EE2703: Applied Programming Lab
Assignment No 3: Fitting Data to Models

Ishaan Agarwal
EE20B046

February 18, 2022

1 Introduction

In this assignment, we are using a special function called the Bessel function along with normally distributed noise added to it to learn to plot different types of plot in python. We also study the effect of changing standard deviation on the best estimates of A and B for the given datasets.
The Bessel function:

$$f(t) = AJ_2(t) + Bt + n(t)$$

2 Questions

2.1 Question 1: Generating Data

On running the given script `generate_data.py`, we get a file `fitting.dat` which consists of all the required data. The first column consists of the time values and the next 9 columns consist of the data values. The data values are calculated by adding Gaussian noise to the Bessel function calculated on the numpy array *time*.

2.2 Question 2: Extracting the Data

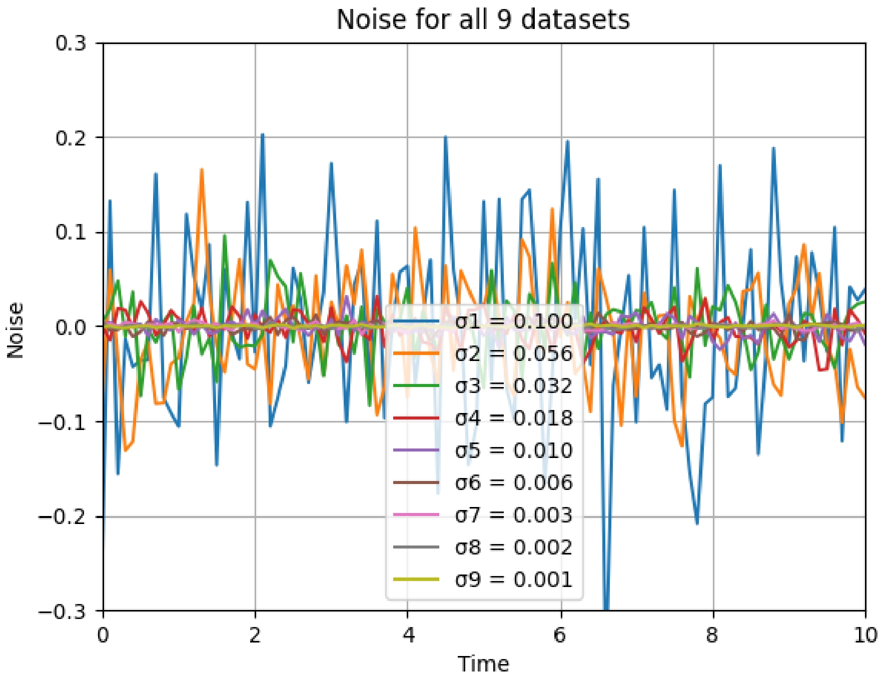
The data from `fitting.dat` is loaded into a 2D numpy array *arr*. The first column consisting of the time values is loaded into a 1D numpy array *time* and the remaining 9 columns are loaded into a 2D numpy array *data*.

