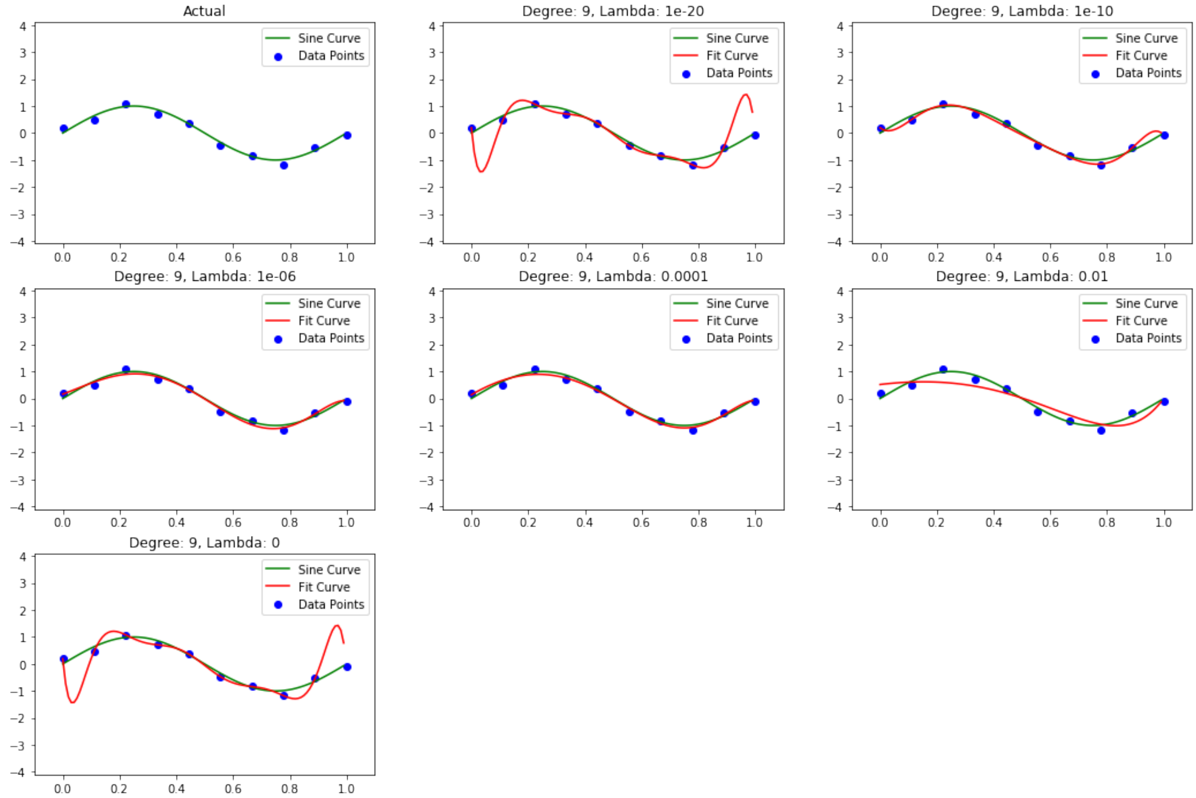
Machine Learning
Exam 1

Naman Jain Vimal Kumar
1001670153

Graphs:



Data:

```
Points:
[[0.        ]
 [0.11111111]
 [0.22222222]
 [0.33333333]
 [0.44444444]
 [0.55555556]
 [0.66666667]
 [0.77777778]
 [0.88888889]
 [1.        ]]
------

Sin Value with noise:
[[ 0.19572367]
 [ 0.4829559 ]
 [ 1.07710363]
 [ 0.70991957]
 [ 0.36567503]
 [-0.47000435]
 [-0.83735475]
 [-1.17374154]
 [-0.53841795]
 [-0.08293239]]
------
```

Values:

