Fitting Data to Models

Mohammed Muqeeth
EE16B026
Electrical Engineering Department
IIT Madras

February 28, 2018

Abstract

This report presents linear fitting of data and the effect of noise on fitting process.This report assumes two linear models for bessel function of first type.

1 Introduction

1. Bessel function of first type $J_\nu(x)$ for large x can be approximated as,

$$J_\nu(x) \approx \sqrt{\frac{2}{\pi x}} \cos(x - \frac{\nu\pi}{2} - \frac{\pi}{4}) \tag{1}$$

2. The best fit in the least-squares sense minimizes the sum of squared residuals (a residual being: the difference between an observed value, and the fitted value provided by a model).

3. Two linear models taken for bessel function of first type are :

$$A\cos(x) + B\sin(x) \approx J_1(x) \tag{2}$$

$$A\frac{\cos(x)}{\sqrt{x}} + B\frac{\sin(x)}{\sqrt{x}} \approx J_1(x) \tag{3}$$

4. A,B values for modelA(Eqn 2) , modelB(Eqn 3) are estimated in least square sense.

2 Methods

2.1 Get $J_1(x)$ values which is obtained data

1. Generate a vector x of 41 values from 0 to 20 using linspace.

2. Define a function $jv(x)$ to return $J_1(x)$ vector.

3. Below is python code to get $J_1(x)$ vector :

```
#import required packages
from pylab import *
import matplotlib.pyplot as plt
from numpy import *
import scipy.special as sp
#define bessel function
def jv(x):
```

```
        return sp.jv(1,x)
    #define vector x using linspace
    n = 41 #number of observations
    x = linspace(0,20,n)
```

2.2 Estimation of A,B parameters of a Model and νvalues

1. Take an x_0 from 0.5 to 18 . For each x_0 extract a subvector x where $x \geq x_0$ and find vector $J_1(x)$ for that corresponding vector x.

2. for each x_0 fit vector x into models as :

$$\cos(x).A + \sin(x).B = J_1(x)$$

$$\frac{\cos(x)}{\sqrt{x}}.A + \frac{\sin(x)}{\sqrt{x}}.B = J_1(x)$$

3. This reduces to matrix equation of the form $P.\overrightarrow{a} = \overrightarrow{q}$

$$\begin{pmatrix} \cos(x_1) & \sin(x_1) \\ \cos(x_2) & \sin(x_2) \\ \dots & \dots \\ \cos(x_{41}) & \sin(x_{41}) \end{pmatrix} \begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} J_1(x_1) \\ J_1(x_2) \\ \dots \\ J_1(x_{41}) \end{pmatrix}$$

$$\begin{pmatrix} \frac{\cos(x_1)}{\sqrt{x_1}} & \frac{\sin(x_1)}{\sqrt{x_1}} \\ \frac{\cos(x_2)}{\sqrt{x_2}} & \frac{\sin(x_2)}{\sqrt{x_2}} \\ \dots & \dots \\ \frac{\cos(x_{41})}{\sqrt{x_{41}}} & \frac{\sin(x_{41})}{\sqrt{x_{41}}} \end{pmatrix} \begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} J_1(x_1) \\ J_1(x_2) \\ \dots \\ J_1(x_{41}) \end{pmatrix}$$

4. The vector $\overrightarrow{a}$ is estimated by $lstsq(P, q)$ method of python which essentially does $a = inv(P' * P) * P' * q$

5. After getting A,B parameters calculate ϕ from $\cos(\phi) = \frac{A}{\sqrt{A^2+B^2}}$.The value of ν is calculated by equating $\phi = \frac{\nu\pi}{2} + \frac{\pi}{4}$

