Assignment 3

Omar Ahmed

UCID:30154382

Code:

```
#Question 1
import numpy as np
import matplotlib.pyplot as plt
coeff = [-2, 0, 3, 2, 1]
x = np.linspace(1, 3, 100)

# Evaluate polynomial
y = np.polyval(coeff, x)

# Plot
plt.plot(x, y, label='f(x)')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Question 1 Plot')
plt.legend()
plt.grid(True)
plt.show()
```

```
#Question 2
""" We can see that it is not exactly a polynomial, rather it is a combionation of trigonometric functions,
so we can't directly use polyval() to evaluate it.instead, To solve this we can break down the equation in
terms of power of each sin equation where sin**5 == (sin(x))**5, in this way we can rewrite the function as a polynomial of y where
f(y) = y**5 + 4y
now we can use polyval() to evaluate it
"""
import numpy as np
import matplotlib.pyplot as plt
# Define the function
def f(x):
    return np.sin(x)**5 + 4 * np.sin(x)
# Create an array of x values within the specified range
x = np.linspace(0, 2, 1000)
# Evaluate the function at these x values
y = f(x)
# Plot the curve
plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Plot of f(x) = sin^5(x) + 4sin(x) for 0 < x < 2')
plt.grid(True)
plt.show()
```

```python
#Question 3 Part A
import numpy as np
import matplotlib.pyplot as plt
# Define the coefficients of the polynomial f(x) = x^5 - 3x^4 + 4x + 1
coeff = [1, -3, 0, 0, 4, 1]
# Define the range of x values
x = np.linspace(0, 2, 1000)
# Compute the values of the polynomial, its derivative, second derivative, and integral
f_x = np.polyval(coeff, x)
df_x = np.polyval(np.polyder(coeff), x)
d2f_x = np.polyval(np.polyder(np.polyder(coeff)), x)
integral_f_x = np.polyint(coeff)
integral_values = np.polyval(integral_f_x, x)
# Create a single plot with four subplots
plt.figure(figsize=(12, 8))
# Plot f(x)
plt.subplot(2, 2, 1)
plt.plot(x, f_x, label='f(x)')
plt.xlabel('x')
plt.ylabel('f(x)')
# Plot df(x)/dx
plt.subplot(2, 2, 2)
plt.plot(x, df_x, label="df(x)/dx")
plt.xlabel('x')
plt.ylabel("df(x)/dx")
```

```python
# Plot d^2f(x)/dx^2
plt.subplot(2, 2, 3)
plt.plot(x, d2f_x, label="d^2f(x)/dx^2")
plt.xlabel('x')
plt.ylabel("d^2f(x)/dx^2")
# Plot the integral of f(x)
plt.subplot(2, 2, 4)
plt.plot(x, integral_values, label='∫[0 to x] f(t) dt')
plt.xlabel('x')
plt.ylabel('∫[0 to x] f(t) dt')
# Set titles for all subplots
plt.suptitle("Question 3A")
plt.subplots_adjust(wspace=0.3, hspace=0.3)
plt.show()
```

```python
#Question 3B
import numpy as np
# Define the coefficients of the polynomial
coefficients = [1, -3, 0, 0, 4, 1]
# Find the roots of the polynomial
roots = np.roots(coefficients)
# Initialize a list to store the real roots
real_roots = []
# Loop through the roots and identify real roots
for root in roots:
  if np.isreal(root):
    real_roots.append(np.real(root))
# Print the real roots
print("Real roots:", real_roots)
# Verify the real roots using np.polyval()
for root in real_roots:
  polynomial_value = np.polyval(coefficients, root)
print(f"Verification for root {root}: f({root}) = {polynomial_value}")
```

