Gabriel Gorospe

ece103

Lab 6 report

Due: 5/25/2023

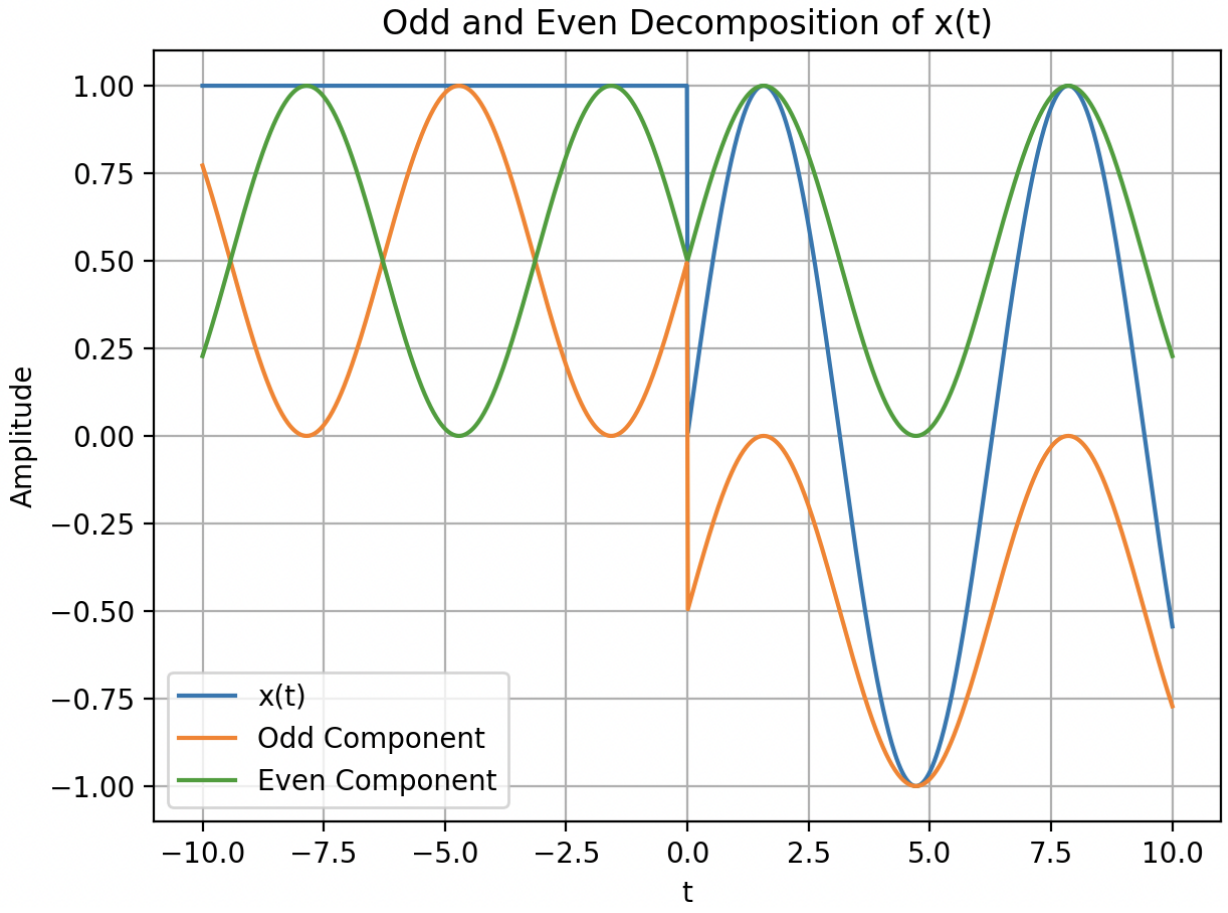
P1

Code:

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

def x(t):
    """Signal x(t)"""
    return np.sin(t) * (t > 0) + 1 * (t < 0)

def odd_decomposition(t):
    """Odd component of x(t)"""
    return 0.5 * (x(t) - x(-t))

def even_decomposition(t):
    """Even component of x(t)"""
    return 0.5 * (x(t) + x(-t))

t = np.linspace(-10, 10, 1000)

