EE2703 Assignment 3

Nihal John George (EE19B131)

March 3, 2021

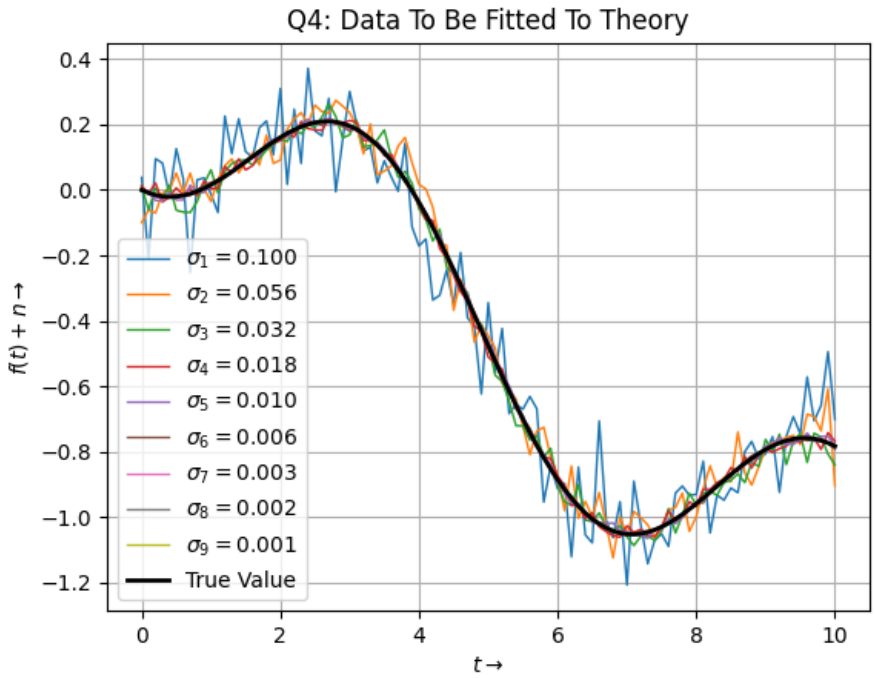
Abstract

This assignment was an exploration of the least squares estimation method, using PyLab and Scipy in Python. First, *generate_data.py* was run, to generate noisy data modelled using Bessel function $J_2(t)$ and Gaussian noise. The noisy data was plotted along with the pure function along with errorbars. Then, the mean-squared error (MSE) of the noisy data vs. $AJ_2(t)+Bt$ was computed for various values of A and B. A contour was plotted for MSE vs. A and B. This revealed a single minimum. The *lstsq* function was used to find A and B that minimised MSE. The error of the estimated A and B vs. noise standard deviation was plotted on linear-linear and log-log scales.

The detailed process and observations from each of the plots is given below.

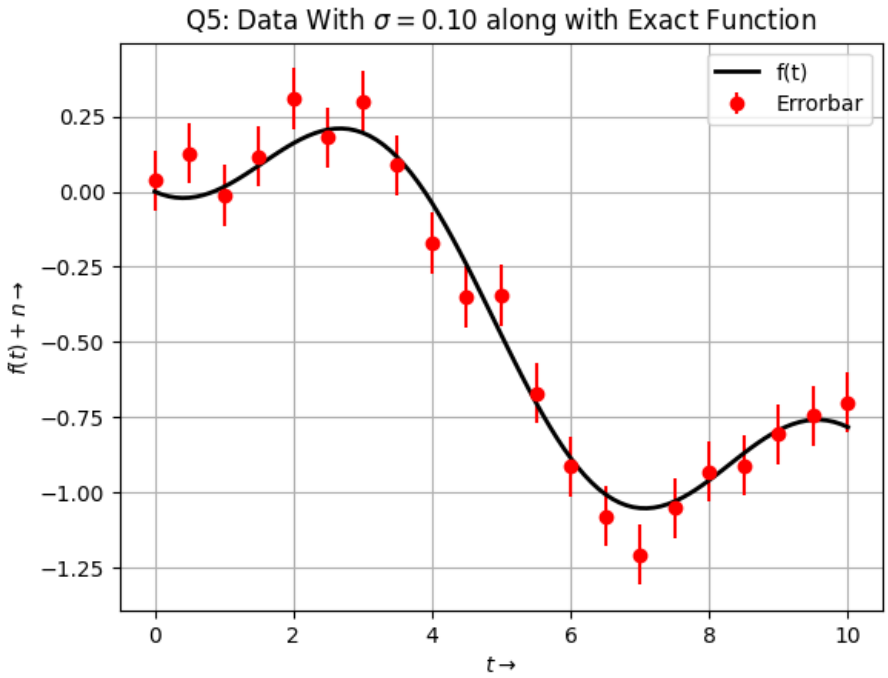
1 Generation and Visualization

Firstly, *generate_data.py* was used to generate noisy data modelled using the Bessel function $J_2(t)$ and Gaussian noise of different standard deviations σ. The pure function was defined as a Python function $g(t, A, B)$ that returns the Bessel given the time instants, A and B. The pure $AJ_2(t) + B$ at $A = 1.05, B = -0.105$ as well as the noisy data at different σ are plotted below.