6. All the above computation is done by calling a function calcnu defined ourselves which takes complete vector x defined under section 2.1 , x_0,model(whether A or B) and returns ν for each x_0

$$nu = calcnu(x, x0, eps, model)$$

7. The eps argument in above calcnu is discussed under section 2.3. For noise less model take eps = 0

8. Append the nu values returned for each x_0 into a list and plot it versus $x_0 range$ ie from 0.5 to 18 .

```
    def calcnu(x,x0,eps,model):
        indices = where(x>=x0)
        #take from x0 to x
        x = x[indices]
        #get bessel function values in q matrix
        q = jv(x)+ eps*randn(size(x))
        #define matrix P
        P = zeros((len(x),2))
        if(model == 'A'):
```

```
        P[:,0] = cos(x)
        P[:,1] = sin(x)
        A,B=lstsq(P,q)[0]
        phi = arccos(A/sqrt(A*A + B*B))
        nu = 2*(phi-pi/4)/pi
        return nu
    if(model == 'B'):
        P[:,0] = cos(x)/sqrt(x)
        P[:,1] = sin(x)/sqrt(x)
        A,B=lstsq(P,q)[0]
        phi = arccos(A/sqrt(A*A + B*B))
        nu = 2*(phi-pi/4)/pi
        return nu

#define vector x using linspace
n = 41 #number of observations
x = linspace(0,20,n)
#for n=41 x0 ranges from 0.5 to 18 in steps of 0.5
x0_range = linspace(0.5,18,36)
nu_listA =[]
nu_listB =[]
nu_listnoiseB = []
for x0 in x0_range:
    nu_A = calcnu(x,x0,0,'A')
    nu_B = calcnu(x,x0,0,'B')
    nu_noiseB = calcnu(x,x0,0.01,'B')
    nu_listA.append(nu_A)
    nu_listB.append(nu_B)
    nu_listnoiseB.append(nu_noiseB)
```

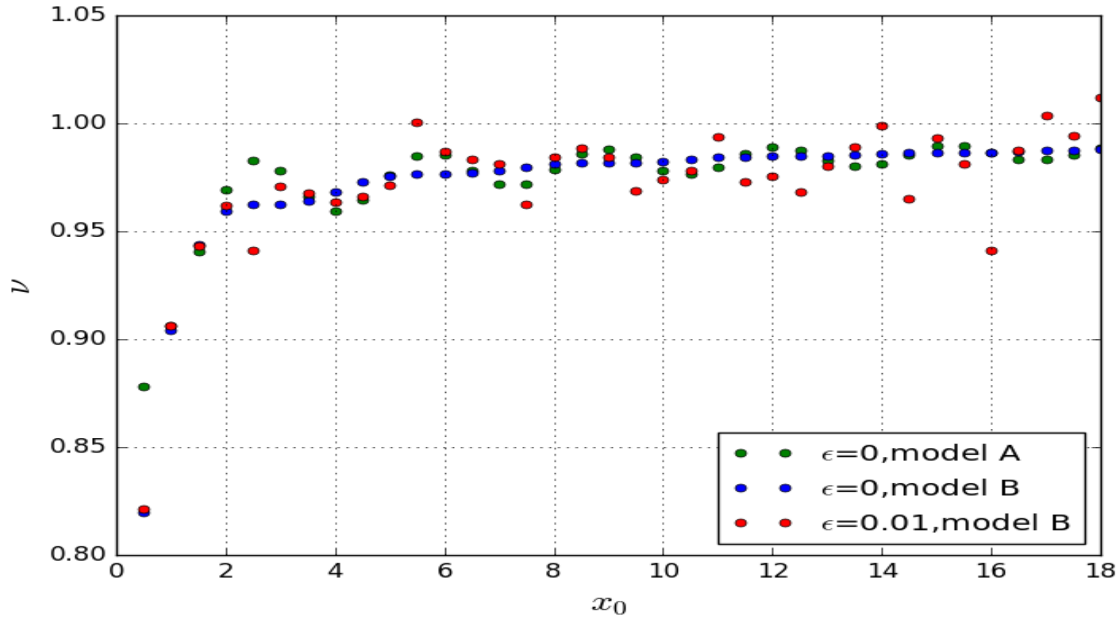
9. The corresponding plots for model A, B are in Figure 1 with blue dots and green dots respectively

2.3 Adding noise to model B

1. The measurements made generally involve noise.
2. To account for noise in model , add $randn(size(x))$ to measured data ie $J_1(x)+eps*randn(size(x))$ where $size(x)$ is number of measurements.
3. This adds normalised noise to measured values with standard deviation of value eps.