# Compute the odd and even components
odd = odd_decomposition(t)
even = even_decomposition(t)

# Plot the original signal and its odd and even components
plt.plot(t, x(t), label='x(t)')
```

```
plt.plot(t, odd, label='Odd Component')
plt.plot(t, even, label='Even Component')
plt.legend()
plt.xlabel('t')
plt.ylabel('Amplitude')
plt.title('Odd and Even Decomposition of x(t)')
plt.grid(True)
plt.show()
```

Results:



P2

Code:

```
import numpy as np
```

```
t_values = [0.0, 0.1, 0.2, 0.3, 0.4]
frequencies = [10, 15, 20]

results = np.zeros((len(frequencies), len(t_values)))

for i, f in enumerate(frequencies):
    for j, t in enumerate(t_values):
        value = 3 * np.cos(2 * np.pi * f * t + 0.1)
        results[i, j] = value

print("Results:")
print(results)
```

Results:

```
gabrielgorospe@gabemg525 assn6 % python3 p2.py
Results:
[[ 2.9850125  2.9850125  2.9850125  2.9850125  2.9850125]
 [ 2.9850125 -2.9850125  2.9850125 -2.9850125  2.9850125]
 [ 2.9850125  2.9850125  2.9850125  2.9850125  2.9850125]]
gabrielgorospe@gabemg525 assn6 %
```

P3

Code:

```
import numpy as np

# Define the angular frequencies
omegas = [35, 40, 45]

# Define the step size for t
step = 0.01

# Compute the largest value of t for each angular frequency
```

```
for omega in omegas:
    t = 0.0
    while True:
        value1 = np.exp(1.2) * np.cos(omega * t)
        value2 = t**3
        if value1 < 10 and value2 < 10:
            t += step
        else:
            break

    # Round the result to the nearest 0.01
    t = round(t - step, 2)
    print(f"For omega={omega}: Largest t = {t}")
```

Results:

```
gabrielgorospe@gabemg525 assn6 % python3 p3.py
For omega=35: Largest t = 2.15
For omega=40: Largest t = 2.15
For omega=45: Largest t = 2.15
gabrielgorospe@gabemg525 assn6 %
```

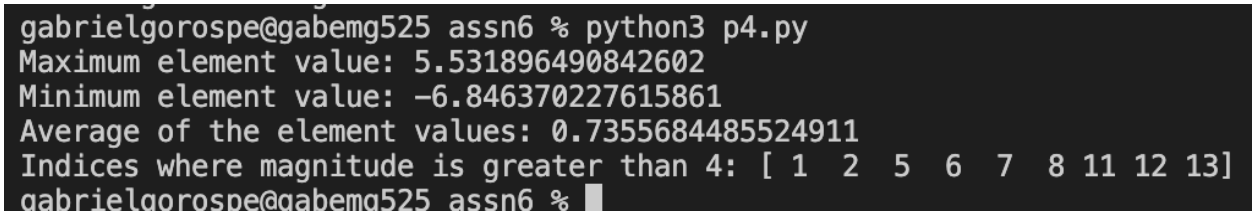
P4

Code:

```
import numpy as np

# Define the number of elements in the vector
num_elements = 15

# Create the equally spaced interval from 0 to 1
t_values = np.linspace(0, 1, num_elements)

# Compute the values of x(t)
```

```
x_values = 4 * np.cos(2 * np.pi * t_values + 0.2) + 3 * np.sin((np.pi**2) * t_values)

