TTT4120 Digital Signal Processing

Problem Set 1

Peter Pham

2023-08-31

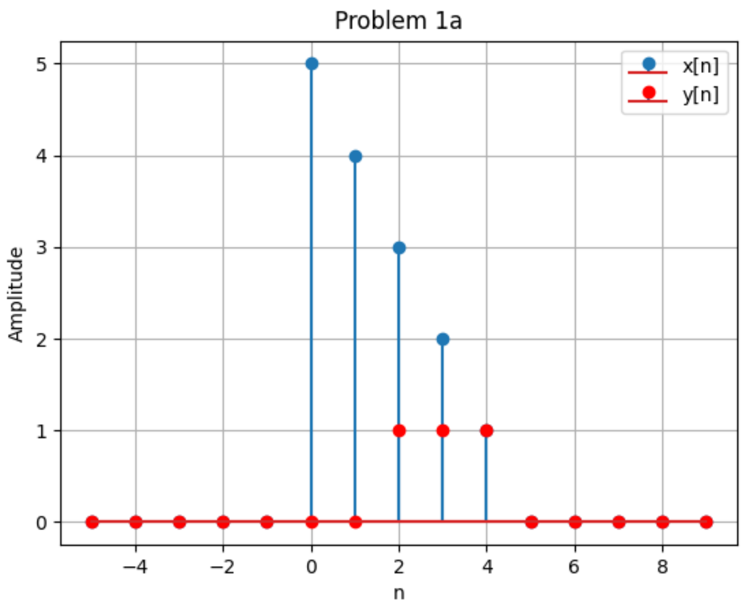
Contents

Problem 1 (2 points) **3**

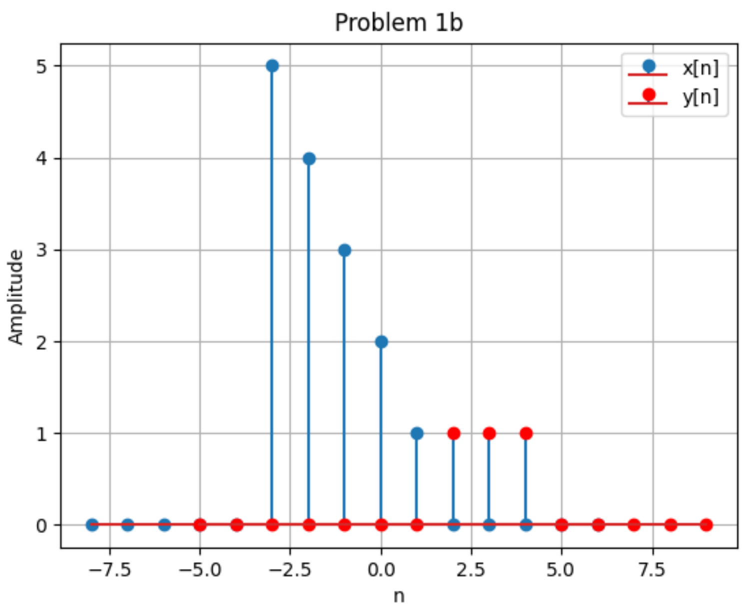
Problem 2 (2 points) **10**

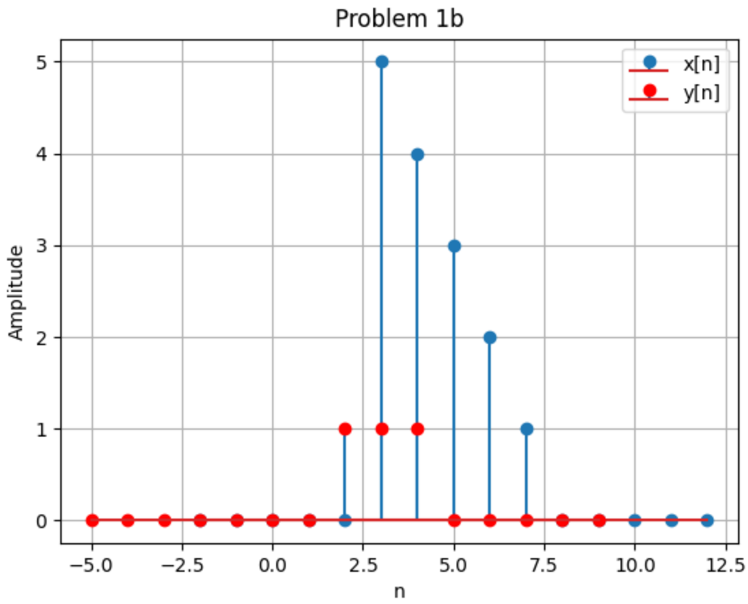
Problem 1 (2 points)

