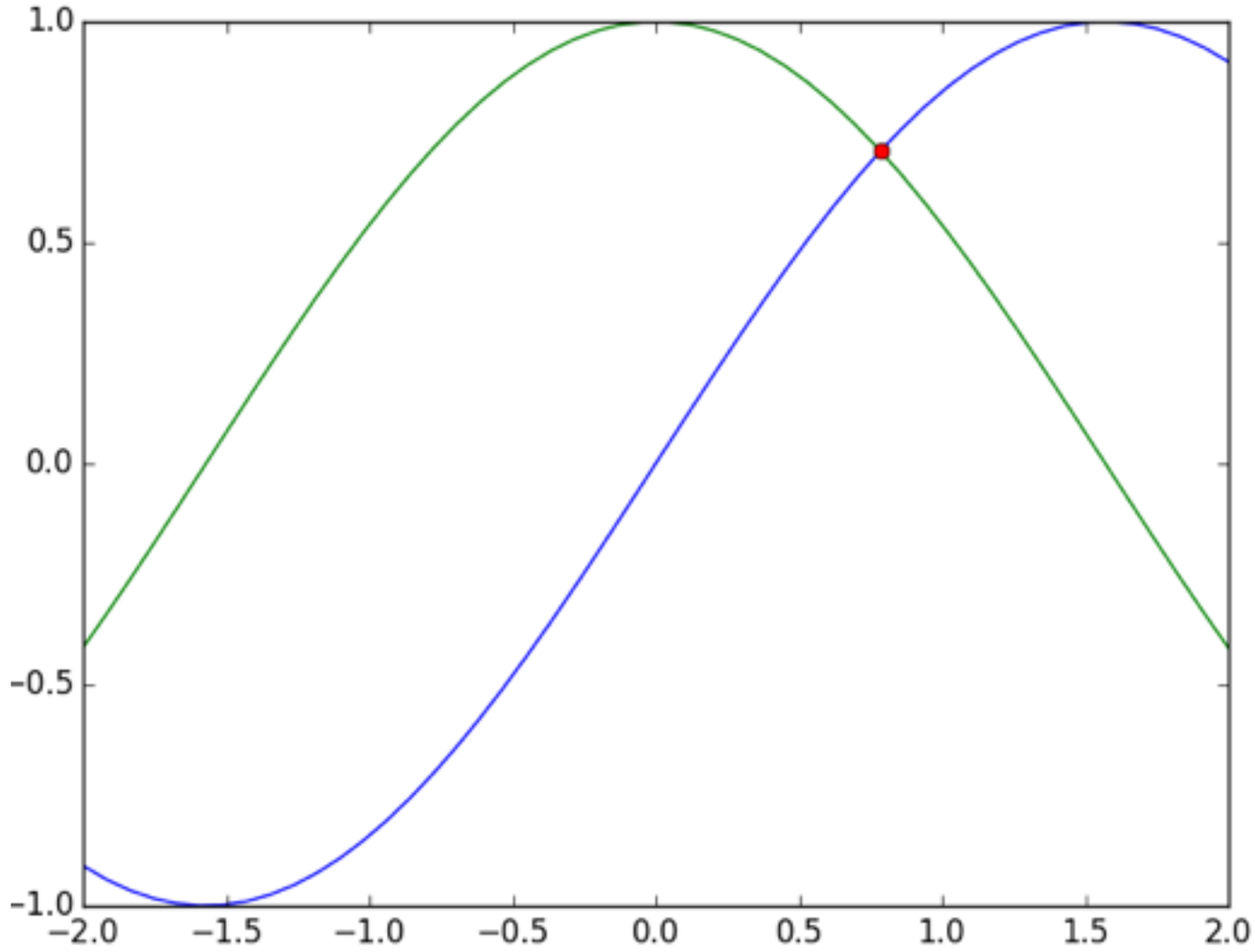


- The function fsolve provided return the roots of the (mostly non-linear) equations defined by func(x) = 0 given a starting estimate.

Intersection of two functions

```
from scipy.optimize import fsolve

def findIntersection(fun1,fun2,x0):
 return fsolve(lambda x: fun1(x) -
fun2(x),x0)

result = findIntersection(np.sin,np.cos,
0.0)
print(result)
```

```
0.78539816 # (for 45 degree angle)
```