# Find the maximum and minimum values
max_value = np.max(x_values)
min_value = np.min(x_values)

# Compute the average of the element values
avg_value = np.mean(x_values)

# Find the indices where element magnitude is greater than 4
indices_greater_than_4 = np.where(np.abs(x_values) > 4)[0]

# Print the results
print("Maximum element value:", max_value)
print("Minimum element value:", min_value)
print("Average of the element values:", avg_value)
print("Indices where magnitude is greater than 4:", indices_greater_than_4)
```

Results:



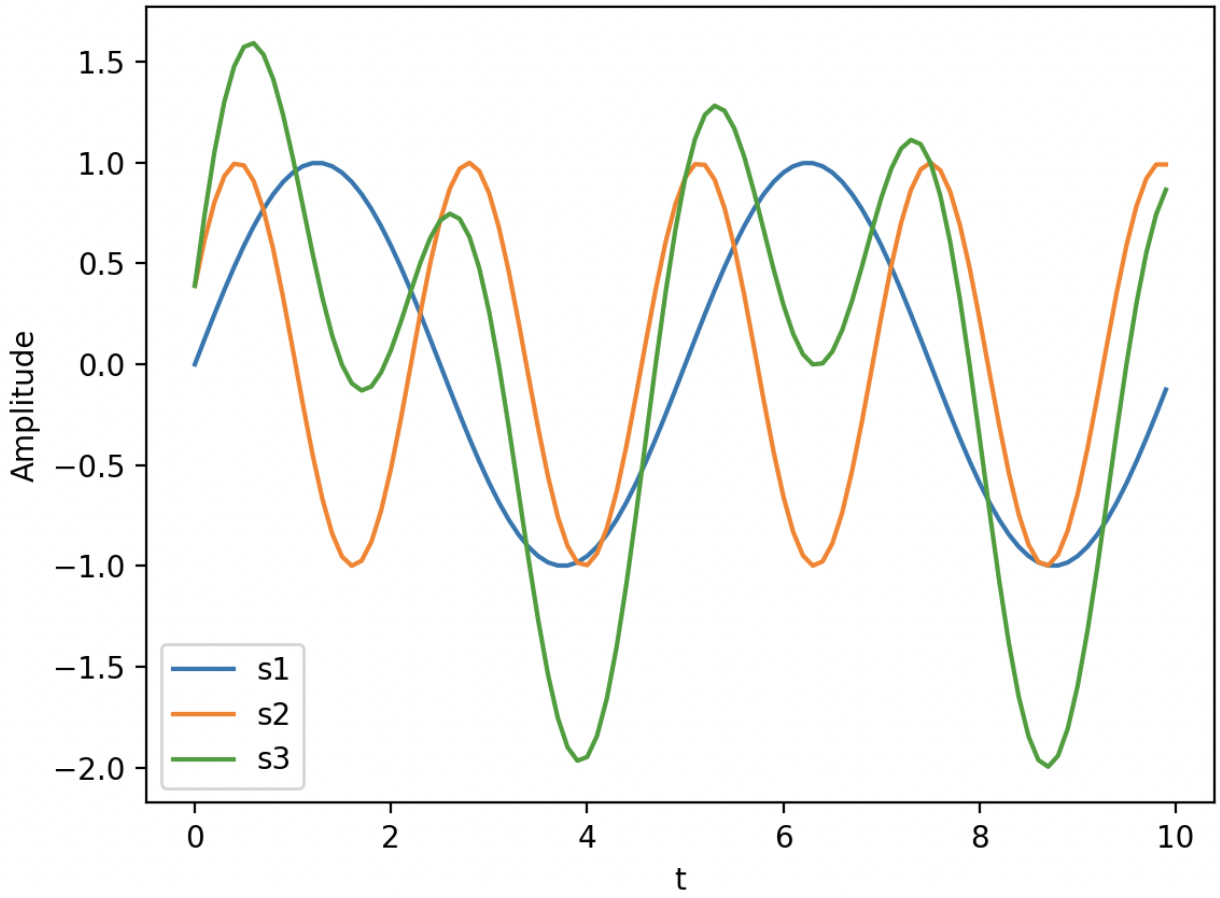
P5

Code:

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

# Define the frequencies
f1 = 0.2
```

```
f2 = 0.425

# Define the time range
t = np.arange(0, 10, 0.1)

# Compute the values of s1, s2, and s3
s1 = np.sin(2 * np.pi * f1 * t)
s2 = np.sin(2 * np.pi * f2 * t + 0.4)
s3 = s1 + s2

# Plot s1, s2, and s3 on the same graph
plt.plot(t, s1, label='s1')
plt.plot(t, s2, label='s2')
plt.plot(t, s3, label='s3')

# Add labels and legends
plt.xlabel('t')
plt.ylabel('Amplitude')
plt.legend()

# Show the plot
plt.show()
```

Results:



P6

Code:

```
x_values = linspace(-2*pi, 2*pi);
sinc1 = MySinc(x_values);
sinc2 = sinc(x_values);
plot(x_values, sinc1,x_values,sinc2);
legend('MySinc', 'matlab sinc');
function y = MySinc(x)

    for i = 1:length(x)
        if x(i) == 0
            y(i) = 1;
        else
            y(i) = sin(x(i))/x(i);
```

```
        end

    end
end
```

Results: