Fitting Data to Models

Name: Sankalp Saoji (EE16B063)

Date: 27/02/2018

Abstract

In this week, we studied the fitting of data using different models. At the end, we learnt about the effect of noise on the fitting process.

1 INTRODUCTION

This week's Python assignment focussed on the following topics:

- Studying fitting of data using models
- Studying the effect of noise on the fitting process

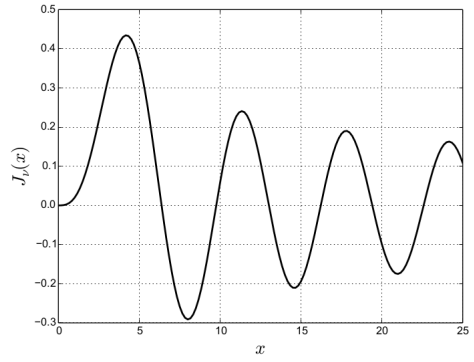
1.1 Functions Used:

We used Bessel function of the first type, $J_v(x)$.

For large x,

$J_v(x) \approx \sqrt{\frac{2}{\pi x}} cos(v - \frac{\pi x}{2} - \frac{\pi}{4})$

It looks like the following,



2 Lab Questions

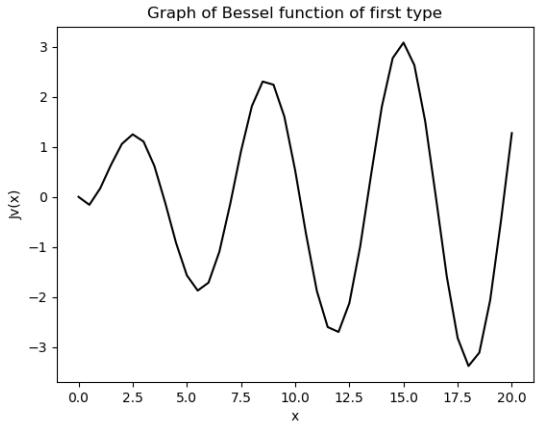
2.1 Vector generation and function declaration:

In this question we were supposed to generate a vector of 41 values from 0 to 20.

```
# Importing modules

from scipy.integrate import quad
from pylab import *
from matplotlib import pyplot as plt
import numpy


    # Generating vector

    x = linspace(0,20,41) print(x)

    # Declaring function and obtaining values

    def bessel(t, v):
            j = (sqrt(2/(math.pi)*t))*
        (numpy.cos(t-v*(math.pi)/2-(math.pi)/4))
            return j

    m = bessel(x, 1) print(m)
```

Below are the values I got for the bessel function when I used the vector x.

```
0 :  -0.000000 1 :  -0.158842 2 :  0.169916 3 :
0.640379 4 :  1.057552 5 :  1.248544 6 :  1.105329 7 :
0.618180 8 :  -0.116402 9 :  -0.917648 10 :  -1.567605
11 :  -1.871201 12 :  -1.713079 13 :  -1.095300 14 :
-0.144667 15 :  0.913716 16 :  1.810972 17 :  2.303654
18 :  2.239689 19 :  1.603348 20 :  0.526407 21 :
-0.738876 22 :  -1.879468 23 :  -2.599655 24 :
-2.697920 25 :  -2.122613 26 :  -0.991232 27 :  0.432967
28 :  1.802523 29 :  2.771009 30 :  3.080934 31 :
2.631966 32 :  1.511478 33 :  -0.021516 34 :  -1.596327
35 :  -2.820564 36 :  -3.378173 37 :  -3.111006 38 :
-2.062882 39 :  -0.474050 40 :  1.273837
```



2.2 Extracting subvectors of x for different x_0:

In this question we were supposed to do the following:

For different $x_0 = 0.5, 1, \ldots, 18$, we had to extract the subvectors of x and find $J_1(x)$ that correspond to $x \geq x\ 0$. For each x_0 , we constructed the matrix corresponding to $A\cos(x_i) + B\sin(x_i) \approx J_1(x_i)$ and obtained the best fit of A and B. Later, we obtained φ corresponding to the solution by,