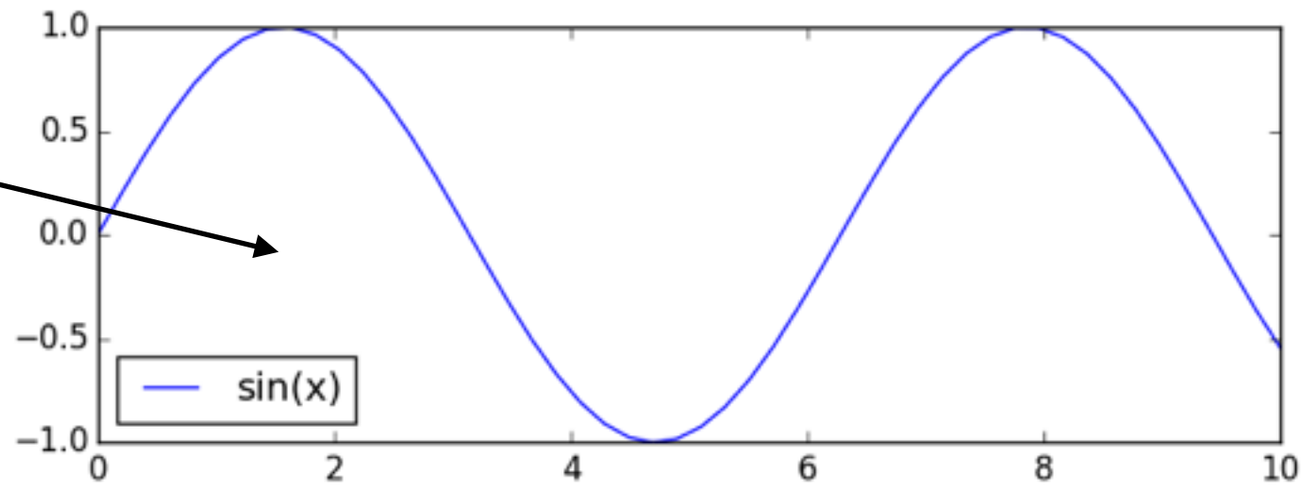


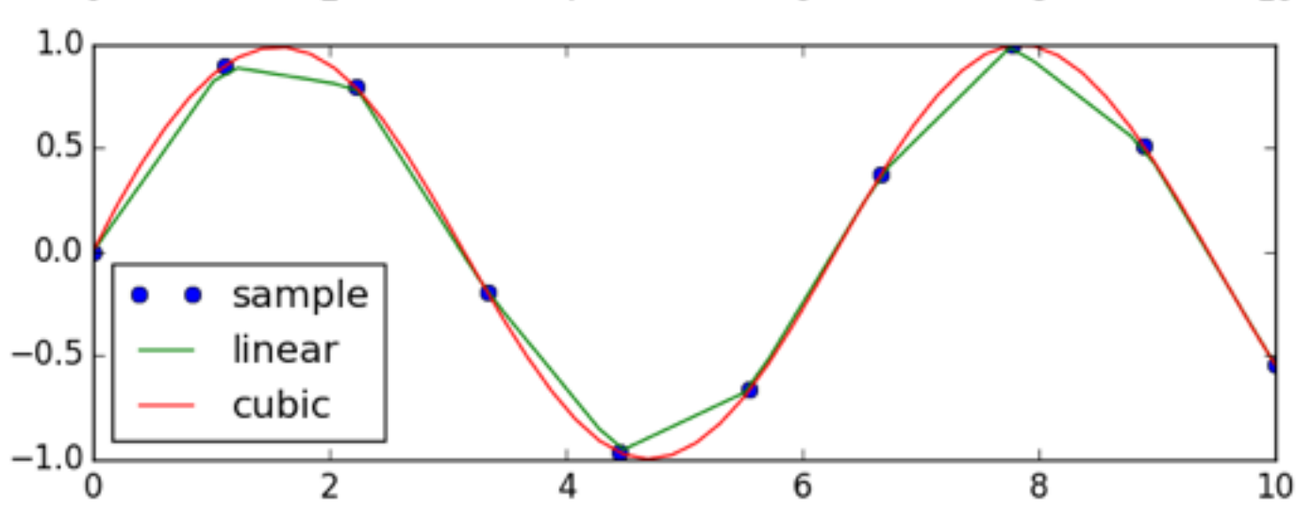
Interpolation

- Given a set of sample data, obtaining the intermediate values between the points is useful to understand and predict what the data will do in the non-sampled domain.
- Type of interpolation in Scipy
  - Univariate: only one independent variable
  - Multivariate: more than one independent variable

Interpolation

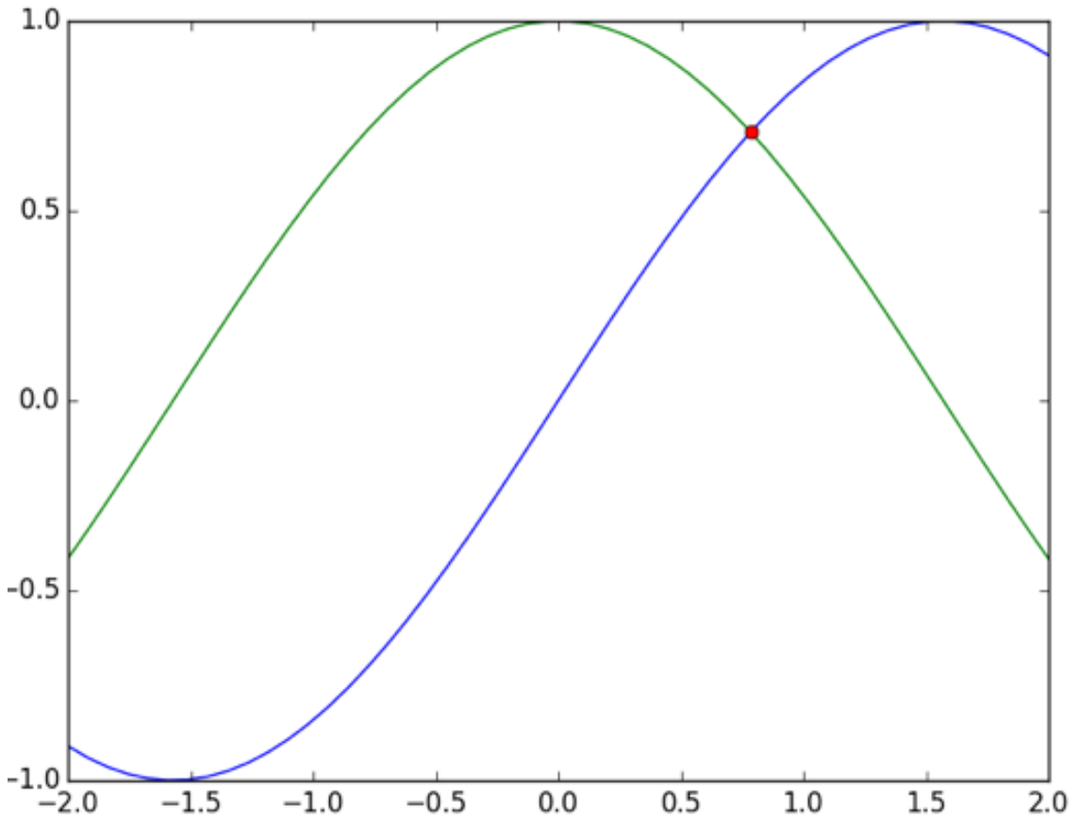
```
import scipy.interpolate as sp

# 50 real points of sin(x) in [0 10]
xx = np.linspace(0, 10, 50)
yy = np.sin(xx)

# 10 sample of sin(x) in [0 10]
x = np.linspace(0, 10, 10)
y = np.sin(x)

# interpolation
fl = sp.interp1d(x, y,kind='linear')
fc = sp.interp1d(x, y,kind='cubic')

# create sample data
xnew = np.linspace(0, 10, 50)

# interpolation
y_il =  fl(xnew)
y_ic = fc(xnew))
```





