In [1]:

```
# importing and extracting required data and libraries
!wget https://www.iith.ac.in/~shantanud/testdata.dat

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

```
--2022-02-17 09:49:36--  https://www.iith.ac.in/~shantanud/testdata.dat
Resolving www.iith.ac.in (www.iith.ac.in)... 218.248.6.135
Connecting to www.iith.ac.in (www.iith.ac.in)|218.248.6.135|:443... conn
ected.
HTTP request sent, awaiting response... 200 OK
Length: 707
Saving to: ‘testdata.dat’

testdata.dat        100%[===================>]     707  --.-KB/s    in 0
s

2022-02-17 09:49:37 (34.2 MB/s) - ‘testdata.dat’ saved [707/707]
```

Question 1

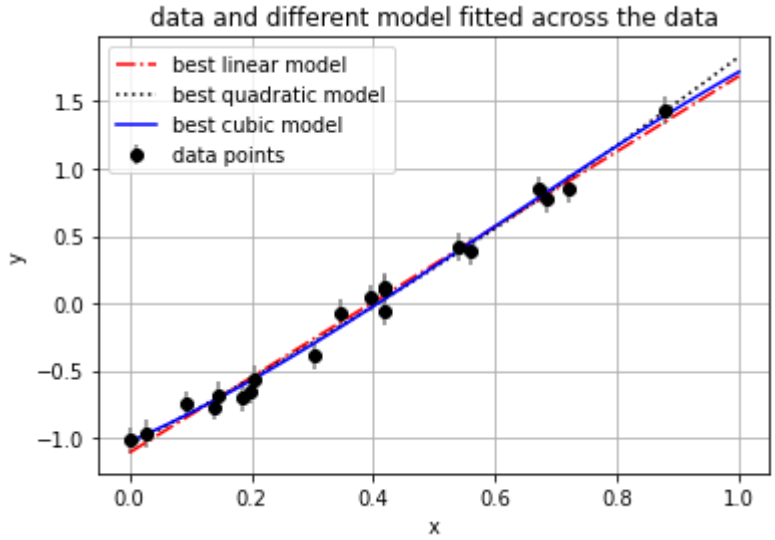
In [2]:

```
# loading data in variables
data = np.loadtxt('testdata.dat', unpack = True)
x = data[0,:]
y = data[1,:]
sigma_y = data[2,:]

# defining linear, quadratic and cubic function
def linear_fit(t, a, b):
    return a*t + b

def quad_fit(t, a, b, c):
    return a*(t**2) + b*t + c

def cubic_fit(t, a, b, c, d):
    return a*(t**3) + b*(t**2) + c*t + d

# helper function for AIC & BIC calculation
def logLH(degree, param, data):
    x, y, sigma_y = data
    if degree == 1:
        y_fit = linear_fit(x, param[0], param[1])
    if degree == 2:
        y_fit = quad_fit(x, param[0], param[1], param[2])
    if degree == 3:
        y_fit = cubic_fit(x, param[0], param[1], param[2], param[3])

    return sum(stats.norm.logpdf(*args) for args in zip(y, y_fit, sigma_y))

# functions for finding AIC & BIC
def find_AIC(degree, param, data):
    return -2* logLH(degree, param, data) + 2*(degree + 1)

def find_AICc(degree, param, N, data):
    k = degree + 1
    return find_AIC(degree, param, data) + (2*k*k + 2*k)/(N-k-1)

def find_BIC(degree, param, N, data):
    return -2 * logLH(degree, param, data) + (degree + 1) * np.log(N)

# function for finding error wrt curve fit parameters
def find_error(degree, param, data):
    if degree == 1:
        return (y - linear_fit(x, param[0], param[1]))/sigma_y
    if degree == 2:
        return (y - quad_fit(x, param[0], param[1], param[2]))/sigma_y
    if degree == 3:
        return (y - cubic_fit(x, param[0], param[1], param[2], param[3]))/sigma_y

# helper function for finding chi-square values
def compute_chi_square(degree, param, data):
    x, y, sigma_y = data
    temp = find_error(degree, param, data)
    return np.sum(temp ** 2)

# function for computing degree of freedom
def compute_dof(degree, data):
    return data.shape[1] - (degree + 1)

# function for finding chi-square likelihood
```

```
def chi_square_likelihood(degree,param, data):
    chi2 = compute_chi_square(degree, param, data)
    dof = compute_dof(degree, data)
    return stats.chi2(dof).pdf(chi2)

# finding curve fit parameters
linear_param, _ = optimize.curve_fit(linear_fit, xdata=x, ydata=y, sigma=sigma_y)
quad_param, _ = optimize.curve_fit(quad_fit, xdata=x, ydata=y, sigma=sigma_y)
cubic_param, _ = optimize.curve_fit(cubic_fit, xdata=x, ydata=y, sigma=sigma_y)