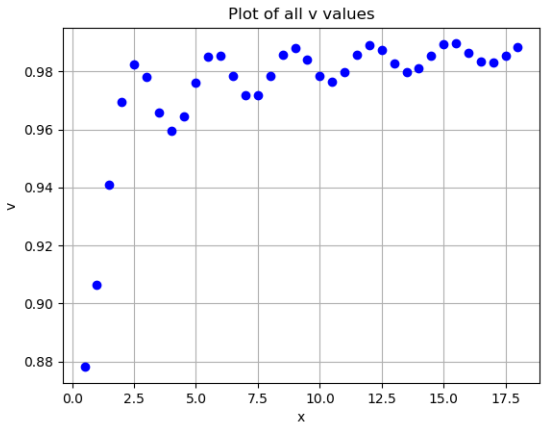
$\cos(\phi) = \frac{A}{\sqrt{A^2+B^2}}$

Hence, we predicted ν.

```
# Getting values of v

w = list()
v = list()
d = []

y = linspace(0.5,18,36)

for k in y:
    z = numpy.arange(k,20.5,0.5)
    l = len(z)
    P = numpy.zeros((l,2))
    P[:,0] = numpy.cos(z)
    P[:,1] = numpy.sin(z)
    b = spe.jv(1,z)
    c = lstsq(P,b,rcond=-1)[0]
    A = c[0]
    B = c[1]
    p = (A/(numpy.sqrt(A**2+B**2)))
    f = numpy.arccos(p)
    w.append(f)
```

```
    v = ((2*f)/(numpy.pi)-(1/2))
    d.append(v)

print(v)


    # Plotting v values

    plt.figure(1)
    plt.plot(y,d,'ro')
    plt.xlabel('x')
    plt.ylabel('v')
    plt.title('Plot of all v values')
    plt.grid(True)
    plt.show()
```

Following is the graph I obtained.



2.3 Repeating above question with amplitude considered:

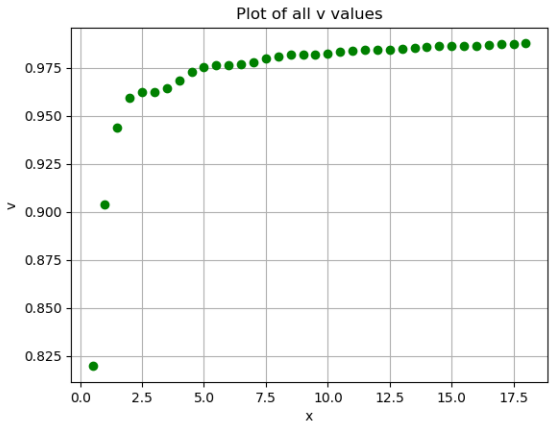
In this question we were supposed to us,

$A\frac{cos(x_i)}{\sqrt{x_i}} + B\frac{sin(x_i)}{\sqrt{x_i}} \approx J_1(x_i)$

```
#Changing earlier algorithm a bit considering amplitude

for k in y:
    z = numpy.arange(k,20.5,0.5)
    l = len(z)
    P = numpy.zeros((l,2))
    P[:,0] = numpy.cos(z)/(numpy.sqrt(z))
    P[:,1] = numpy.sin(z)/(numpy.sqrt(z))
```

```
    b = spe.jv(1,z)
    c = lstsq(P,b,rcond=-1)[0]
    A = c[0]
    B = c[1]
    p = (A/(numpy.sqrt(A**2+B**2)))
    f = numpy.arccos(p)
    w.append(f)
    v = ((2*f)/(numpy.pi)-(1/2))
    d.append(v)

print(v)
```

Below is the graph I obtained.



As we see from the above two graphs, the formula which considered amplitude is far more accurate than the one which does not consider amplitude.

2.4 Conversion to a function:

In this question, we were supposed to create a function with our algorithm.

function=calcnu(x, x_0, model)

Here,

calcnu is our function,

x is our array for which we find bessel function,

$x_0$is our starting point,

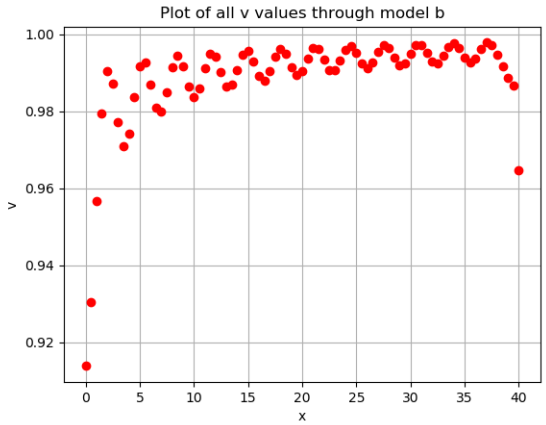
model means the model we want to select like the one with amplitude not considered or the one with considered.