Assignment 8(b) (20 mins)

- Try to solve the linear equation for y = x+10 (2pt)
- Draw this plot for previous function intersection for 50 data points between -2 to 2 (2pt)



1. File name : **as8b_yoursurname_name.py**
2. Upload your solution at Moodle under A08 folder after completion.

Assignment 8(b) (20 mins)

- Try to interpolate for cos x for 100 sample points and draw the plot similar to sin x. (2pt)

1. File name : **as8b_yoursurname_name.py**
2. Upload your solution at Moodle under A08 folder after completion.

Integration

- Integration is a crucial tool in math and science.
- What is integration: Given a curve from a function or a dataset, it is the area under the curve
- The main purpose of integration with SciPy is to obtain numerical solutions. For indefinite integral solutions, refer SymPy.
- It solves mathematical problems symbolically for many types of computation beyond calculus.

Analytical Integration

- Analytical solutions can be obtained exactly with pencil and paper.
- Example:

```
from scipy.integrate import quad

# Defining function to integrate
func = lambda x: np.cos(np.exp(x)) ** 2

# Integrating function with upper and lower
# limits of 0 and 3, respectively
solution = quad(func, 0, 3)
print(solution)
# The first element is the desired value
# and the second is the error.
(1.296467785724373, 1.3977971862659880e-09)
```

$$\int_0^3 \cos^2(e^x)\,\mathrm{d}x$$

Numerical Integration

- In numerical integration, we are given data instead of some known equation

```
from scipy.integrate import trapz

x = np.sort(np.random.randn(150)*4+ 4).clip(0,5)
func1 = lambda x: np.sin(x) * np.cos(x ** 2) + 1
y = func1(x)

dsolution = trapz(y, x=x)

print(dsolution)
5.05810965371
```

Assignment 8(c) (15mins)

- Find the difference between the numerical solution and analytical for the previous example having function as “func” in the limit (0,5) respectively. (2pt)

1. File name : **as8c_yoursurname_name.py**
2. Upload your solution at Moodle under A08 folder after completion.

Statistics

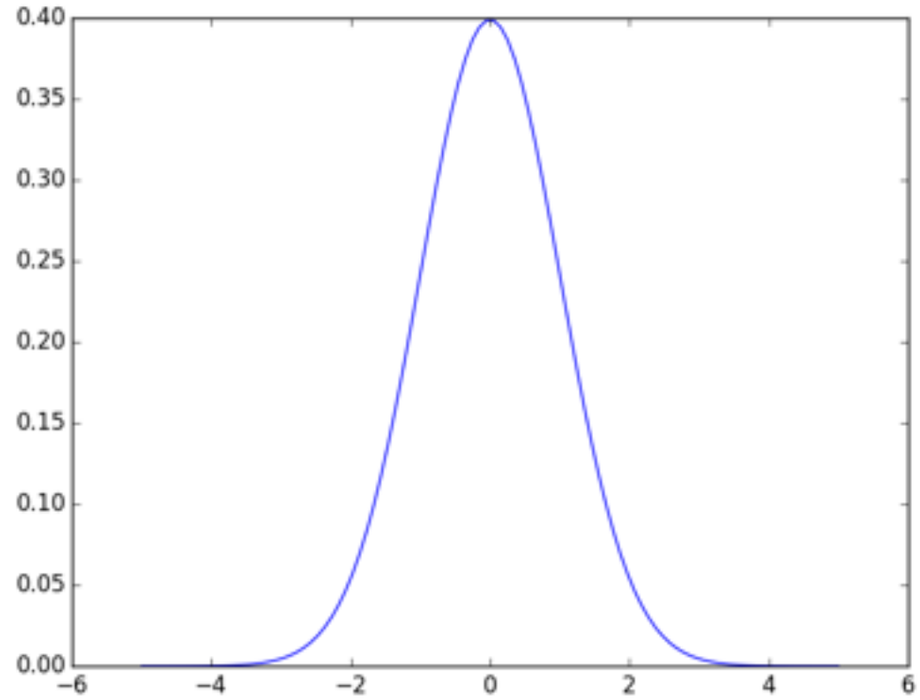
- Numpy already covered the basic functions of statistics.
- SciPy contains all of the statistical functions that you'll probably ever need.
- The scipy.stats module is based around the idea of a 'random variable' type.
- A whole variety of standard distributions are available:
  - ➡Continuous distributions: Normal, Maxwell, Cauchy, Chi-squared,, Gilbrat, Nakagami etc.
  - ➡ Discrete distributions: Poisson, Binomial, Geometric, Bernoulli, . . .
- The 'random variables' have all of the statistical properties of the distributions built into them already: cdf, pdf, mean, variance, moments etc..