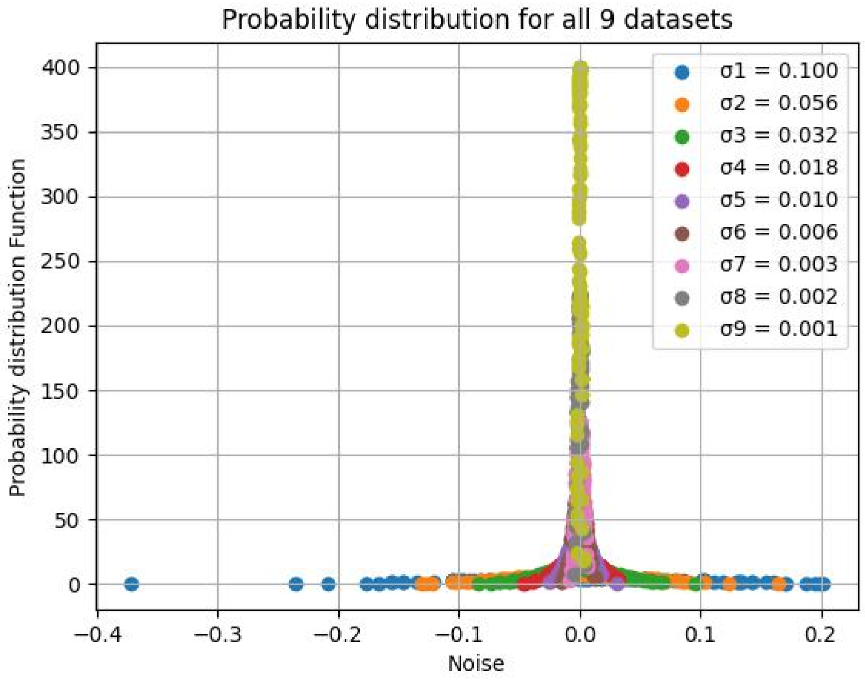
```
arr = np.loadtxt("fitting.dat")
time = np.loadtxt("fitting.dat", usecols=0)
data = np.loadtxt("fitting.dat", usecols=range(1,10))
```

2.3 Question 3: Extracting and observing noise and noise distribution

The noise is extracted using $n(t) = f(t) - AJ_2(t) - Bt$ and plotted using `plt.plot` and the Gaussian distribution for the different noises is also plotted using `plt.scatter`.

```python
#plotting the noise
noise = []
for i in range(len(data[0])):
    noise.append(data[:,i] - 1.05*sp.jn(2, time)+0.105*time)
    plt.plot(time, noise[i])
# print(shape(noise)) 9*101
plt.title("Noise for all 9 datasets")
plt.xlabel("Time")
plt.ylabel("Noise")
plt.legend(["$\sigma$1 = 0.100", "$\sigma$2 = 0.056",
 "$\sigma$3 = 0.032", "$\sigma$4 = 0.018", "$\sigma$5 = 0.010",
  "$\sigma$6 = 0.006", "$\sigma$7 = 0.003", "$\sigma$8 = 0.002",
   "$\sigma$9 = 0.001"])
plt.axis([0, 10, -0.3, 0.3])
plt.grid()
plt.show()

#plotting the probability distribution
noise_dist = []
sigma = logspace(-1, -3, 9)
j=0

for i in sigma:
    noise_dist.append(np.exp(-(noise[j]**2)/(2*i**2))/(i*np.sqrt(2*np.pi)
    plt.scatter(noise[j], noise_dist[j])
    j+=1
plt.title("Probability distribution for all 9 datasets")
plt.xlabel("Noise")
plt.ylabel("Probability distribution Function")
plt.legend(["$\sigma$1 = 0.100", "$\sigma$2 = 0.056",
 "$\sigma$3 = 0.032", "$\sigma$4 = 0.018", "$\sigma$5 = 0.010",
  "$\sigma$6 = 0.006", "$\sigma$7 = 0.003", "$\sigma$8 = 0.002",
   "$\sigma$9 = 0.001"])
plt.grid()
plt.show()
```


Noise for all 9 datasets
Noise
Time
σ1 = 0.100
σ2 = 0.056
σ3 = 0.032
σ4 = 0.018
σ5 = 0.010
σ6 = 0.006
σ7 = 0.003
σ8 = 0.002
σ9 = 0.001


Probability distribution for all 9 datasets
Probability distribution Function
Noise
σ1 = 0.100
σ2 = 0.056
σ3 = 0.032
σ4 = 0.018
σ5 = 0.010
σ6 = 0.006
σ7 = 0.003
σ8 = 0.002
σ9 = 0.001

