Week 4 : Fitting Data to Models

Ganga Meghanath
EE15B025

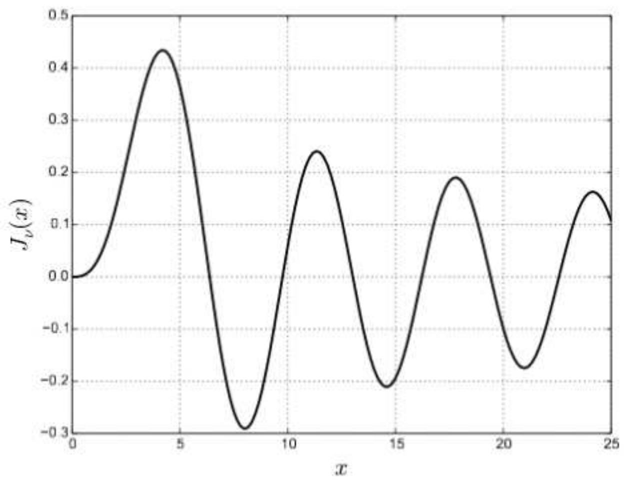
February 6, 2017

Abstract

1.Fitting of data using Models
2.Study the effect of noise on the fitting process

Introduction :

Bessel functions are very often seen in cylindrical geometry. They come in several forms, and we will look at the Bessel function of the first type, J_v(x).



For large x,

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

1. Generate a vector of 41 values from 0 to 20 and obtain a vector of J_1(x) values.

```
def Jv(x) :     #function computes model (b)
        return(((2/(m.pi*x))**(1/2.0))*sp.cos(x-3*m.pi/4))

x=np.linspace(0,20,num=42)
x=x[1:]
print Jv(x)
```

2. For different x_0 = 0.5,1,....,18 extract the subvectors of x and J_1(x) that correspond to x$\geq$ x_0. For each x_0 , construct the matrix corresponding to

$$Acos(x_i) + Bsin(x_i) \approx J_1(x_i) \tag{1}$$

$$A\frac{cos(x_i)}{\sqrt{x_i}} + B\frac{sin(x_i)}{\sqrt{x_i}} \approx J_1(x_i) \tag{2}$$

Obtain the best fit (A,B) . Obtain the ϕ corresponding to the solution (divide by $\sqrt{A^2 + B^2}$ and identify $\frac{A}{\sqrt{A^2+B^2}}$ to be cos f). Hence predict v. Convert the above to a function that can be called via

$$nu = calcnu(x, x_0, \text{'}r\text{'}, eps, model)$$

where eps is the amount of noise added and model is whether to run (1) or (2) above. See the effect of noise on the fit. Plot the fit for eps of 0.01.

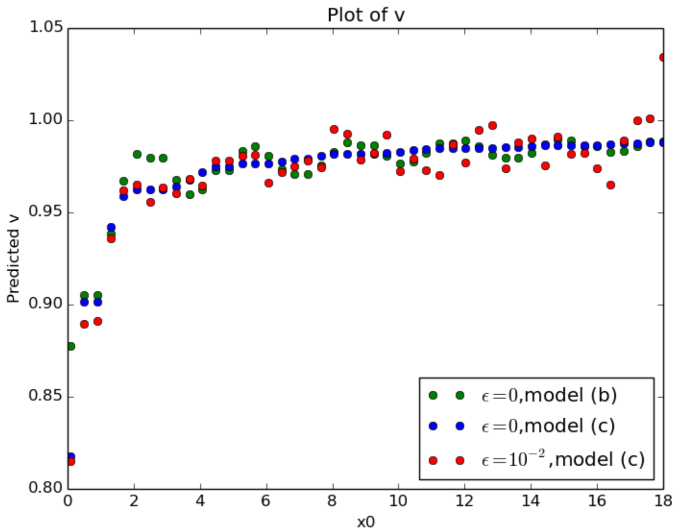
```
def fun(x): #function computes matrix [cosx,sinx]
        Ans=np.zeros((len(x),2))
        Ans[:,0]=sp.cos(x)
        Ans[:,1]=sp.sin(x)
        return Ans

#model a manipulated to get the further models
def calcnu(x,x0,color,eps,model):
        A=np.zeros((41,2))
        v=[]
        for i in x0 :
                y=x[np.where(x>=i)]     #only required x
                A=fun(y)

                if model=='b' :
                        A[:,0]=A[:,0]/((y)**(1/2.0))
                        A[:,1]=A[:,1]/((y)**(1/2.0))

                J=jv(1.0,y)     #in-build bessel function
```