Normal distribution

- All continuous distributions take **loc** and **scale** as keyword parameters to adjust the location and scale of the distribution.
- In general the distribution of a random variable X is obtained from

  (X - loc) / scale. The default values are loc = 0 and scale = 1.

```
from scipy.stats import norm
x = np.linspace(-5, 5, 100)
plt.plot(x, norm.pdf(x))
```



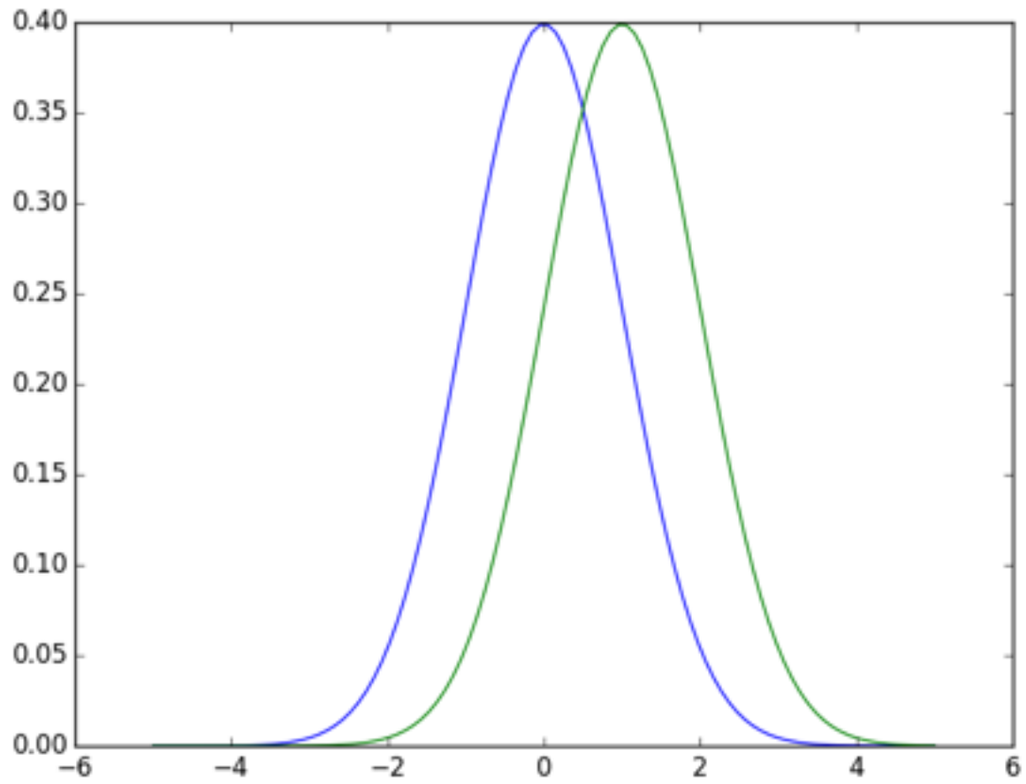
More info about Normal distribution: https://en.wikipedia.org/wiki/Normal_distribution#Moments

Normal distribution

- All continuous distributions take loc and scale as keyword parameters to adjust the location and scale of the distribution.
- In general the distribution of a random variable X is obtained from

  (X - loc) / scale. The default values are loc = 0 and scale = 1.

```
from scipy.stats import norm
x = np.linspace(-5, 5, 100)
plt.plot(x, norm.pdf(x))
plt.plot(x, norm.pdf(x,loc=1))
```



Assignment 8(d) (20mins)