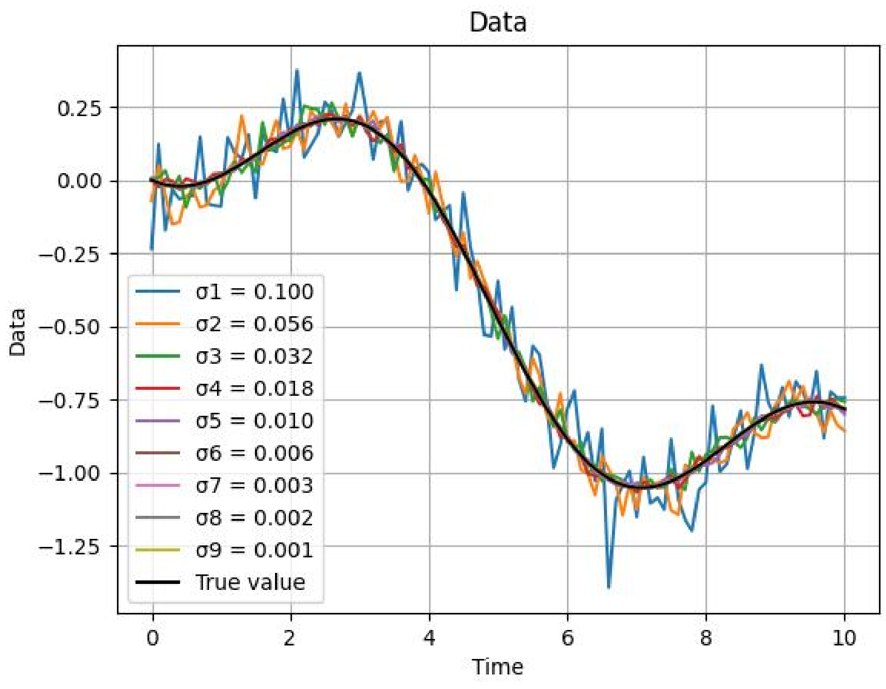
2.4 Question 4: Defining Bessel Function

We define a new python function to compute the Bessel function for any given A, B, given by:

$$g(t; A, B) = AJ_2(t) + Bt$$

The true value is given by passing the *time* array into this function for the true values of A and B given by $A = 1.05$ and $B = -0.105$.
All the datasets are plotted along with the true value computed above in black.

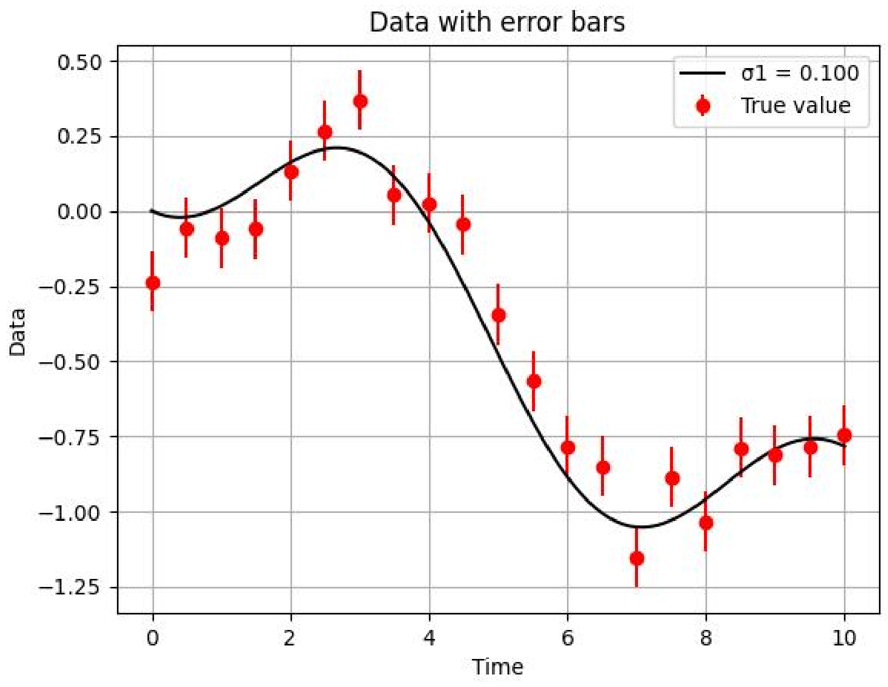
```
#Creating Function to fit the data
def g(t, A, B):
    return A*sp.jn(2, t) + B*t

true_A = 1.05
true_B = -0.105

#plotting the data
for i in range(len(data[0])):
    plt.plot(time, data[:,i])
plt.plot(time, g(time, 1.05, -0.105), 'k')
plt.title("Data")
plt.xlabel("Time")
plt.ylabel("Data")
plt.legend(["$\sigma$1 = 0.100", "$\sigma$2 = 0.056",
 "$\sigma$3 = 0.032", "$\sigma$4 = 0.018", "$\sigma$5 = 0.010",
  "$\sigma$6 = 0.006", "$\sigma$7 = 0.003", "$\sigma$8 = 0.002",
   "$\sigma$9 = 0.001", "True Value"])
plt.grid()
plt.show()
```