(a) Sketch $x[n]$ and $y[n]$

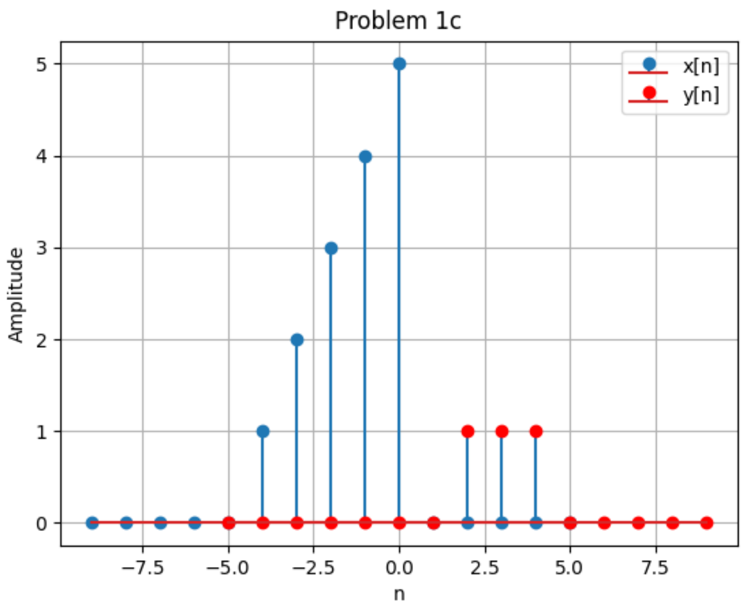


(b) Sketch $x[n-k]$ for $k = 3$ and $k = -3$.

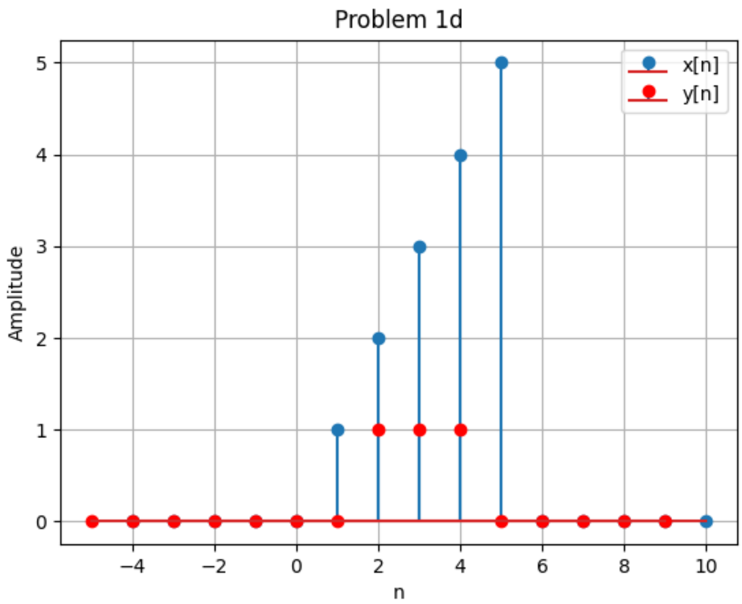



Problem 1b
Amplitude
n
x[n]
y[n]
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
0
1
2
3
4
5

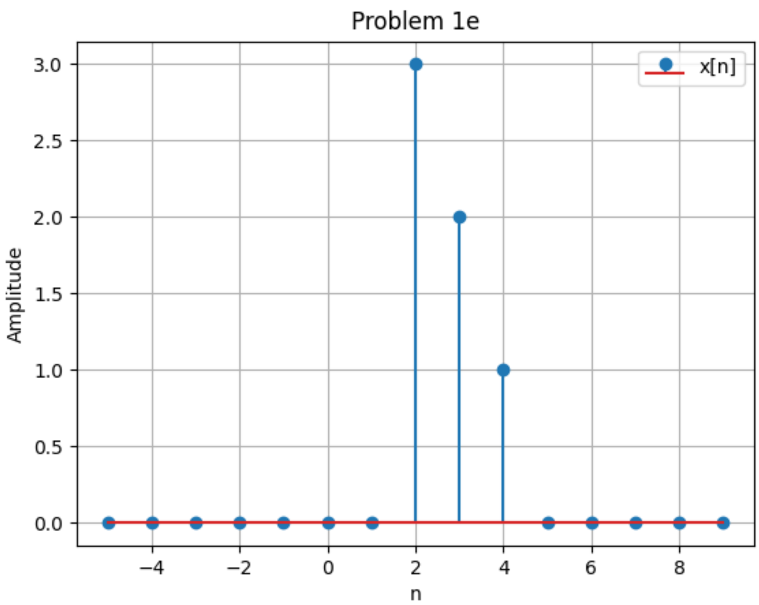
(c) Sketch $x[-n]$.