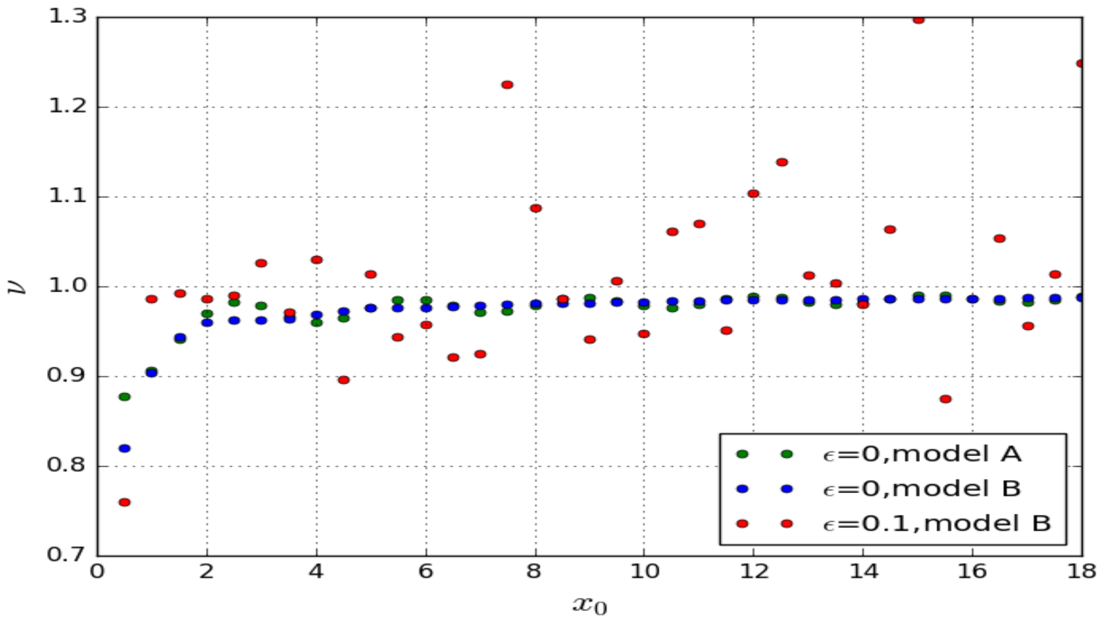
3 Results and discussion

Figure 1: ν vs x_0



1. Model B is better than Model A since it accounts for $\sqrt{x}$ in denominator of amplitude of Eqn 1 . It can be seen from Figure 1.