2.5 Question 5: Plotting with error bars

Here, we plot every 5th data item of the first column data along with the true value to get an idea of how much the data diverges, using `plt.errorbar`.

```
#Plotting with error bars
plt.errorbar(time[::5], data[::5,0], yerr=0.1, fmt='ro')
plt.plot(time, g(time, 1.05, -0.105), 'k')
plt.title("Data with error bars")
plt.xlabel("Time")
plt.ylabel("Data")
plt.legend(["$\sigma$1 = 0.100", "True value"])
plt.grid()
plt.show()
```



2.6 Question 6: Constructing Matrices and verifying

The vectors M and p are constructed using `c_[x, time]` and `np.array` respectively. The product $M * p$ needs to be equal to the true value of the Bessel function computed on *time* using true values of A and B.
To check if two vectors made of floating point values are equal, we use `numpy.allclose()` which returns `True` if all elements are equal within a tolerance.

```
#Creating Matrix Equation and verifying the solution
x = sp.jn(2, time)
M = c_[x, time]
p = np.array([1.05, -0.105])
if np.allclose(np.matmul(M, p),g(time, 1.05, -0.105)):
    print("Matrix Equation is correct \n")
else:
    print("Matrix Equation is incorrect \n")
```

2.7 Question 7: Computing Mean Squared Error

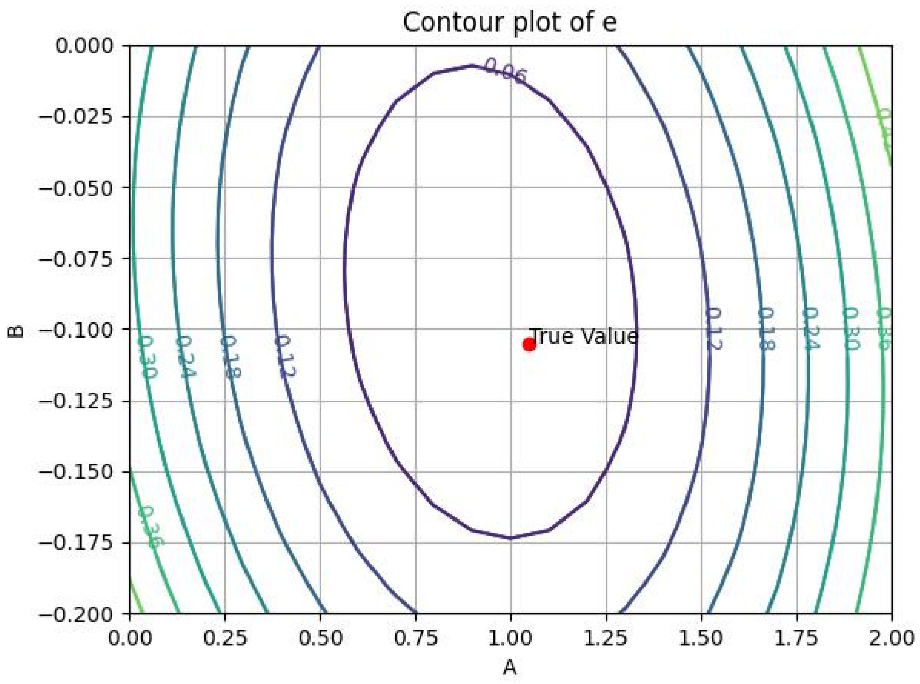
The mean squared error between the given noisy data and the true model is calculated for every value of A and B and the results are stored in an error matrix e.

```
#Computing mean squared errors
A = np.linspace(0, 2, 21)
B = np.linspace(-0.2, 0, 21)
e = np.zeros((len(A), len(B)))
for i in range(len(A)):
    for j in range(len(B)):
        e[i,j] = np.mean((g(time, A[i], B[j]) - data[:,0])**2)
```