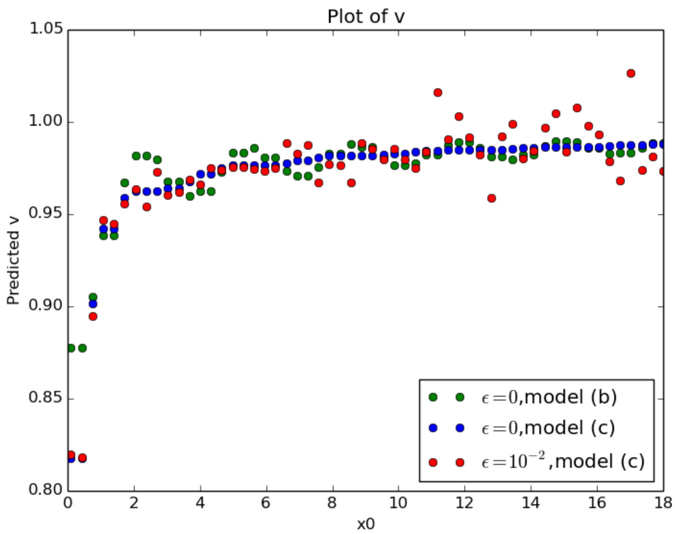
```
                if model=='b' :
                        noise=eps*randn(len(J))
                        J+=noise        #adding noise to J

                B=lstsq(A,J)[0] #computes coefficients

                C=B[0]/((B[0]*B[0] + B[1]*B[1])**(1/2.0))
                C=sp.arccos(C)
                v.append((C-m.pi/4)*2/m.pi)     #computes v

        l, =mp.plot(x0,v,color)
        line.append(l)  #for displaying all plots
```

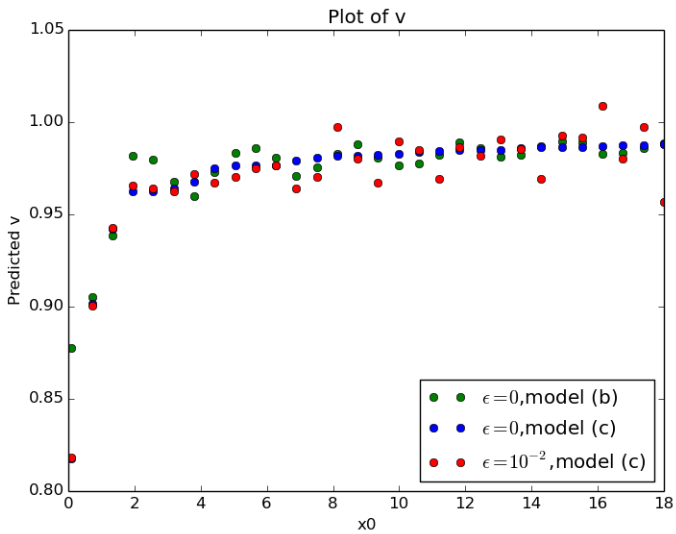
3.Try varying the number of measurements (keeping the range of x the same). What happens?