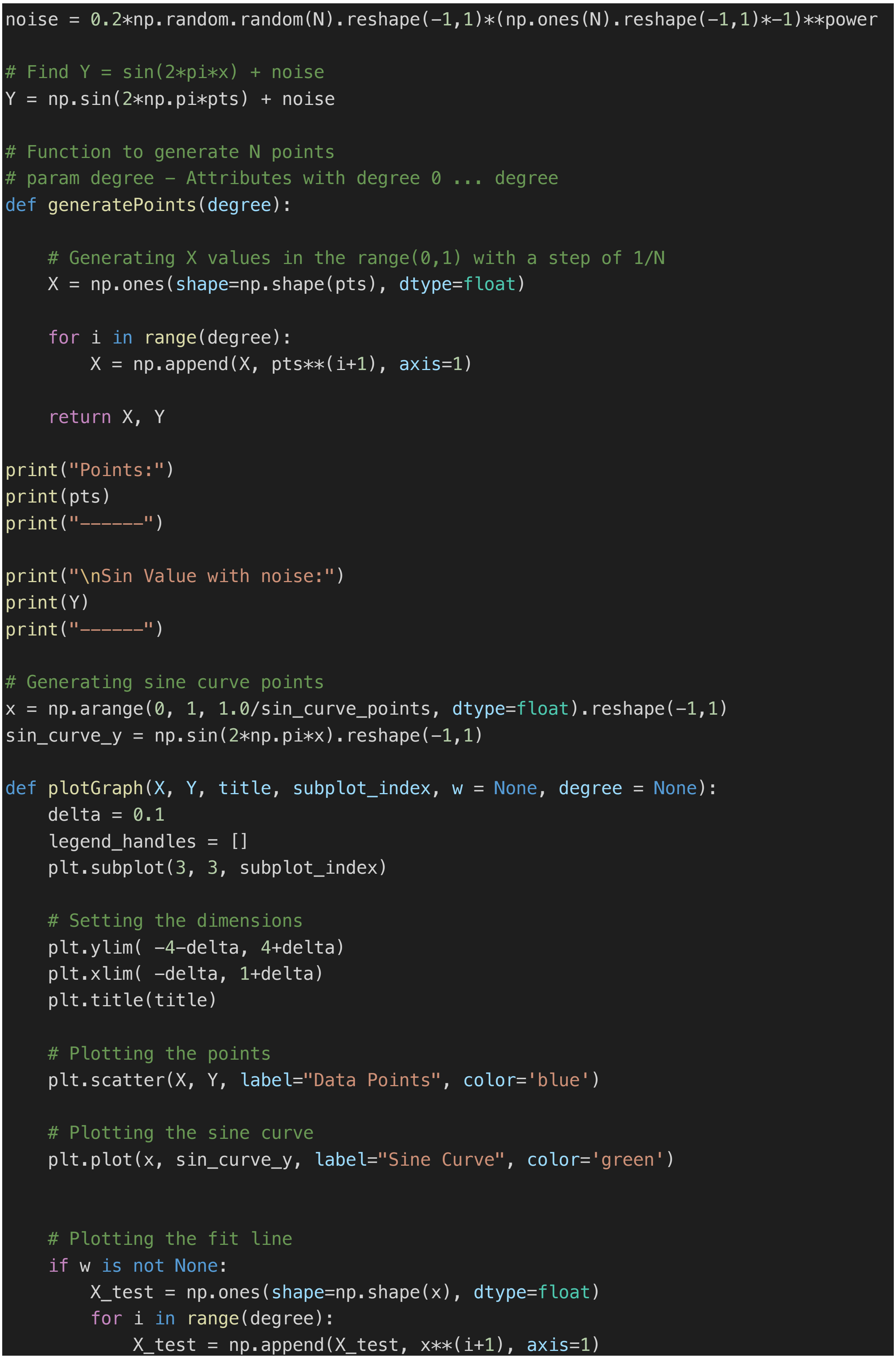
```
With Regularization
Degree: 9
```

```
Lambda: 1e-20
Weights:
[[ 1.95722613e-01]
 [-1.15103872e+02]
 [ 2.62804876e+03]
 [-2.28021694e+04]
 [ 1.04046468e+05]
 [-2.78065342e+05]
 [ 4.49758850e+05]
 [-4.32968289e+05]
 [ 2.28133518e+05]
 [-5.06162580e+04]]
------------------
Degree: 9
Lambda: 1e-10
Weights:
[[ 1.93483770e-01]
 [-8.12121392e+00]
 [ 1.74888424e+02]
 [-8.77918239e+02]
 [ 1.84084688e+03]
 [-1.63236361e+03]
 [ 1.13757305e+02]
 [ 5.56425669e+02]
 [-1.04592481e+01]
 [-1.57331849e+02]]
------------------
Degree: 9
Lambda: 1e-06
Weights:
[[  0.16200834]
 [  3.84163392]
 [  7.09809192]
 [-49.83397924]
 [ 26.78644613]
 [ 23.46818478]
 [  0.35595348]
 [ -5.49538398]
 [ -0.64693407]
 [ -5.81474268]]
------------------
Degree: 9
Lambda: 0.0001
Weights:
[[  0.13981189]
 [  6.08329395]
 [-10.21276965]
 [ -9.17334308]
 [  1.34478577]
 [  8.01132425]
 [  8.97457753]
 [  5.45711393]
 [ -1.10118723]
 [ -9.5892894 ]]
------------------
Degree: 9
Lambda: 0.01
Weights:
```

```
[[ 0.51875836]
 [ 1.24717019]
 [-3.56855273]
 [-2.178768  ]
 [-0.32004717]
 [ 0.83961864]
 [ 1.28237682]
 [ 1.20559716]
 [ 0.79665714]
 [ 0.19376729]]
------------------
Degree:  9
Lambda:  0
Weights:
[[ 1.95722613e-01]
 [-1.15103872e+02]
 [ 2.62804876e+03]
 [-2.28021694e+04]
 [ 1.04046468e+05]
 [-2.78065342e+05]
 [ 4.49758850e+05]
 [-4.32968289e+05]
 [ 2.28133518e+05]
 [-5.06162580e+04]]
------------------
```