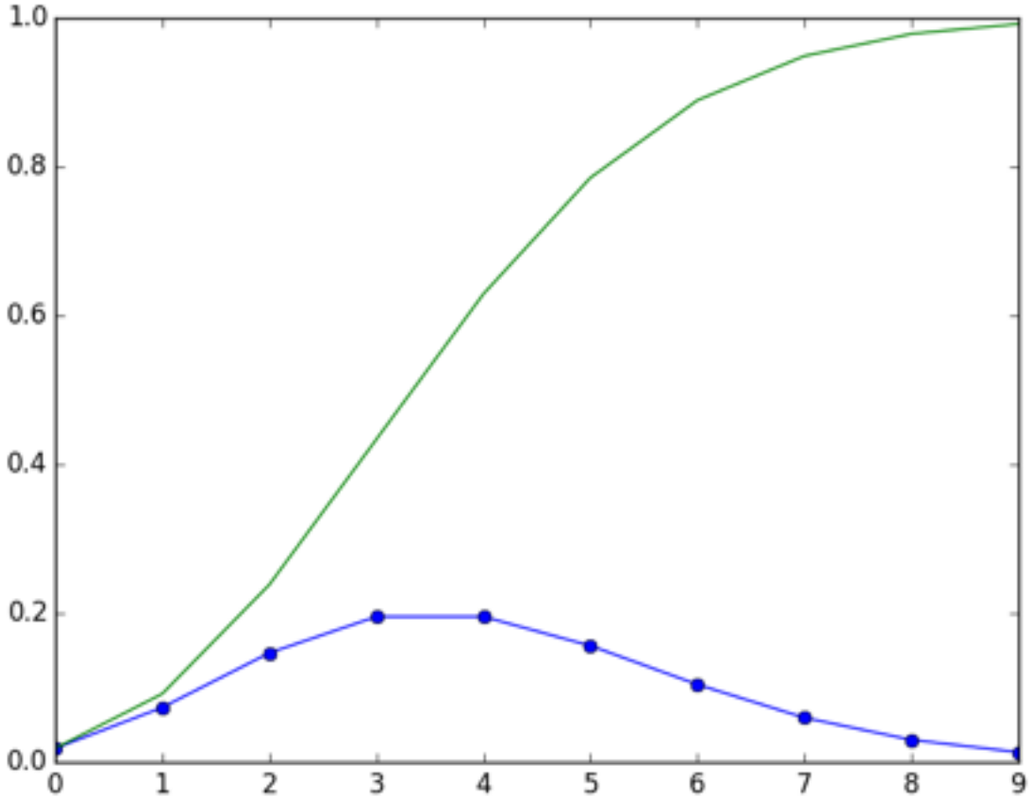
- Find the mean and standard deviation and three moments of normal distribution having loc = -1, scale =2 (3pts)
- Draw 1000 random sample from the distribution with same loc and scale values and plot their histogram. (3pts)
- Fit the pdf of the distribution over histogram (4pts)

1. File name : **as8d_yoursurname_name.py**
2. Upload your solution at Moodle under A08 folder after completion.

Poisson distribution

- An example of discrete distribution
- Note that discrete distributions have Probability Mass Functions (PMF) instead of Probability Distribution Functions (PDF).

```
from scipy.stats import poisson
x = np.arange(10)
plt.plot(x, poisson.pmf(x, 4), 'o-')
plt.plot(x, poisson.cdf(x, 4))
```



More info about Poisson distribution: https://en.wikipedia.org/wiki/Poisson_distribution

Percentiles

- The percentile is an estimator of the CDF: cumulative distribution function.
- Mean is the percentile 50 because 50% of the observation are below it.

```
from scipy.stats import stats
a = np.random.normal(size=1000)
stats.scoreatpercentile(a, 50)
stats.scoreatpercentile(a, 90)
```

Statistical Tests

- A statistical test is a decision indicator.
- Statistical tests are describe to test samples.
- Example: T-test can be used to see if two sets generated from gaussian process are significantly different or not ?

Kolmogorow-Smirnow (KS)Test

- Do two data samples come from the same distribution?

```
import scipy.stats as stats

# generate two data samples from different distributions
samp1 = stats .norm. rvs ( loc=0., scale=1., size=100)
samp2 = stats .norm. rvs ( loc=2., scale=1., size=100)

# perform ks test : null hypothesis is distributions are the same
D, pval = stats.ks2samp(samp1, samp2)
>>> D =.58
>>> pval=1.34e-15
# reject the null hypothesis
# D is the KS statistic and the closer it is to 0 the better.

# generate two data samples from the same distribution
samp1 = stats .norm. rvs ( loc=0., scale=1., size=100)
samp2 = stats .norm. rvs ( loc=0., scale=1., size=100)

# perform ks test
D, pval = stats . ks2samp(samp1, samp2)
>>> D=.09
>>> pval=0.79
# fail to reject the null hypothesis
```

Clustering of data

- Clustering analysis are key to identifying patterns, groups, and clusters.
- In big data and data mining, identifying data clusters is becoming important to organize discovered information.
- SciPy provides a cluster analysis class (scipy.cluster).