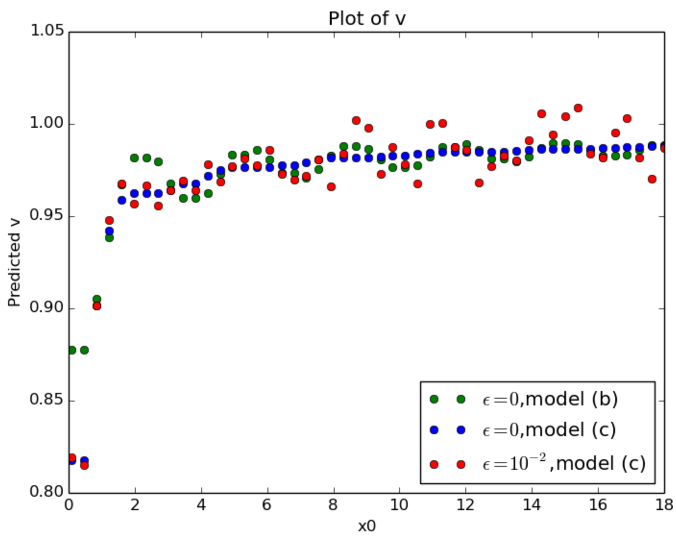
```
for i in range(8): #for varying the measurements
        number=randint(10,60)
        x0=np.linspace(0.1,18,num=number)
        line=[]
        calcnu(x,x0,'go',0,'a')
        calcnu(x,x0,'bo',0,'b')
        calcnu(x,x0,'ro',0.01,'b')

        mp.legend([line[0],line[1],line[2]],\
                [r'$\epsilon =0$,model (b)',r'$\epsilon =0$,model (c)'
                r'$\epsilon=10^{-2}$,model (c)'], loc='lower right')
        mp.xlabel('x0')
        mp.ylabel('Predicted v')
        mp.title('Plot of v')
        mp.savefig('Fig{}.png'.format(i+1))
        mp.show()
```

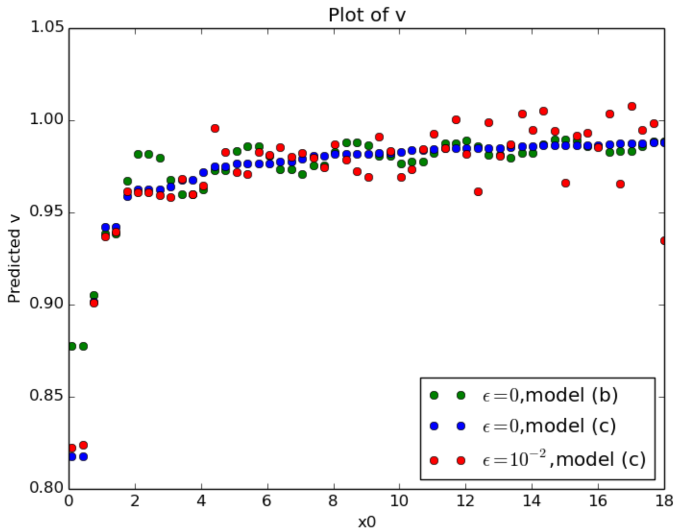
When 8 random number of measurements, the result is as shown below :

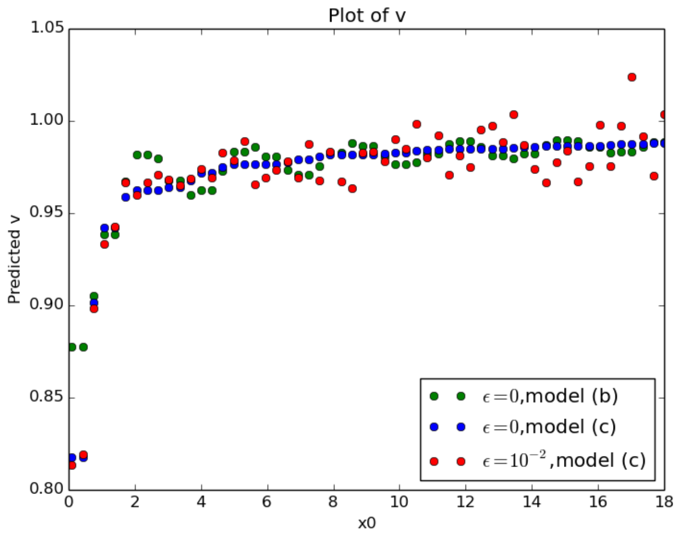

Plot of v
Predicted v
x0
ε=0,model (b)
ε=0,model (c)
ε=10^{-2},model (c)