Figure 2: effect of noise for eps =0.1



2. As noise increases values deviate more from 1 for large x_0 as shown in Figure 2,3.

Figure 3: Effect of noise for eps = 0.05

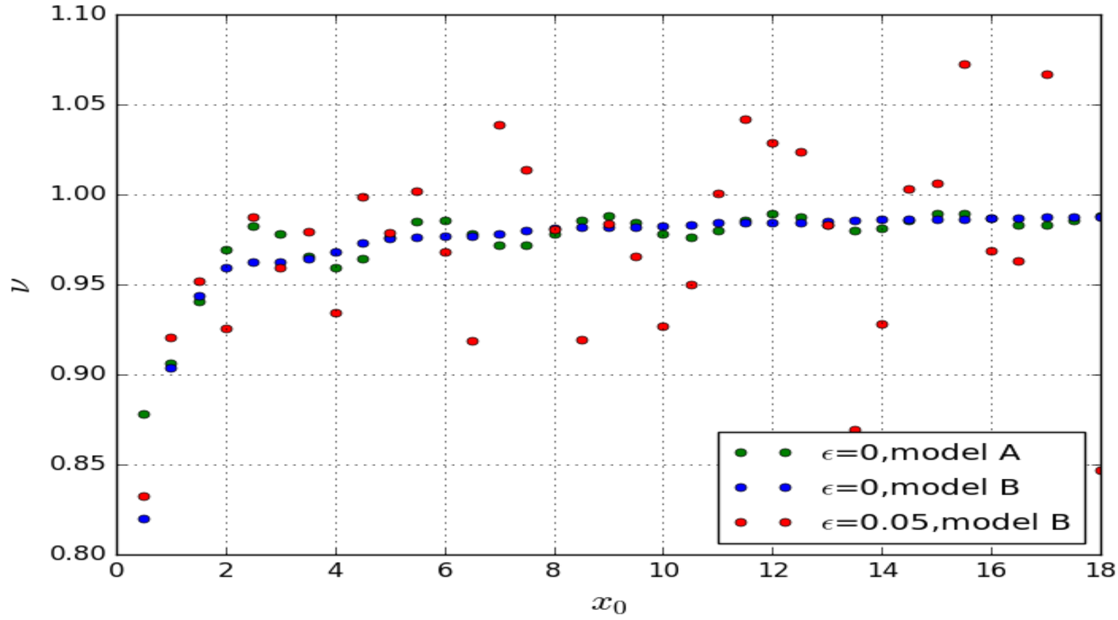


Figure 4: Effect on quality of fit for number of measurements=101

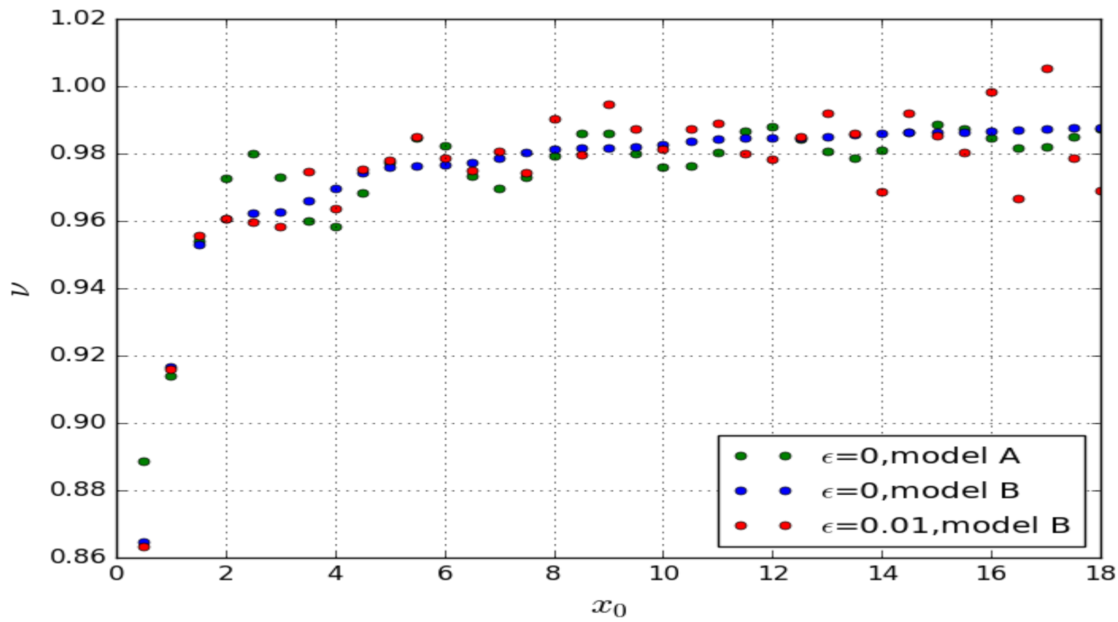


Figure 5: Effect on quality of fit for number of measurements=201

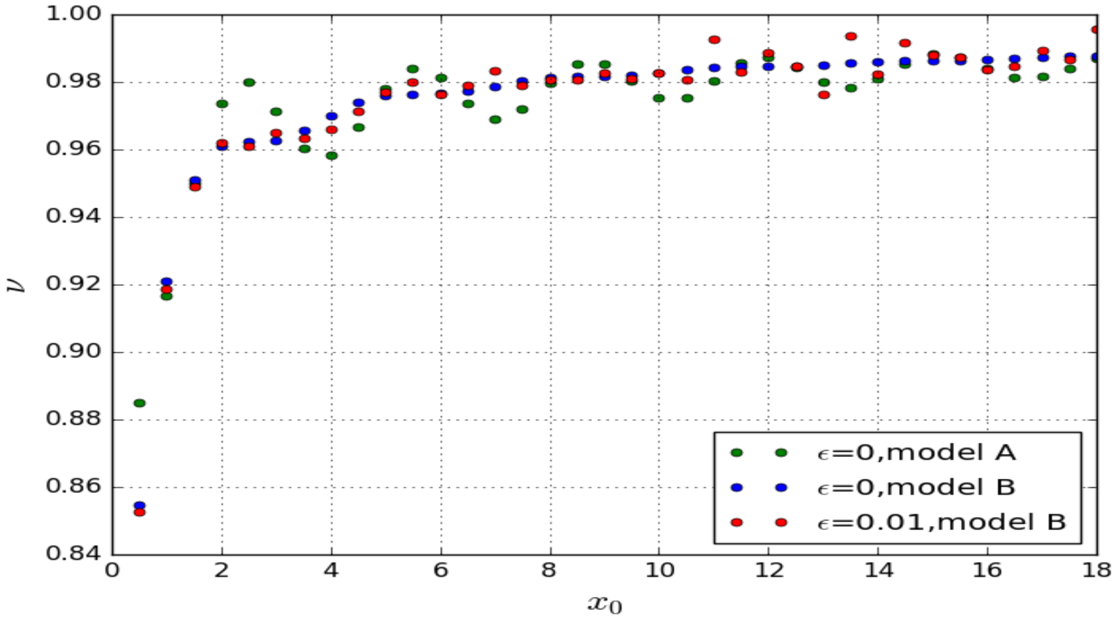


Figure 6: Effect on quality of fit for number of measurements=501

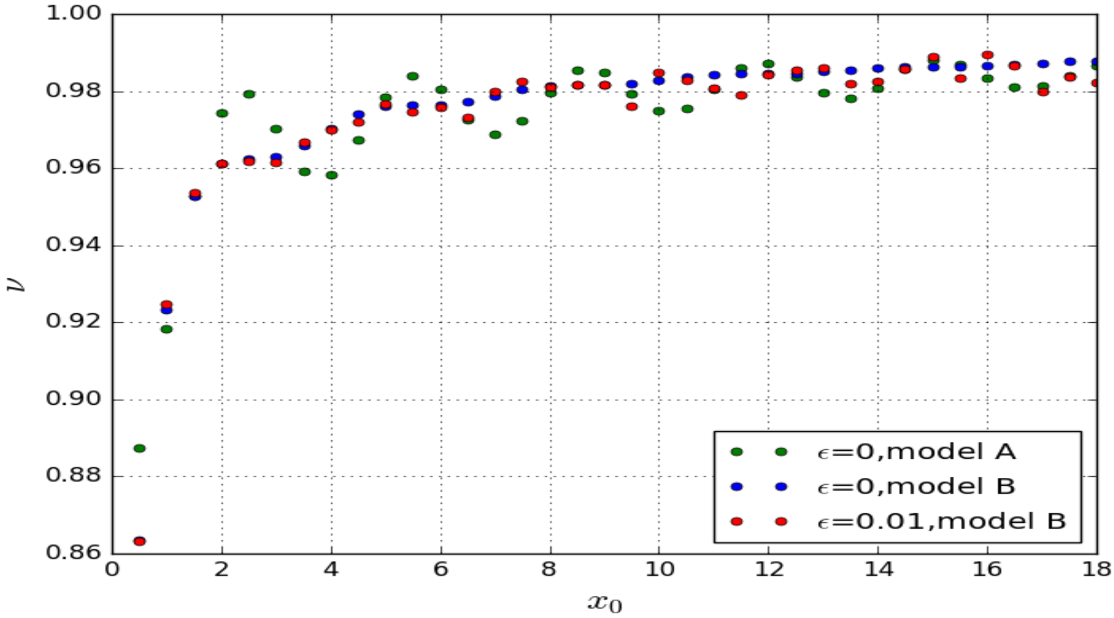
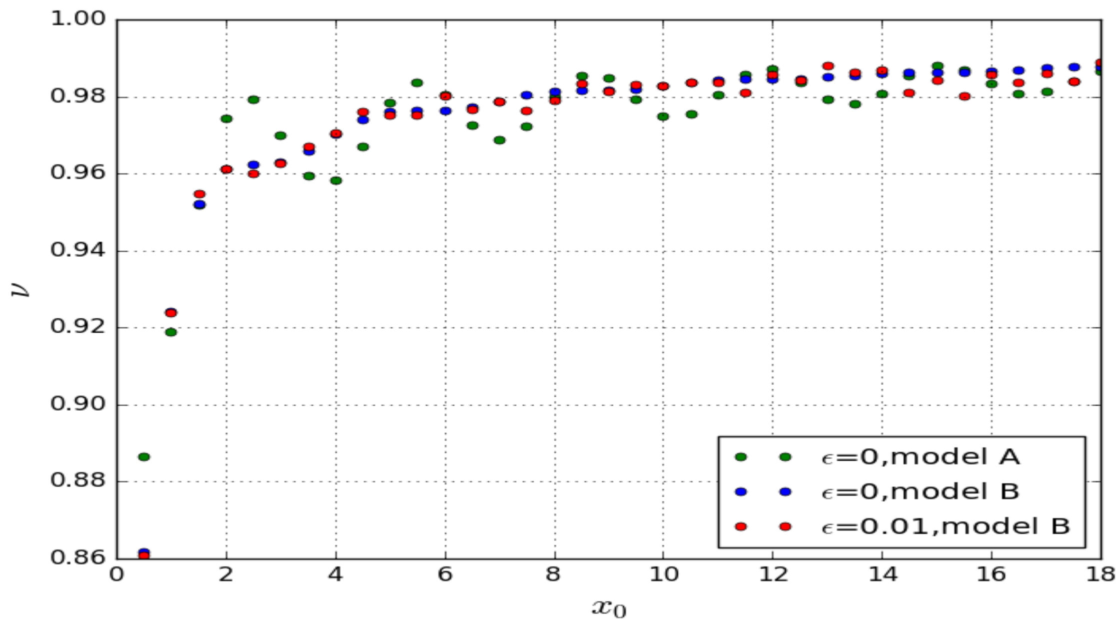