# finding error corresponding to each fitting
linear_error = find_error(1, linear_param, data)
quadratic_error = find_error(2, quad_param, data)
cubic_error = find_error(3, cubic_param, data)

# finding and printing required values
print(f"Best fit values for Linear model (ax+b) are [a b] = {linear_param}")
print(f"Best fit values for Quadratic model (ax^2 + bx + c) are [a b c] = {quad_param}"
)
print(f"Best fit values for Cubic model (ax^3 + bx^2 + cx + d) are [a b c d] = {cubic_p
aram}\n")

print("chi-square likelihood")
print(f"linear model: {chi_square_likelihood(1, linear_param, data)}")
print(f"quadratic model: {chi_square_likelihood(2, quad_param, data)}")
print(f"cubic model: {chi_square_likelihood(3, cubic_param, data)}")
print("Linear model is most suitable according to above data.\n")

print(f"AIC Values (Linear Model) = {find_AIC(1, linear_param, data)}")
print(f"AIC Values (Quadratic Model) = {find_AIC(2, quad_param, data)}")
print(f"AIC Values (Cubic Model) = {find_AIC(3, cubic_param, data)}")
print("Since Linear model has least AIC value, it is most suitable.\n")

print(f"AICc Values (Linear Model) = {find_AICc(1, linear_param, len(x), data)}")
print(f"AICc Values (Quadratic Model) = {find_AICc(2, quad_param, len(x), data)}")
print(f"AICc Values (Cubic Model) = {find_AICc(3, cubic_param, len(x), data)}")
print("Since Linear model has least AICc value, it is most suitable.\n")

print(f"BIC Values (Linear Model) = {find_BIC(1, linear_param, len(x), data)}")
print(f"BIC Values (Quadratic Model) = {find_BIC(2, quad_param, len(x), data)}")
print(f"BIC Values (Cubic Model) = {find_BIC(3, cubic_param, len(x), data)}")
print("Since Linear model has least BIC value, it is most suitable.\n")

# finding and printing p-value of quadratic and cubic model with respect to linear mode
l
p_val_quad = 1-stats.chi2(1).cdf(stats.chi2(len(x)-2).pdf(np.sum(linear_error**2))- sta
ts.chi2(len(x)-3).pdf(np.sum(quadratic_error**2)))
p_val_cubic = 1-stats.chi2(2).cdf(stats.chi2(len(x)-2).pdf(np.sum(linear_error**2))- st
ats.chi2(len(x)-4).pdf(np.sum(cubic_error**2)))

print(f"Quadratic P-value wrt Linear Model is {p_val_quad}")
print(f"Cubic P-value wrt Linear Model is {p_val_cubic}\n")

# print("best model suited will be linear model")

# declaring points on x-axis
x1 = np.linspace(0, 1, 2500)

# plot
fig, ax = plt.subplots()
ax.errorbar(x, y, sigma_y, fmt='ok', ecolor='gray', label='data points')
```

```
ax.plot(x1, linear_fit(x1, linear_param[0], linear_param[1]) , color = 'r', label='best
linear model', linestyle='-.')
ax.plot(x1, quad_fit(x1, quad_param[0], quad_param[1], quad_param[2]), color = 'black',
label='best quadratic model', linestyle=':')
ax.plot(x1, cubic_fit(x1, cubic_param[0], cubic_param[1], cubic_param[2], cubic_param[3
]), color = 'b', label='best cubic model')
ax.legend(loc='best')
ax.set(xlabel='x', ylabel='y', title='data and different model fitted across the data')
plt.grid()
plt.show()
```

```
Best fit values for Linear model (ax+b) are [a b] = [ 2.79789861 -1.110280
82]
Best fit values for Quadratic model (ax^2 + bx + c) are [a b c] = [ 0.5026
1293  2.38475187 -1.05578915]
Best fit values for Cubic model (ax^3 + bx^2 + cx + d) are [a b c d] = [-
0.96724992  1.74451332  1.97184055 -1.02910462]

chi-square likelihood
linear model: 0.045383795585918596
quadratic model: 0.036608447550140304
cubic model: 0.04215280601005994
Linear model is most suitable according to above data.

AIC Values (Linear Model) = -40.0366868160727
AIC Values (Quadratic Model) = -39.84982062400561
AIC Values (Cubic Model) = -38.26081851760256
Since Linear model has least AIC value, it is most suitable.

AICc Values (Linear Model) = -39.33080446313153
AICc Values (Quadratic Model) = -38.34982062400561
AICc Values (Cubic Model) = -35.594151850935894
Since Linear model has least AICc value, it is most suitable.

BIC Values (Linear Model) = -38.04522226896472
BIC Values (Quadratic Model) = -36.86262380334364
BIC Values (Cubic Model) = -34.2778894233866
Since Linear model has least BIC value, it is most suitable.