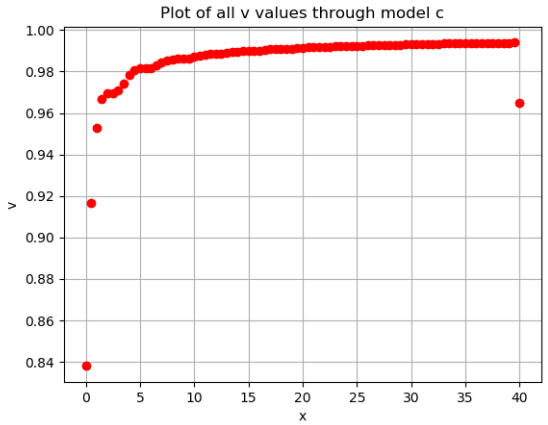
```
# Creating a function without noise

def calcnu(x, x0, model):
```

```
if (model == 'b'):
        w = list()
        v = list()
        d = []
        q = linspace(x0,40,81)
        for k in q:
                z = numpy.arange(k, x[len(x)-1] + 0.5  ,0.5)
                l = len(z)
                P = numpy.zeros((l,2))
                P[:,0] = numpy.cos(z)
                P[:,1] = numpy.sin(z)
                b = spe.jv(1,z)
                c = lstsq(P,b,rcond=-1)[0]
                A = c[0]
                B = c[1]
                p = (A/(numpy.sqrt(A**2+B**2)))
                f = numpy.arccos(p)
                w.append(f)
                v = (f - (math.pi)/4)*(2/(math.pi))
                d.append(v)

        plt.figure(1)
        plt.plot(x,d,'ro')
        plt.xlabel('x')
        plt.ylabel('v')
        plt.title('Plot of all v values through model b')
        plt.grid(True)
        plt.show()

if (model == 'c'):
        w = list()
        v = list()
        d = []
        q = linspace(x0,40,81)
        for k in q:
                z = numpy.arange(k, x[len(x)-1] + 0.5  ,0.5)
                l = len(z)
                P = numpy.zeros((l,2))
                P[:,0] = numpy.cos(z)/(numpy.sqrt(z))
                P[:,1] = numpy.sin(z)/(numpy.sqrt(z))
                b = spe.jv(1,z)
                c = lstsq(P,b,rcond=-1)[0]
                A = c[0]
                B = c[1]
                p = (A/(numpy.sqrt(A**2+B**2)))
                f = numpy.arccos(p)
```

```
                w.append(f)
                v = (f - (math.pi)/4)*(2/(math.pi))
                d.append(v)

        plt.figure(1)
        plt.plot(x,d,'ro')
        plt.xlabel('x')
        plt.ylabel('v')
        plt.title('Plot of all v values through model c')
        plt.grid(True)
        plt.show()

s = calcnu(x, 0.5, 1)
t = calcnu(x, 0.5, 2)
print(s)
print(t)
```

The graphs are obtained as below:





2.5 Creating the function and adding noise:

We add noise to our function created and see the difference.

function = calcnu(x,x_0,eps,model)

Here,

calcnu is our function,

x is our array for which we find bessel function,

$x_0$is our starting point,

eps means our error that is to be added,

model means the model we want to select like the one with amplitude not considered or the one with considered.

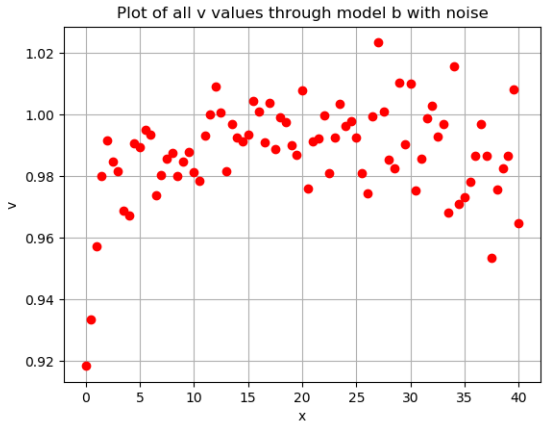
We add the error to our bessel function values using,