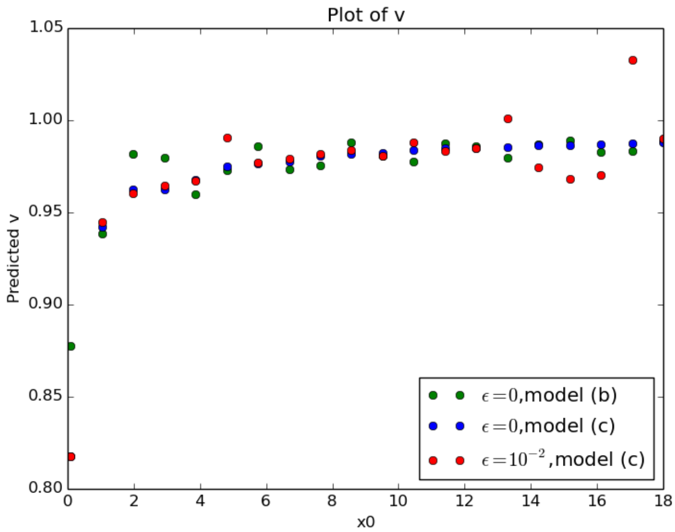

Plot of v
Predicted v
x0
ε=0,model (b)
ε=0,model (c)
ε=10^{-2},model (c)

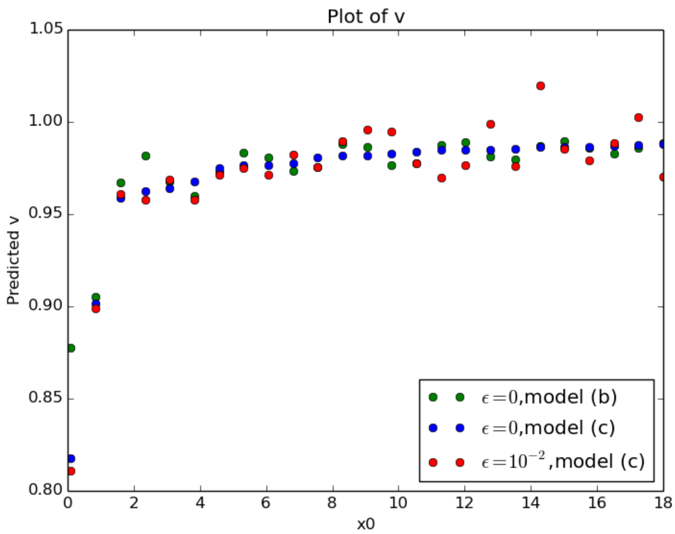

Plot of v
Predicted v
x0
ε=0,model (b)
ε=0,model (c)
ε=10^-2,model (c)
1.05
1.00
0.95
0.90
0.85
0.80
0
2
4
6
8
10
12
14
16
18


Plot of v
Predicted v
x0
ε=0,model (b)
ε=0,model (c)
ε=10^-2,model (c)
1.05
1.00
0.95
0.90
0.85
0.80
0
2
4
6
8
10
12
14
16
18


Plot of v
Predicted v
x0
$\epsilon=0$,model (b)
$\epsilon=0$,model (c)
$\epsilon=10^{-2}$,model (c)


Plot of v
Predicted v
x0
$\epsilon=0$,model (b)
$\epsilon=0$,model (c)
$\epsilon=10^{-2}$,model (c)





4. Discuss the effect of model accuracy, number of measurements, and the effect of noise on the quality of the fit.

As we can infer from the plots, model (c) has more accuracy compared to that of model (b). Noise is an inherant aspect of any real life systems which cannot be avoided fully. And hence, usage of model (c) with noise will be much more advisable than model (b) with noise. Increasing the number of measurements by keeping the range of x constant doesn't seem to be affecting the plots much other than reducing the number of points plotted on the graph. But yes, it does give a more clearer picture of the approach towards v=1 when there are more points plotted. Noise of even $\epsilon = 0.01$ has a lot of effect on the accuracy of the graph. We can see that we get values of v above 1 in the plots. This shows us how noise affects the quality of the fit. The divergence from the actual value seems to increse as x increases meaning larger deviation for larger x though this cannot be generalised. The deviations seems to increase as the calculated value approaches v=1.