Lyapunov Exponent Mackey Glass

January 9, 2020

0.1 The following method is implemented based on the understanding of the mail and from the documnet.

```
[27]: import pandas as pd
      import nolds
      import numpy as np
      from math import sqrt
```

0.1.1 df_without_noise is the set of predicted values from 1500 to 1800 time steps with out adding noise

##Step 2: for the previous mail

```
[28]: df_without_noise =  pd.read_excel(r'C:\Users\INFO-DSK-02\Desktop\Lorentz Multi␣
       ↪Dimension␣
       ↪Prediction-Phase-2\Final_Version\3D_ReservoirComputing\Output\MC_Data\MG_Output_without_Noi
       ↪xlsx', index = False)
```

```
[29]: df_without_noise.head()
```

```
[29]:      X_pred    X_test  Test_T
      0  1.109942  1.110668   151.0
      1  1.112352  1.113016   151.1
      2  1.114706  1.115310   151.2
      3  1.117005  1.117551   151.3
      4  1.119249  1.119739   151.4
```

0.1.2 df_noise is the set of predicted values with noise from 1500 to 1800

Step3 and step4 from the mail

```
[30]: df_noise =  pd.read_excel(r'C:\Users\INFO-DSK-02\Desktop\Lorentz Multi␣
       ↪Dimension␣
       ↪Prediction-Phase-2\Final_Version\3D_ReservoirComputing\Output\MC_Data\MG_Output_with_Noise.
       ↪xlsx', index = False)
```

```
[31]: df_without_noise.head()
```

[31]:

```
      X_pred    X_test  Test_T
0  1.109942  1.110668   151.0
1  1.112352  1.113016   151.1
2  1.114706  1.115310   151.2
3  1.117005  1.117551   151.3
4  1.119249  1.119739   151.4
```

Here we are calculating the difference between noise and with out noise predicted values. The difference is squared and then applied log for each of these values.

step 5 from the mail

[32]:

```
x_diff = np.log(np.sqrt((df_noise.X_pred.values- df_without_noise.X_pred.
 ↪values)**2))
time = df_noise['Test_T'].values
```

step : from the mail

[33]:

```
len(time)
```

[33]:

```
317
```

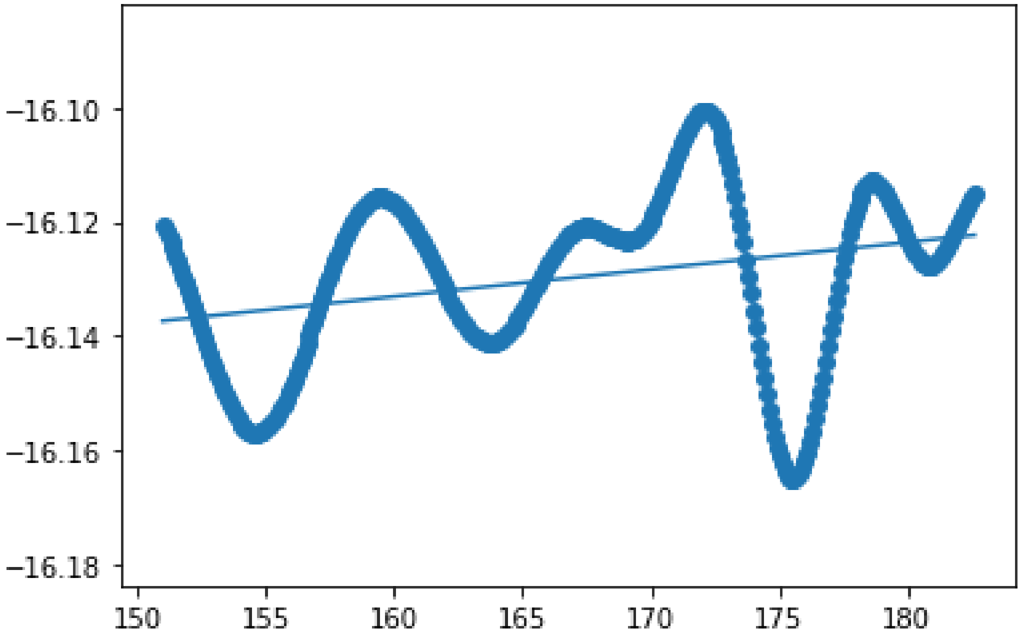
[34]:

```
# sample points
X = time
Y = x_diff

# solve for a and b
def best_fit(X, Y):

    xbar = sum(X)/len(X)
    ybar = sum(Y)/len(Y)
    n = len(X) # or len(Y)

    numer = sum([xi*yi for xi,yi in zip(X, Y)]) - n * xbar * ybar
    denum = sum([xi**2 for xi in X]) - n * xbar**2

    b = numer / denum
    a = ybar - b * xbar

    print('best fit line:\ny = {:.2f} + {:.8f}x'.format(a, b))

    return a, b

# solution
a, b = best_fit(X, Y)
#best fit line:
#y = 0.80 + 0.92x
```

```python
# plot points and fit line
import matplotlib.pyplot as plt
plt.scatter(X, Y)
yfit = [a + b * xi for xi in X]
plt.plot(X, yfit)
```

```
best fit line:
y = -16.21 + 0.00047514x
```

```
[34]: [<matplotlib.lines.Line2D at 0x18c40cdc668>]
```



1 L.E. = 0.00047514