2 Noise with Errorbars

By plotting errorbars, the noise of every 5th datapoint in the first data column is visualized against the pure function.



3 Matrix Formulation of g(t,A,B)

$g(t, A, B)$ was constructed in the matrix form as $M \cdot p$, where M had 2 column vectors $(J_2(t_i), t_i)$, and p was had two column vectors A and B in the form $(A, B)^T$. This formulation is confirmed to be equal by comparing $M \cdot p$ with $g(t, A, B)$ using the Numpy function $array\_equal()$.

```
M = c_[g(t, 1, 0), t]
p = array([A0,B0])
g_Mp = dot(M,p)
assert array_equal(g_Mp, g(t,A0,B0)), "g(t,A,B) and M.p do not match"
```

4 Mean Squared Error

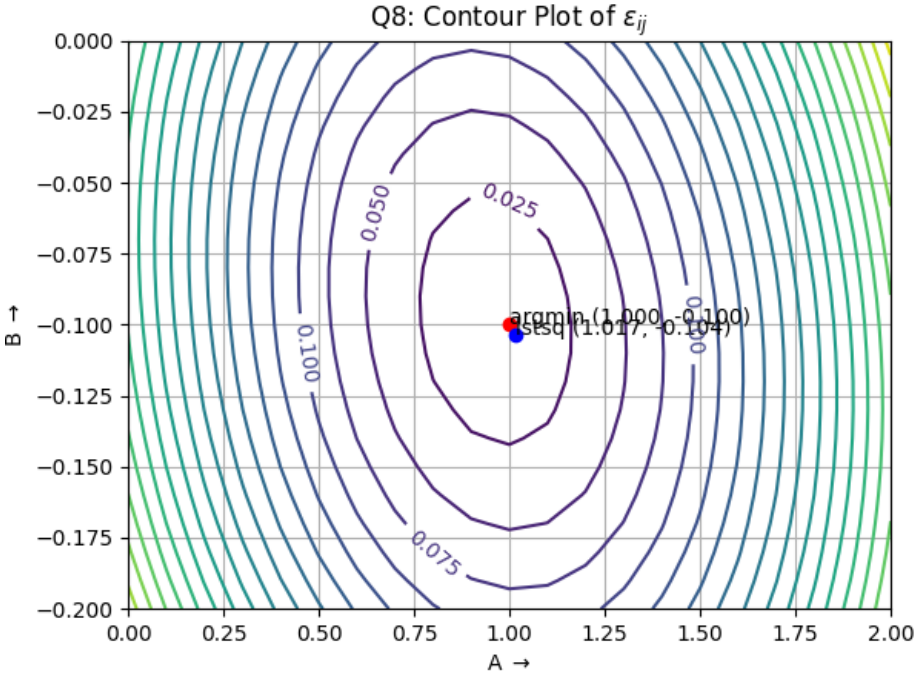
The MSE ε_{ij} of the noisy data versus the pure function is calculated for various A and B. The calculation is done using the code below.

```
fk = y[:, 0]
mse = np.array([[mean(square(fk-g(t,i,j))) for j in B] for i in A])
```

The *argmin* of the MSE matrix is used as another estimate of the best A and B.

```
ind = unravel_index(argmin(mse, axis=None), mse.shape)
plot(A[ind[0]], B[ind[1]], 'ro')
```

A contour plot of ε_{ij} was plotted. The uniform decrease in the contour towards the centre indicates a single minimum. The *lstsq* estimate (blue point) and the *argmin* A and B (red point) of the MSE matrix are plotted below. There is a discrepancy in the values, which could be due to the limited resolution of A and B values in the MSE matrix.