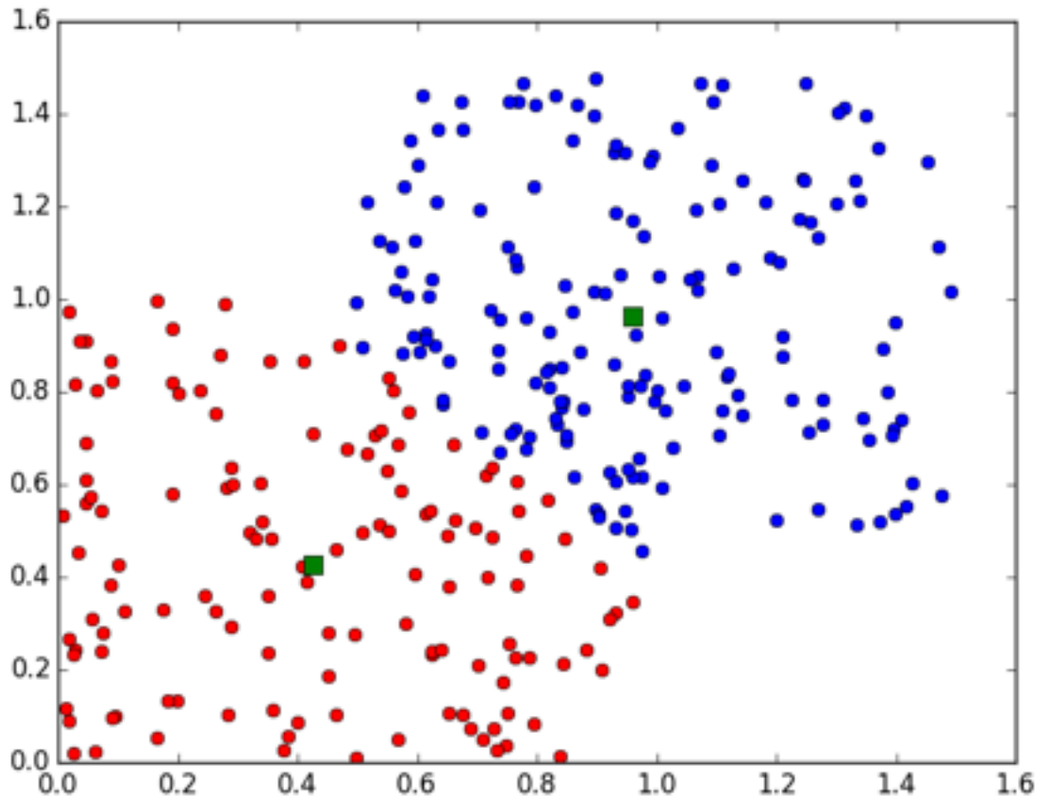
K-means clustering

- K-means clustering is a method for finding clusters and cluster centers in a set of un-labeled data.

1. Choose k number of points randomly and make them initial centroids.
2. Select a data point from the collection, compare it with each centroid and if the data point is found to be similar with the centroid then assign it into the cluster of that centroid.
3. When each data point has been assigned to one of the clusters, re-calculate the value of the centroids for each k number of clusters.
4. Repeat steps 2 to 3 until no data point moves from its previous cluster to some other cluster (termination criterion has been satisfied).

K-means clustering

```
from scipy.cluster.vq import kmeans,vq

# data generation
data = np.vstack((np.random.rand(150,2) + np.array([.
5,.5]),np.random.rand(150,2)))

# computing K-Means with K = 2 (2 clusters)
centroids,distortion = kmeans(data,2)

# assign each sample to a cluster
idx,distortion1 = vq(data,centroids)

# # some plotting using numpy's logical indexing
plt.plot(data[idx==0,0],data[idx==0,1],'ob',data[idx==1,0]
,data[idx==1,1],'or')
plt.plot(centroids[:,0],centroids[:,1],'sg',markersize=8)
```



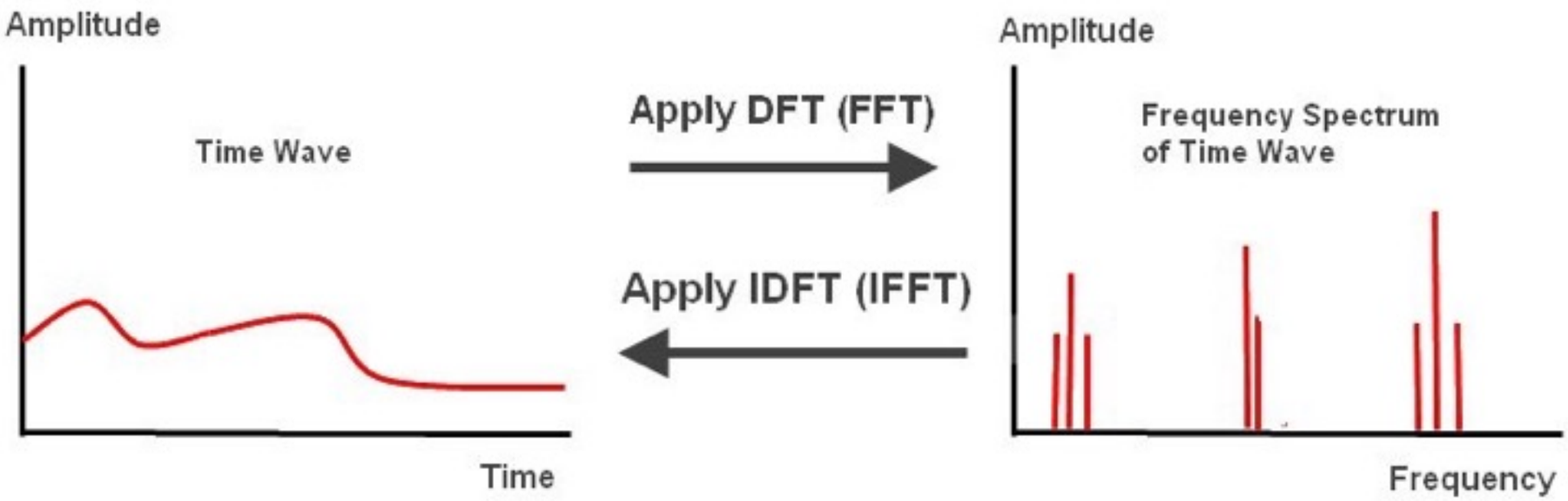
Assignment 8(e) (20 mins)

- Repeat the K-means clustering algo with 3 clusters and plot it. (5pt)