Figure 7: Effect on quality of fit for number of measurements=1001



3. As number of measurements increase from 41 to 1001 for the same range of x, the model with noise matches close to model B without noise as shown in Figure 4,5,6.The models with no noise almost remain the same.

4 Python code :

```
#import required packages
from pylab import *
import matplotlib.pyplot as plt
from numpy import *
import scipy.special as sp
#define bessel function
def jv(x):
    return sp.jv(1,x)
#define calcnu function
def calcnu(x,x0,eps,model):
    indices = where(x>=x0)
    #take from x0 to x
    x = x[indices]
    #get bessel function values in q matrix
    q = jv(x)+ eps*randn(size(x))
    #define matrix P
    P = zeros((len(x),2))
    if(model == 'A'):
        P[:,0] = cos(x)
        P[:,1] = sin(x)
        A,B=lstsq(P,q)[0]
        phi = arccos(A/sqrt(A*A + B*B))
        nu = 2*(phi-pi/4)/pi
        return nu
    if(model == 'B'):
        P[:,0] = cos(x)/sqrt(x)
```

```
        P[:,1] = sin(x)/sqrt(x)
        A,B=lstsq(P,q)[0]
        phi = arccos(A/sqrt(A*A + B*B))
        nu = 2*(phi-pi/4)/pi
        return nu

#define vector x using linspace
n = 41 #number of observations
x = linspace(0,20,n)
#for n=41 x0 ranges from 0.5 to 18 in steps of 0.5
x0_range = linspace(0.5,18,36)
nu_listA =[]
nu_listB =[]
nu_listnoiseB = []
for x0 in x0_range:
    nu_A = calcnu(x,x0,0,'A')
    nu_B = calcnu(x,x0,0,'B')
    nu_noiseB = calcnu(x,x0,0.01,'B')
    nu_listA.append(nu_A)
    nu_listB.append(nu_B)
    nu_listnoiseB.append(nu_noiseB)
plt.xlabel('$x_0$',fontsize = 18)
plt.ylabel(r'$\nu$',fontsize = 18)
plt.plot(x0_range,nu_listA,'go', markersize=5)
plt.plot(x0_range,nu_listB,'bo', markersize=5)
plt.plot(x0_range,nu_listnoiseB,'ro', markersize=5)
plt.legend(['$\epsilon$=0,model A','$\epsilon$=0,model B','$\epsilon$=0.01,model B'],lo
plt.grid(linestyle='dotted')
plt.show()
```