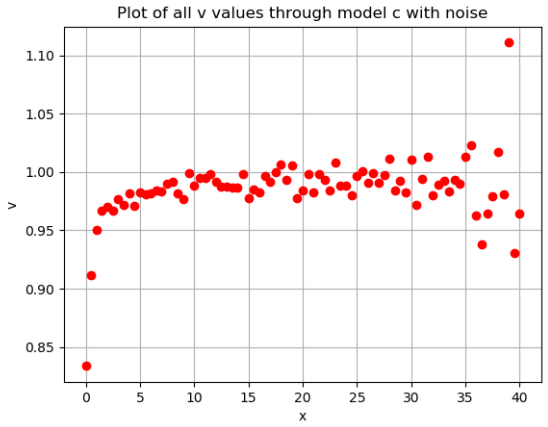
eps*(numpy.random.randn(l)

In the following function, the error terms is
b = spe.jv(1,z) + eps*(numpy.random.randn(l))

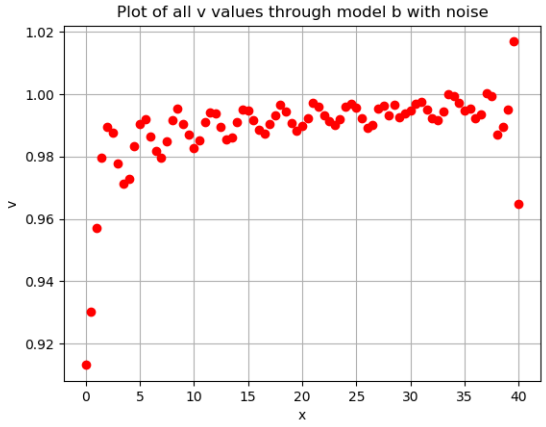
```
# Adding error term

def calcnu(x, x0, eps, model):
        if (model == 1):
                w = list()
                v = list()
                d = []
                q = linspace(x0,40,81)
                        z = numpy.arange(k, x[len(x)-1] + 0.5  ,0.5)
                        l = len(z)
                        P = numpy.zeros((l,2))
                        P[:,0] = numpy.cos(z)
                        P[:,1] = numpy.sin(z)
                        b = spe.jv(1,z) + eps*(numpy.random.randn(l))
                        c = lstsq(P,b,rcond=-1)[0]
                        A = c[0]
                        B = c[1]
                        p = (A/(numpy.sqrt(A**2+B**2)))
                        f = numpy.arccos(p)
                        w.append(f)
                        v = ((2*f)/(numpy.pi)-(1/2))
                        d.append(v)

                plt.figure(1)
                plt.plot(x,d,'ro')
                plt.xlabel('x')
                plt.ylabel('v')
                plt.title('Plot of all v values through model 1')
                plt.grid(True)
                plt.show()
```

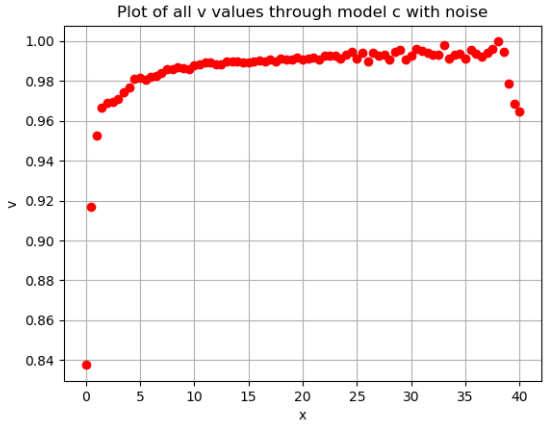
Following two are the graphs with eps = 0.01





Following two are the graphs with eps = 0.001.





Even with a small eps change , there is a significant difference in the graphs and the noise added.

2.6 Effect of model accuracy, number of measurements and effect of noise on quality of fit:

From the above discussion, we come to the following conclusion,

a) The model which takes into account the amplitude gives more accurate results (model 'c') than the one which does not include it (model 'b').

b) We see from the graph that increasing the number of measurements gives more accuracy as we approach to th actual values even more.

c) Adding noise distorts the whole graph and there is a huge loss of accuracy.