#**Modeling and Analysis of Micro-Scale Mechanical System with Machine Learning Integration**  
import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

# Charaterization of on Chip CMOS-MEMS Eletro-thermal (ET) Actuator

# Perameter Initialization in micron

dl = 755 \* 10\*\*(-6) # Length of Device

dw = 680 \* 10\*\*(-6) # Width of Device

lb = 300 \* 10\*\*(-6) # Long beams Length

wb = 3 \* 10\*\*(-6) # Long beams width

tb = 4.5 \* 10\*\*(-6) # Long beams thickness

lshuttle = 155 \* 10\*\*(-6) # shuttle Length

wshuttle = 100 \* 10\*\*(-6) # shuttle Width

tshuttle = 4.5 \* 10\*\*(-6) # shuttle thickness

lfinger = 60 \* 10\*\*(-6) # shuttle Fingers length

wfinger = 3 \* 10\*\*(-6) # shuttle Fingers width

tfinger = 4.5 \* 10\*\*(-6) # shuttle Fingers thickness

lstator\_fing = 60 \* 10\*\*(-6) # stator Fingers length

wstator\_fing = 3 \* 10\*\*(-6) # stator Fingers width

tstator\_fing = 4.5 \* 10\*\*(-6) # stator Fingers thickness

Aetch\_hole = 10 \* 10 \* 10\*\*(-12) # Area of etched hole

mshuttle = 541.29 \* (10\*\*(-12)) # Mass of shuttle

# Material Properties

E\_SiO2 = 73 \* 10\*\*9 # Young's Modulus of Silicon dioxide

E\_Si = 130 \* 10\*\*9 # Young's Modulus of Silicon

E\_Al = 77 \* 10\*\*9 # Young's Modulus of Aluminum

E\_Sipoly = 169 \* 10\*\*9 # Young's Modulus of Poly Silicon

P\_SiO2 = 2648 # Density of Silicon dioxide

P\_Si = 2328 # Density of Silicon

P\_Al = 2700 # Density of Aluminum

P\_Sipoly = 2300 # Density of Poly Silicon

# Material Thickness with Silicon Substrate

tSi = 60 \* 10\*\*(-6) # Thickness of Si

tSiO2 = 0.710 \* 10\*\*(-6) # Thickness of SiO2

tAl1 = 0.624 \* 10\*\*(-6) # Thickness of Metal 1 Layers

tAl2 = 0.612 \* 10\*\*(-6) # Thickness of Metal 2 Layers

tAl3 = 0.877 \* 10\*\*(-6) # Thickness of Metal 3 Layers

tAl = tAl1 + tAl2 + tAl3 # Thickness of all three Al Layers

tM1\_M2 = 0.720 \* 10\*\*(-6) # Dielectric Thickness from M1 to M2

tM2\_M3 = 0.660 \* 10\*\*(-6) # Dielectric Thickness from M2 to M3

tPoly = 0.285 \* 10\*\*(-6)

t\_T = tSi + tSiO2 + tAl + tM1\_M2 + tM2\_M3 + tPoly # Total thickness With Silicon

# Spring Constant or Stiffness

E = (E\_Al + E\_SiO2) / 2 # Average of Young's Modulus of Aluminum and Polysilicon

N = 4 # Total number of beams

ky = N \* ((E \* tb \* (wb)\*\*3) / (lb)\*\*3)

# Shuttle Volume

l\_rod = 310 \* 10\*\*(-6) # Length of upper rod

w\_rod = 30 \* 10\*\*(-6) # Width of lower rod

Area\_rod1 = l\_rod \* w\_rod # Upper Rod Area

Area\_rod2 = l\_rod \* w\_rod # Lower Rod Area

C\_lshuttle = 155 \* 10\*\*(-6) # Central Shuttle Length

C\_wshuttle = 100 \* 10\*\*(-6) # Central Shuttle Width

C\_area = C\_lshuttle \* C\_wshuttle # Central Shuttle Area

tAetch\_hole = 86 \* Aetch\_hole # Total area of 86 holes

Net\_area\_rotor = (Area\_rod1 + C\_area + Area\_rod2) - (tAetch\_hole)

# Net area of Rotor Part

Vol\_rotor = Net\_area\_rotor \* tshuttle # Volume of Rotor without finger

Vol\_finger = lfinger \* wfinger \* tfinger # Volume of one finger

Vol\_tfingers = 52 \* Vol\_finger # Volume of 52 fingers

Vol\_shuttle = Vol\_rotor + Vol\_tfingers # Volume of shuttle with fingers

# Volume of Beams

Vol\_beam1 = lb \* wb \* tb # Volume of one beam

Vol\_beams = 4 \* Vol\_beam1

# Effective Mass

P\_average = (P\_SiO2 + P\_Al) / 2

M = P\_average \* (Vol\_shuttle + Vol\_beams)

m = [M] \* 10

# Resonance Frequency

wn = (ky / M)\*\*0.5 # Angular Resonance frequency

fr = wn / (2 \* 3.14159) # Resonance Frequency

gema = 1852 # Damping Constant

zeta = gema / (2 \* wn)

b = gema \* M # Damping Coefficient

Q = wn / gema # Quality Factor

# Maximum Amplitude Vs Driving Frequency

fd = []

A = []

for x in range(5):

FT = 0

FT = f\_t[x]

for wd in range(52210, 56011, 200): # Angular Driving Frequency

f = wd / (2 \* np.pi) # Driving frequency

fd.append(f)

A.append((((FT) / M) / (np.sqrt(((wn\*\*2 - wd\*\*2)\*\*2) + (wd \* gema)\*\*2))) / 10\*\*(-6))

# Convert lists to numpy arrays

X = np.array(i).reshape(-1, 1) # Input current (mA)

y = np.array(A) # Amplitude (Micrometers)

# Perform Linear Regression

regressor = LinearRegression()

regressor.fit(X, y)

predicted\_A = regressor.predict(X)

plt.figure(4)

plt.plot(fd, A, 'b', label='Actual Amplitude')

plt.plot(i, predicted\_A, 'r', label='Predicted Amplitude (Linear Regression)')

plt.grid()

plt.xlabel('Driving Frequency')

plt.ylabel('Amplitude (Micrometers)')

plt.title('Amplitude Vs Frequency')

plt.legend()

plt.savefig('figure4.png')