2.8 Question 8: Contour Plot

The error e is plotted on meshgrid made out of all values of A and B and a contour plot is plotted and labelled. The true value of A and B is also plotted and annotated.

```
#Plotting a contour plot of e
X, Y = np.meshgrid(A, B) #creating a grid
plt.contour(X, Y, e)
plt.clabel(plt.contour(X, Y, e), fontsize=10) #labelling the contour plot
plt.plot(true_A, true_B, 'ro')
plt.annotate("True␣Value", (true_A, true_B))
plt.title("Contour␣plot␣of␣e")
plt.xlabel("A")
plt.ylabel("B")
plt.grid()
plt.show()
```



2.9 Question 9: Best estimate of A and B

The best estimate of A and B is calculated using `np.linalg.lstsq` and using the matrices M and the given data matrix i.e *data*.

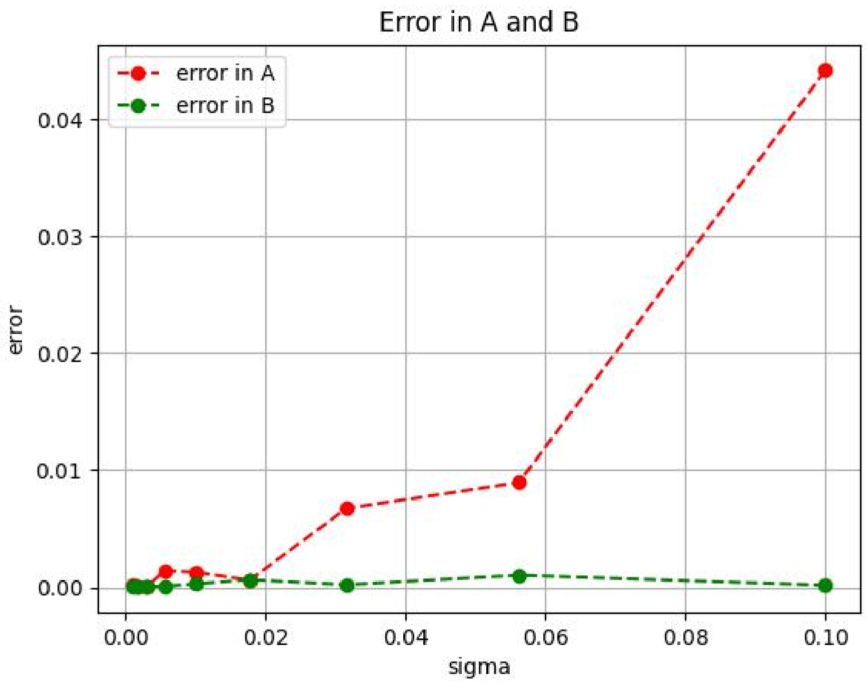
```
#To obtain the best estimate of A and B using lstsq
Best = np.linalg.lstsq(M, data)[0]
A_best = Best[0]
B_best = Best[1]
```