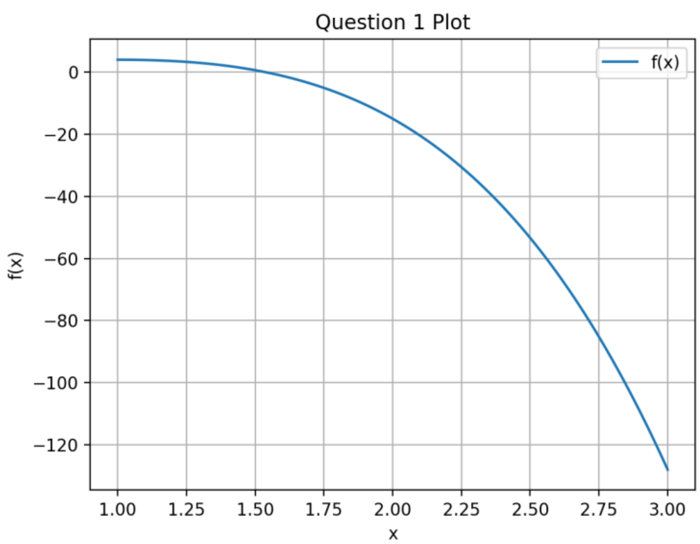
```python
#Question 4
import numpy as np
# Define the coefficients of the polynomial (x^3 + 2x + 1)^5
coefficients = [1, 0, 2, 1] # Coefficients of (x^3 + 2x + 1)
power = 5 # The exponent of the polynomial
# Initialize the result as [1] to start with (the identity element for convolution)
result = [1]
# Perform convolution power times to calculate the coefficients of the polynomial
for _ in range(power):
    result = np.convolve(result, coefficients)
# The result contains the coefficients of the expanded polynomial
print(result)
```

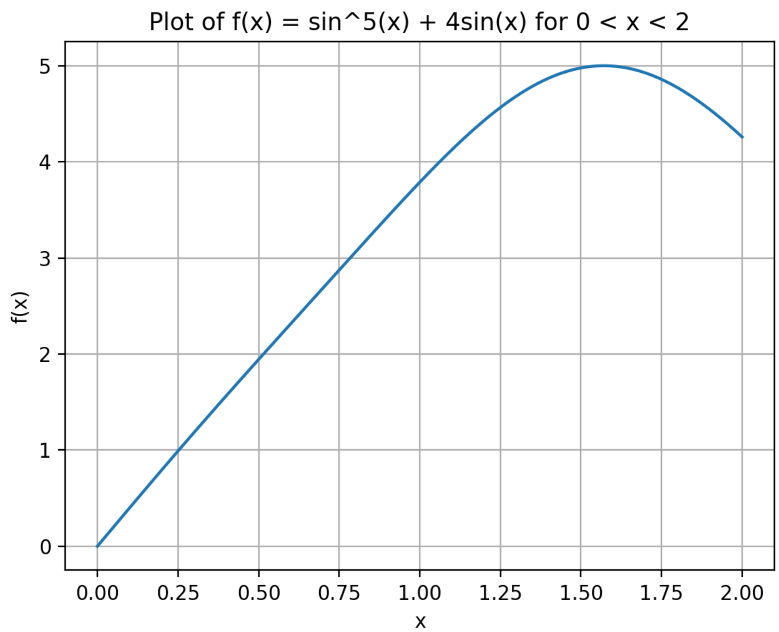
```python
#Question 5
import numpy as np
import matplotlib.pyplot as plt
x = np.array([0, 1, 2, 4, 5])
y = np.array([1, 1.5, 4, 7, 4])
# Fit a fourth-order polynomial to the data
coefficients = np.polyfit(x, y, 4)
# Create a polynomial function using the coefficients
poly = np.poly1d(coefficients)
x_vals = np.linspace(0, 5, 100)
y_vals = poly(x_vals)
# Plot the interpolated polynomial and data points
plt.plot(x_vals, y_vals, label='Poly.')
plt.scatter(x, y, color='red', marker='x', label='Pts') #Red and x as requested in the Question
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend()
plt.title('4th-Order IP')
plt.grid(True)
# Show the plot
plt.show()
```