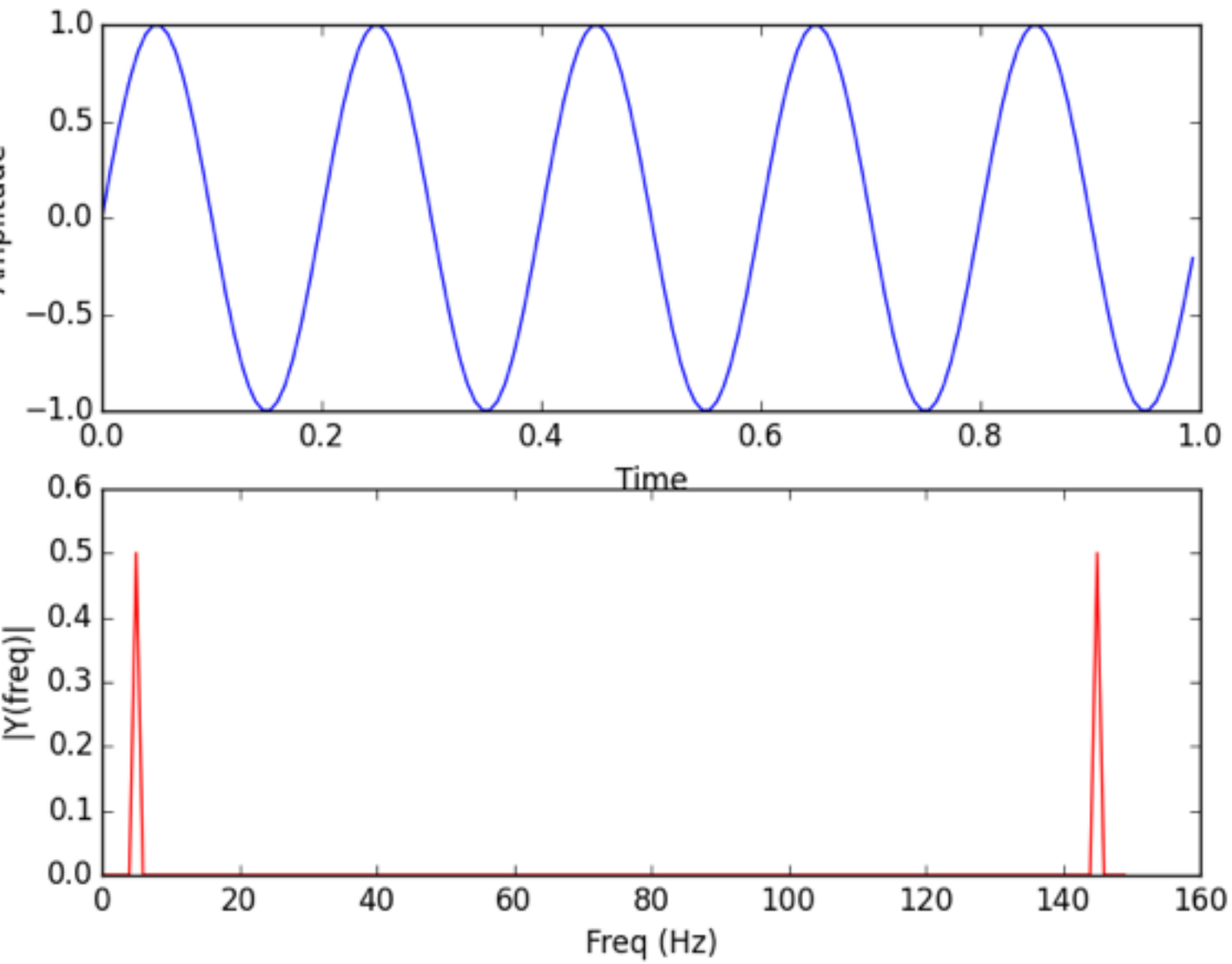
1. File name : **as8e_yoursurname_name.py**
2. Upload your solution at Moodle under A08 folder after completion.

Fast Fourier Transform

- The scipy.fftpack module allows to compute fast Fourier transforms.
- FFT is an algorithm that speeds up the calculation of a Discrete Fourier Transform.
- DFT is an algorithm that takes a signal and determines the 'frequency content' of the signal OR transforms a discrete number of samples of a time wave and converts them into a frequency spectrum.
- Discrete Fourier transforms (DFTs) are extremely useful because they reveal periodicities in input data as well as the relative strengths of any periodic components.



Fast Fourier Transform




```
from scipy import fft

Fs = 150  # sampling rate
Ts = 1/Fs # sampling interval
t = np.arange(0,1,Ts) # time vector

ff = 5  # frequency of the signal
y = np.sin(2*np.pi*ff*t) # signal

n = len(y) # length of the signal
k = np.arange(n)
T = n/Fs
frq = k/T # two sides frequency range

Y = fft(y)/n # fft computing and
             # normalisation
```

Image Processing

- Display image from module

```
from scipy import misc
from scipy import ndimage
```

```
f = misc.face(gray=True)
plt.imshow(f, cmap=plt.cm.gray)
```

Image and code Sources : http://www.scipy-lectures.org/advanced/image_processing/

Image Processing

- Change the contrast

```
from scipy import misc

plt.imshow(f, cmap=plt.cm.gray,
vmin=30, vmax=200)
```