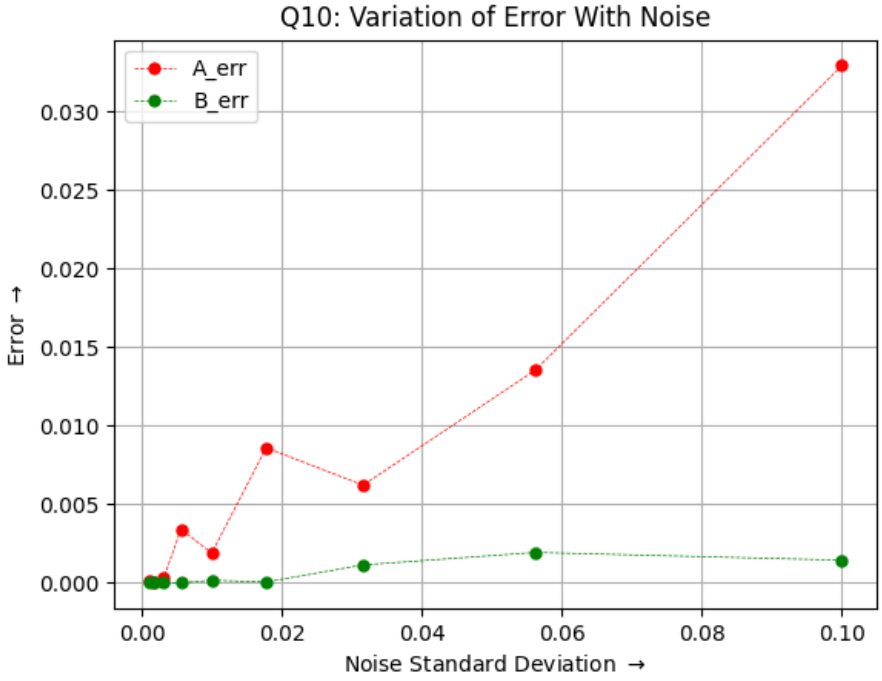
5 Least Square Estimation

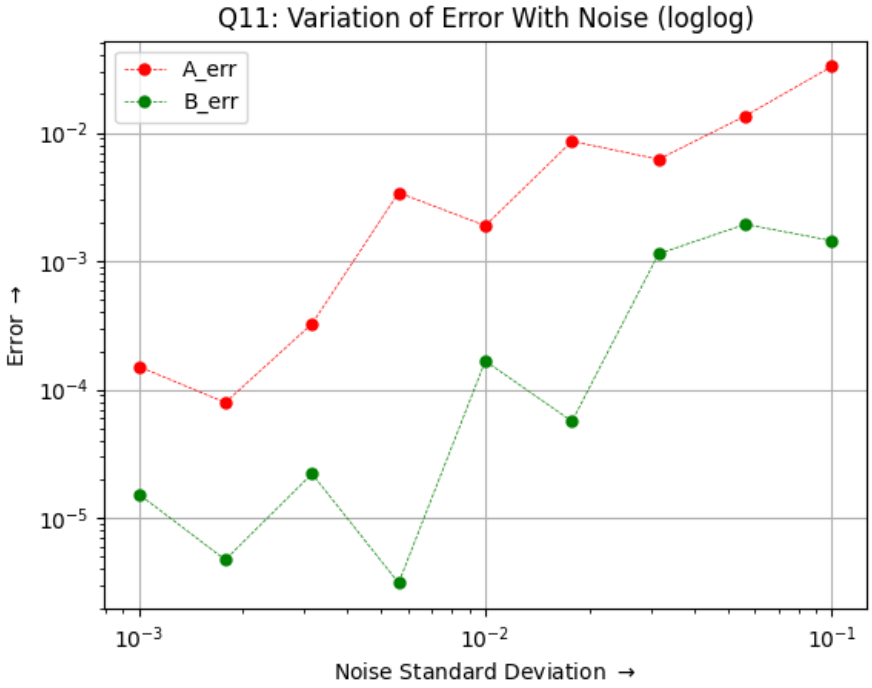
Using the *lstsq* function, the values of A and B are estimated. The function solves for x the equation $Mx = b$ where M may be over-determined (like in our case). In our case, M is as defined before, b is the first data column, while x is a 2-by-1 vector of A and B.

```
ls_est = lstsq(M,b,rcond=None)
A_est = ls_est[0][0]
B_est = ls_est[0][1]
plot(A_est,B_est, 'bo')
```

6 Error Variation with Noise σ

The absolute error of the estimated A and B from 1.05 and -0.105 are plotted on both linear-linear and log-log scales. The lin-lin plot shows A increasing faster than a linear function. The log-log plot shows an approximate straight line relation. The linear relation between error in A and B vs. σ is revealed in the log-log plot since our sigma values were logspaced, so the log-log plot is the logical choice.





7 Conclusion

The parameters of a linear combination of functions have been estimated, and an investigation of the estimation error versus the noise σ has been done. The investigation revealed a linear relation between the estimation error and σ.
Through this assignment, the student has been acquainted with Numpy, Scipy and Matplotlib for the least squares regression task. A variety of plots like lines, errorbars and contours were used to effectively convey the key ideas.