Problem 1c
Amplitude
n
x[n]
y[n]
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
0
1
2
3
4
5

(d) Sketch $x[5-n]$.



(e) Sketch $x[n] \cdot y[n]$.



(f) Express the signal $x[n]$ by using the unit sample sequence $\delta[n]$.

$$x[n] = 5\delta[n] + 4\delta[n-1] + 3\delta[n-2] + 2\delta[n-3] + 1\delta[n-4]$$

(g) Express the signal $y[n]$ by using the unit step signal $u[n]$.

$$y[n] = u[n-2] - u[n-5]$$

(h) Compute the energy of the signal $x[n]$.

The energy og the signal $x[n] = 55$ Solved in python

Python Code

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

# Problem 1:
def x_n(n):
    if 0 <= n <= 4:
        return 5 - n
    else:
        return 0

def y_n(n):
    if 2 <= n <= 4:
        return 1
    else:
        return 0

n_values = np.arange(-5, 10)  # A range of n values
x_values = [x_n(n) for n in n_values]
y_values = [y_n(n) for n in n_values]
plt.figure()


# (a) Sketch x[n] and y[n]
```

```
def problem_1a():
    plt.stem(n_values, x_values, label="x[n]")
    plt.stem(n_values, y_values, label="y[n]", markerfmt='ro')
    plt.xlabel('n')
    plt.ylabel('Amplitude')
    plt.legend()
    plt.title('Problem␣1a')
    plt.grid(True)
    plt.show()
problem_1a()

# (b) Sketch x[n-k] for k=3 and k=-3
def problem_1b1():
    plt.stem(n_values-3, x_values, label="x[n]")
    plt.stem(n_values, y_values, label="y[n]", markerfmt='ro')
    plt.xlabel('n')
    plt.ylabel('Amplitude')
    plt.legend()
    plt.title('Problem␣1b')
    plt.grid(True)
    plt.show()
problem_1b1()
def problem_1b2():
    plt.stem(n_values+3, x_values, label="x[n]")
    plt.stem(n_values, y_values, label="y[n]", markerfmt='ro')
    plt.xlabel('n')
    plt.ylabel('Amplitude')
    plt.legend()
    plt.title('Problem␣1b')
    plt.grid(True)
    plt.show()
problem_1b2()

# (c) Sketch x[-n]
def problem_1c():
    plt.stem(-1*n_values, x_values, label="x[n]")
    plt.stem(n_values, y_values, label="y[n]", markerfmt='ro')
    plt.xlabel('n')
    plt.ylabel('Amplitude')
    plt.legend()
    plt.title('Problem␣1c')
```

```
    plt.grid(True)
    plt.show()
problem_1c()

# (d) Sketch x[5-n]
def problem_1d():
    plt.stem(-1*n_values+5, x_values, label="x[n]")
    plt.stem(n_values, y_values, label="y[n]", markerfmt='ro')
    plt.xlabel('n')
    plt.ylabel('Amplitude')
    plt.legend()
    plt.title('Problem 1d')
    plt.grid(True)
    plt.show()
problem_1d()

# (e) Sketch x[n] * y[n]
def problem_1e():
    def x_n_y_n(n):
        return x_n(n)*y_n(n)
    xy_values = [x_n_y_n(n) for n in n_values]

    plt.stem(n_values, xy_values, label="x[n]")
    plt.xlabel('n')
    plt.ylabel('Amplitude')
    plt.legend()
    plt.title('Problem 1e')
    plt.grid(True)
    plt.show()
problem_1e()



# (h) Compute the energy of the signal x[n]

def problem_1h():
    e=0
    for n in n_values:
        e+=x_n(n)*x_n(n)

    print("The energy og the signal x[n] = ",e)
```

```
problem_1h()

# Energy = sum of (x[n])^2 for all n
```

Problem 2 (2 points)

(a) Which physical frequencies F_1 can f_1 correspond to if $F_s = 6000Hz$?

As the sampling frequency need to be twice as high as the frequency after sampling we get from the equation:

$$-\frac{F_s}{2} \leq f \leq \frac{F_s}{2}$$

this gives us:

$$-3000Hz \leq f \leq 3000Hz$$