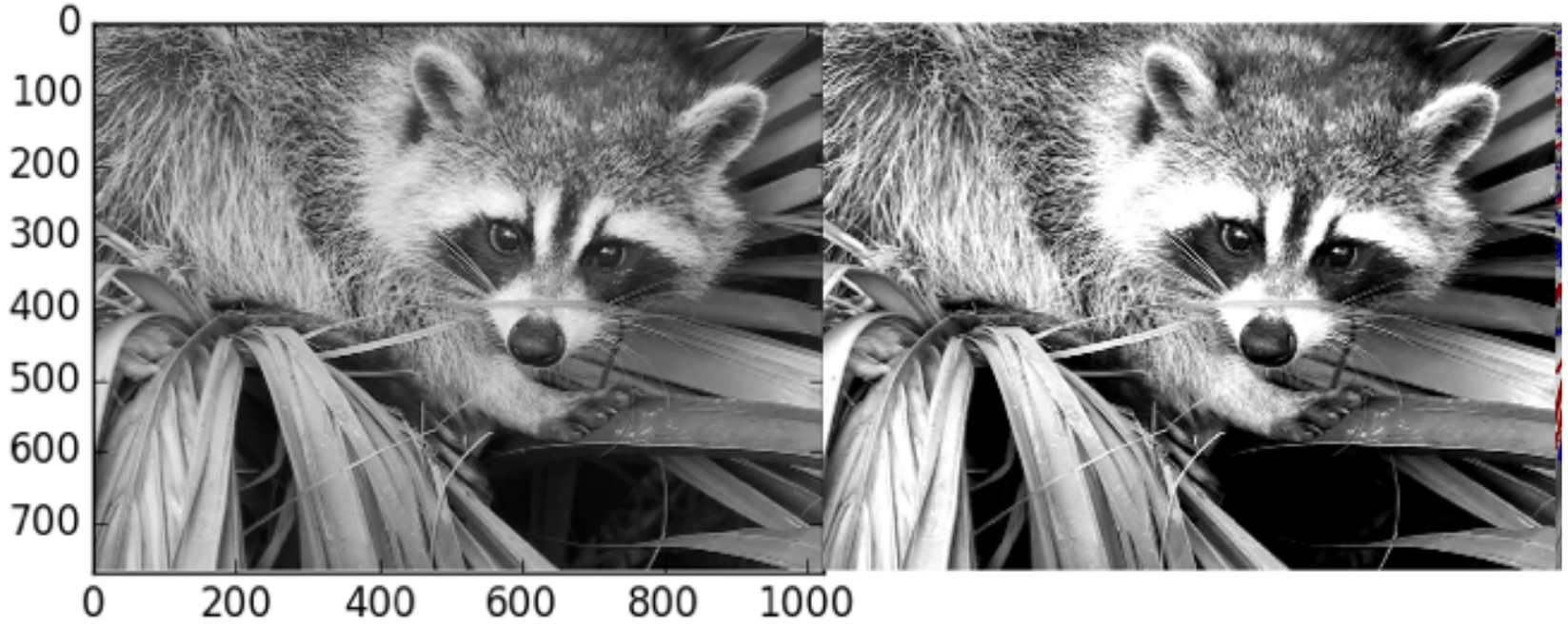
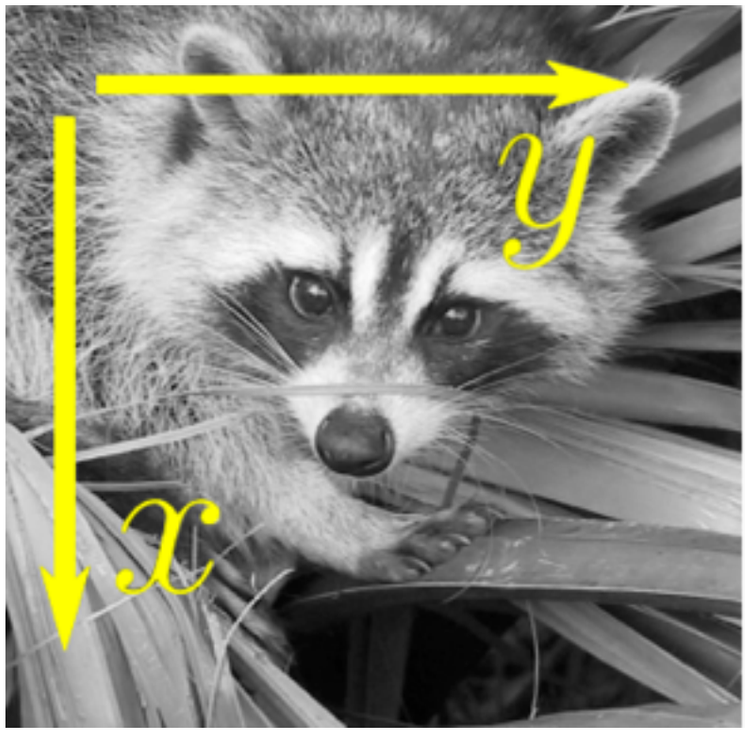


Image and code Sources : http://www.scipy-lectures.org/advanced/image_processing/

Basic Manipulation

Images are arrays: use the whole numpy machinery.



0	1	2
3	4	5
6	7	8

Image and code Sources : http://www.scipy-lectures.org/advanced/image_processing/

Lets apply some masking

```
face = misc.face(gray=True)

lx, ly = face.shape
X, Y = np.ogrid[0:lx, 0:ly]
mask = (X - lx/2)**2 + (Y - ly/2)**2 >
lx*ly/4

face[mask] = 0
```

Image and code Sources : http://www.scipy-lectures.org/advanced/image_processing/

Assignment 8(f) (30mins)

- Play with picture and try to covert it in the following format and plot it (10pt)
- Hint1: second image is almost 1/4th of the figure
- Hint2: 5th image is without reshaping

1. File name : **as8f_yoursurname_name.py**
2. Upload your solution at Moodle under A08 folder after completion.
3. Do NOT look for the solution online