2.10 Question 10: Repeating for all datasets

The errors in A and B have been calculated by subtracting the best A and best B for the corresponding dataset from the true value of A and true value of B. These error values are then plotted against the standard deviation of the corresponding datasets.

```
#Plotting the errors for different datasets

error_A_best = abs(A_best - true_A) #finding the absolute value of the di
between the best estimate of A and the true value of A
error_B_best = abs(B_best - true_B) #finding the absolute value of the di
```

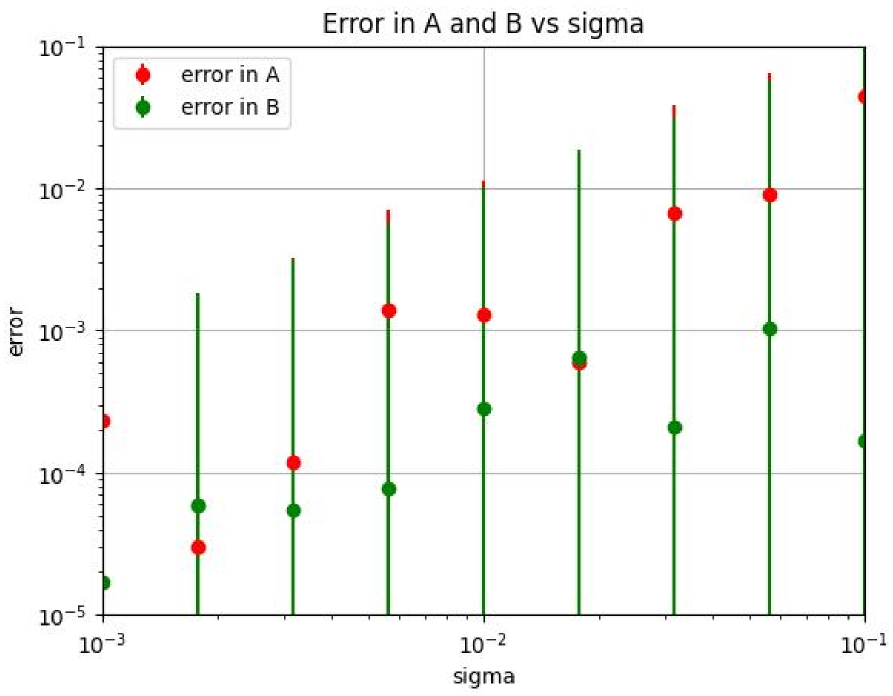
```
between the best estimate of B and the true value of B

plt.plot(sigma, error_A_best, 'ro', linestyle = '--')
plt.plot(sigma, error_B_best, 'go', linestyle = '--')
plt.legend(["error in A", "error in B"])
plt.title("Error in A and B")
plt.xlabel("sigma")
plt.ylabel("error")
plt.grid()
plt.show()
```



2.11 Question 11: Repeating the above plot on a log-log scale

The above plot is replotted as an errorbar plot with errorbar being the standard deviation of the corresponding dataset on a log-log scale.

```
#Plotting loglog curves for the errors vs noise
plt.errorbar(sigma, error_A_best, yerr = sigma, fmt='ro') #error bars for
plt.errorbar(sigma, error_B_best, yerr = sigma, fmt='go') #error bars for
plt.axis([1e-3, .1, 1e-5, 0.1])
plt.xscale('log')
plt.yscale('log')
plt.legend(["error in A", "error in B"])
plt.title("Error in A and B vs sigma")
plt.xlabel("sigma")
plt.ylabel("error")
plt.grid()
plt.show()
```



3 Conclusion

On observing the plots, we notice that the errors in A and B values increase as the noise becomes more diverging i.e the standard deviation of the noise increases. Also, the increase is somewhat linear when plotted on a log-log scale.