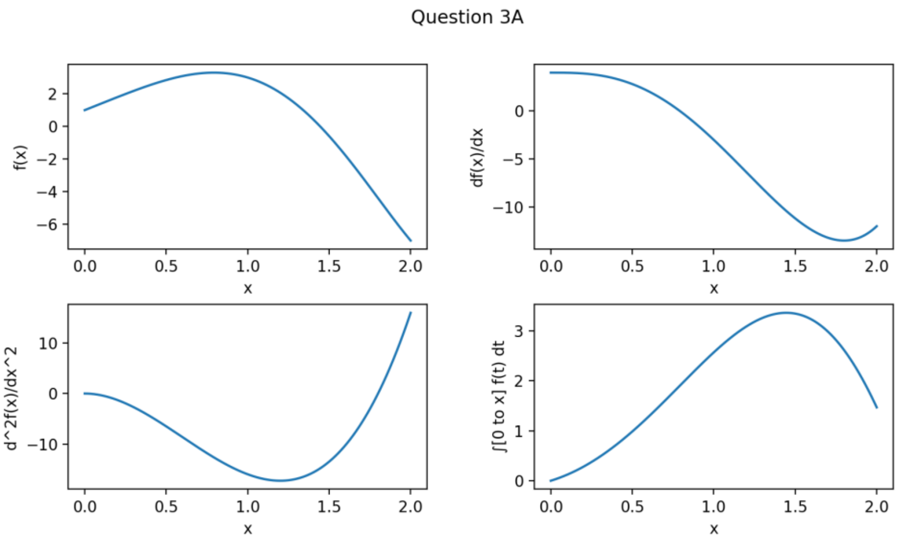
```python
#Question 6
import numpy as np
import matplotlib.pyplot as plt
#program to generate the measurement file
N = 500
x = 0.4 + 4*np.random.normal(loc=0,scale=1,size=N)
a = 0.1; b = 1; c = 2;
y = a + b*np.sin(x) + c*np.sin(x)**2
+ 0.5*np.random.normal(loc=0,scale=1,size=N)
plt.plot(x,y,'r.')
np.save('meas.npy',np.append(x.reshape(N,1),y.reshape(N,1),axis=1))
# Load the data from 'meas.npy'
data = np.load('meas.npy')
xd = data[:, 0]
yd = data[:, 1]
coeff = np.polyfit(xd, yd, 2)
print(coeff)
af = coeff[2]
bf = coeff[1]
cf = coeff[0]
print("Fitted coefficents are: a = {:.3f}, b = {:.3f}, c = {:.3f}".format(af, bf, cf))
yf = af + bf * np.sin(xd) + cf * np.sin(xd)**2
resid = yd - yf
```

```
# Standard Deviation
std_div = np.std(resid)
print("The Standard Deviation of error is: {:.3f}".format(std_div))
# Plotting the data pts
plt.figure()
plt.scatter(xd, yd, color='purple', marker='.', label='Pts')
plt.plot(xd, yf, color='r', label='Regress Curve')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Question 6A')
plt.legend()
plt.grid
plt.show()
```

Output:

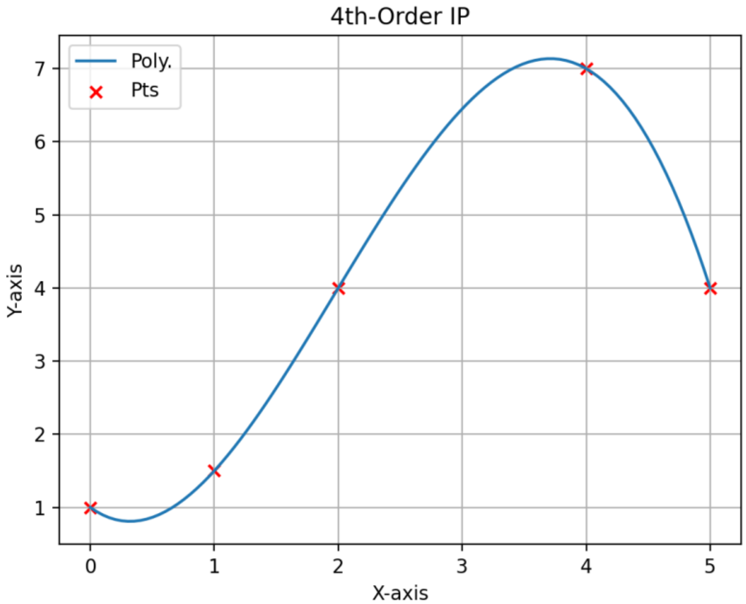


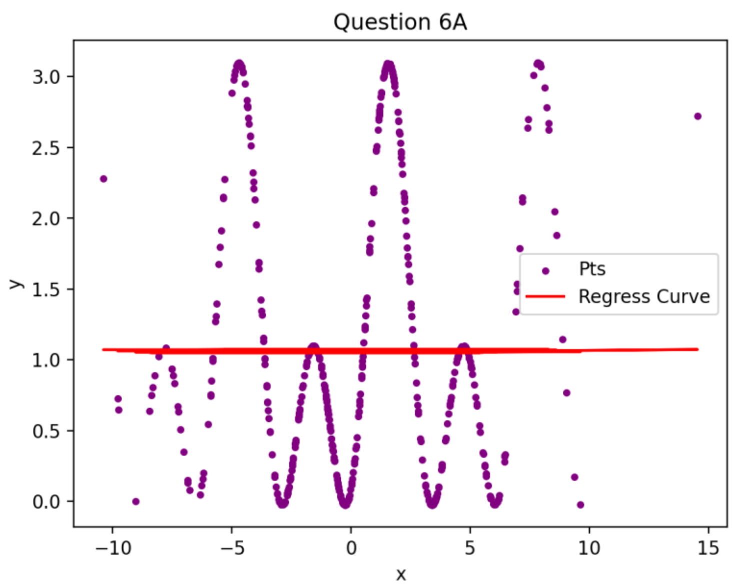




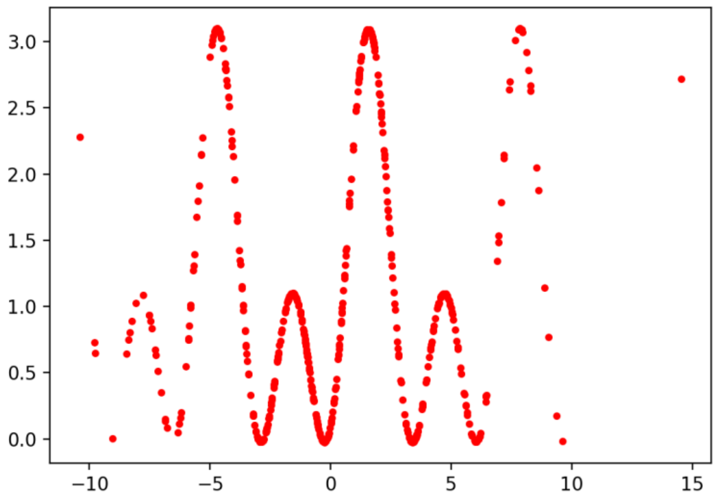

```
Real roots: [2.801937735804837, 1.4450418679126291, -0.24697960371746716]
Verification for root -0.24697960371746716: f(-0.24697960371746716) = -4.440892098500626e-16
```

```
[  1   0  10   5  40  40  90 120 140 170 152 120  85  40  10   1]
```






```
[  1   0  10   5  40  40  90 120 140 170 152 120  85  40  10   1]
[0.00309497 0.01214843 1.05960273]
Fitted coefficents are: a = 1.060, b = 0.012, c = 0.003
The Standard Deviation of error is: 1.001
```



3.0
2.5
2.0
1.5
1.0
0.5
0.0
−10
−5
0
5
10
15