Code:

```
# -*- coding: utf-8 -*-
"""ML_exam1.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1mKjV5JX2OwbTRREAiuClCGF6e9pguPST
"""

import numpy as np
import matplotlib.pyplot as plt
import math

# Constants
N = 10
sin_curve_points = 100
seed = 0
m = [9]

# Generating random seed
np.random.seed(seed)

# Initialization
pts = np.linspace(0,1,N).reshape(-1,1)

# Generating noise - StandX_testard Gaussian Distribution
power = np.arange(N).reshape(-1,1)
```



```
noise = 0.2*np.random.random(N).reshape(-1,1)*(np.ones(N).reshape(-1,1)*-1)**power

# Find Y = sin(2*pi*x) + noise
Y = np.sin(2*np.pi*pts) + noise

# Function to generate N points
# param degree - Attributes with degree 0 ... degree
def generatePoints(degree):

    # Generating X values in the range(0,1) with a step of 1/N
    X = np.ones(shape=np.shape(pts), dtype=float)

    for i in range(degree):
        X = np.append(X, pts**(i+1), axis=1)

    return X, Y

print("Points:")
print(pts)
print("------")

print("\nSin Value with noise:")
print(Y)
print("------")

# Generating sine curve points
x = np.arange(0, 1, 1.0/sin_curve_points, dtype=float).reshape(-1,1)
sin_curve_y = np.sin(2*np.pi*x).reshape(-1,1)

def plotGraph(X, Y, title, subplot_index, w = None, degree = None):
    delta = 0.1
    legend_handles = []
    plt.subplot(3, 3, subplot_index)

    # Setting the dimensions
    plt.ylim( -4-delta, 4+delta)
    plt.xlim( -delta, 1+delta)
    plt.title(title)

    # Plotting the points
    plt.scatter(X, Y, label="Data Points", color='blue')

    # Plotting the sine curve
    plt.plot(x, sin_curve_y, label="Sine Curve", color='green')


    # Plotting the fit line
    if w is not None:
        X_test = np.ones(shape=np.shape(x), dtype=float)
        for i in range(degree):
            X_test = np.append(X_test, x**(i+1), axis=1)
        fit_curve_y = predict(X_test, w)
        plt.plot(x, fit_curve_y, label="Fit Curve", color='red')


    plt.legend(loc='upper right')

def predict(X, w):
    return np.dot(X, w).reshape(-1,1)

"""# Without Regularization"""

print("Without Regularization")
plt.rcParams['figure.figsize'] = [18.0, 12.0]

for i in range(len(m)):
    M = m[i]
    X, Y = generatePoints(M)
    n_col = X.shape[1]
    w = np.linalg.lstsq(X, Y)[0]
    print("Degree: "+str(M))
    print("Weights:")
    print(w)
    print("------------------")
    plotGraph(pts, Y, 'Degree: '+str(M), i+2, w = w, degree = M)
plotGraph(pts, Y, 'Actual', 1)
plt.show()

"""# With Regularization"""

# Ref https://en.wikipedia.org/wiki/Regularized_least_squares
print("With Regularization")
_lambda_values = [10**-20, 10**-10, 10**-6, 10**-4, 10**-2, 0]
M = 9
plt.rcParams['figure.figsize'] = [18.0, 12.0]
for i in range(len(_lambda_values)):
    _lambda = _lambda_values[i]
    X, Y = generatePoints(M)
    n_col = X.shape[1]
    w = np.linalg.lstsq(X.T.dot(X) + _lambda * np.identity(n_col), X.T.dot(Y))[0]
    print("Degree: "+str(M))
    print("Lambda: "+str(_lambda))
    print("Weights:")
    print(w)
    print("------------------")
    plotGraph(pts, Y, 'Degree: '+str(M)+', Lambda: '+ str(_lambda), i+2, w = w, deg
ree = M)
plotGraph(pts, Y, 'Actual', 1)
plt.show()
```