Quadratic P-value wrt Linear Model is 0.925365878185452
Cubic P-value wrt Linear Model is 0.9983858094213666
```



Question 2

In [3]:

```
# extracting data from JVDP website
data1 = np.array([[0.42,  0.72,  0.,  0.3,  0.15,
                   0.09,  0.19,  0.35,  0.4,  0.54,
                   0.42,  0.69,  0.2,  0.88,  0.03,
                   0.67,  0.42,  0.56,  0.14,  0.2],
                  [0.33,  0.41, -0.22,  0.01, -0.05,
                   -0.05, -0.12,  0.26,  0.29,  0.39,
                   0.31,  0.42, -0.01,  0.58, -0.2,
                   0.52,  0.15,  0.32, -0.13, -0.09],
                  [0.1,  0.1,  0.1,  0.1,  0.1,
                   0.1,  0.1,  0.1,  0.1,  0.1,
                   0.1,  0.1,  0.1,  0.1,  0.1,
                   0.1,  0.1,  0.1,  0.1,  0.1]])

# declaring variables
jvdp_x, jvdp_y, jvdp_sigma_y = data1

# finding curve fit parameters
jvdp_linear_param, _ = optimize.curve_fit(linear_fit, xdata=jvdp_x, ydata=jvdp_y, sigma
=jvdp_sigma_y)
jvdp_quad_param, _ = optimize.curve_fit(quad_fit, xdata=jvdp_x, ydata=jvdp_y, sigma=jvd
p_sigma_y)

# finding and printing required values
print(f"Linear Model AIC value = {find_AIC(1, jvdp_linear_param, data1)}")
print(f"Linear Model AICc value = {find_AICc(1, jvdp_linear_param, len(jvdp_x), data
1)}")
print(f"Linear Model BIC value = {find_BIC(1, jvdp_linear_param, len(jvdp_x), data1)}\n
")

print(f"Quadratic Model AIC value = {find_AIC(2, jvdp_quad_param, data1)}")
print(f"Quadratic Model AICc value = {find_AICc(2, jvdp_quad_param, len(jvdp_x), data
1)}")
print(f"Quadratic Model BIC value = {find_BIC(2, jvdp_quad_param, len(jvdp_x), data1)}
\n")

print("Since Linear model has least AIC, AICc and BIC value, it is most suitable.")
```

```
Linear Model AIC value = -40.02173401322526
Linear Model AICc value = -39.31585166028409
Linear Model BIC value = -38.03026946611728

Quadratic Model AIC value = -39.88302717300821
Quadratic Model AICc value = -38.38302717300821
Quadratic Model BIC value = -36.89583035234624

Since Linear model has least AIC, AICc and BIC value, it is most suitable.
```

Question 3

Title of the paper is "The Kolmogorov-Smirnov Test for Goodness of Fit".

The standard tables used for K.S. test, consist of set of observations from a completely specified continuous distribution, mainly this test is used to tell whether a set of observations are from a normal distribution, when mean and variance are not given, but must be estimated from given sample.

Another usage of this non-parametric statistic is that, it can be used to compare two empirical distributions, which defines largest absolute difference between 2 cumulative distribution functions, as a measure of disagreement.

In the paper by by N.D. Gagunashvili, on the comparison of weighted and unweighted histograms, where it is being discussed that, that how X^2 test are used for histograms for different events, the K.S. test for histograms about different statistics of event.

References:

1) Massey, F. J. 1951. “The Kolmogorov-Smirnov Test for Goodness of Fit,”. Journal of the American Statistical Association, 46: 68–78. (Google Scholar)

2) Gagunashvili N 2006 Comparison of weighted and unweighted histograms arXiv: physics/0605123

Question 4

In [4]:

```
# calculation for higgs boson
higgs_pval = [10**-1, 10**-2, 10**-3, 10**-5, 10**-7, 10**-9]
higgs_sign = stats.norm.isf(higgs_pval)

# calculation for LIGO
p_val_ligo = 2*10**-7
sign_ligo = stats.norm.isf(p_val_ligo)

# calculation for super-k discovery
chi_square = 65.2
dof = 67
chi_square_gof = 1-stats.chi2(dof).cdf(chi_square)

# print
print('Significance in terms of number of sigmas for higgs boson: ', higgs_sign)
print('Significance in terms of number of sigmas for LIGO discovery: ', sign_ligo)
print('chi-square GOF is: ', chi_square_gof)
```

```
Significance in terms of number of sigmas for higgs boson:  [1.28155157 2.
32634787 3.09023231 4.26489079 5.19933758 5.99780702]
Significance in terms of number of sigmas for LIGO discovery:  5.068957749
717791
chi-square GOF is:  0.5394901931099038
```