CENG 384 - Signals and Systems for Computer Engineers
Spring 2023
Homework 1

Karabacak, Kerem
e2644417@ceng.metu.edu.tr

I worked individually
exxxxxxx@ceng.metu.edu.tr

April 2, 2023

---

1. (a) We can use basic algebraic operations:

$$\bar{z} = x - yj$$
$$2(x + yj) + 5 = j - (x - yj)$$
$$2x + 2yj + 5 = j - x + yj$$
$$3x + yj = -5 + j$$
$$x = \frac{-5}{3}, y = 1$$
$$|z| = \sqrt{(-\frac{5}{3})^2 + 1^2} = \frac{\sqrt{34}}{3}$$
$$|z|^2 = \frac{34}{9}$$

(b) $z^5 = r^5 e^{j5\theta} = 32j$
We can clearly see r is 2, and we can use Euler's formula to compute θ:
$e^{j\theta} = cos\theta + jsin\theta$
$e^{j5\theta} = cos5\theta + jsin5\theta = j$
We can get two equations from this:
$cos5\theta = 0$ and $sin5\theta = 1$
We can find 5θ as $\frac{\pi}{2}$, and $\theta = \frac{\pi}{10}$(I did not consider periodicity, and worked on [0,2π]).

Therefore z is $2e^{j\frac{\pi}{10}}$.

(c) At first we should multiply both numerator and denumerator with $\overline{z} = (-j - 1)$.

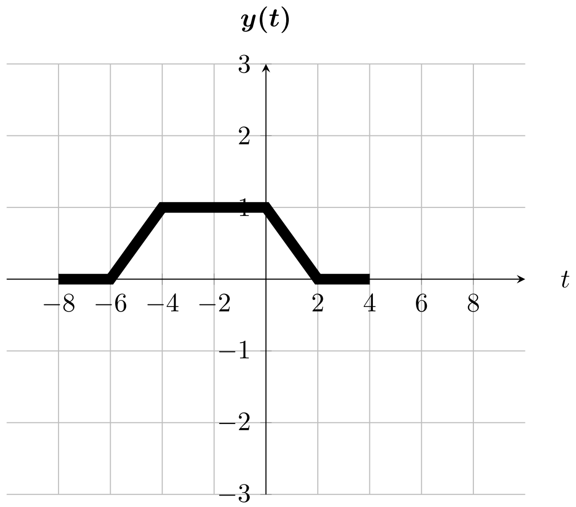
$\frac{-(j+1)^2(\frac{1}{2}+\frac{\sqrt{3}}{2}j)}{(j-1)(-j-1)} = \frac{-(j+1)^2(\frac{1}{2}+\frac{\sqrt{3}}{2}j)}{2} = \frac{-2j(\frac{1}{2}+\frac{\sqrt{3}}{2}j)}{2} = -j(\frac{1}{2} + \frac{\sqrt{3}}{2}j) = -\frac{j}{2} + \frac{\sqrt{3}}{2}$.

$r = \sqrt{(-\frac{1}{2})^2 + (\frac{\sqrt{3}}{2})^2} = 1$

$\theta = tan^{-1}(\frac{\frac{-1}{2}}{\frac{\sqrt{3}}{2}}) = tan^{-1}(-\frac{1}{\sqrt{3}}) = -\frac{\pi}{6} = \frac{11\pi}{6}$

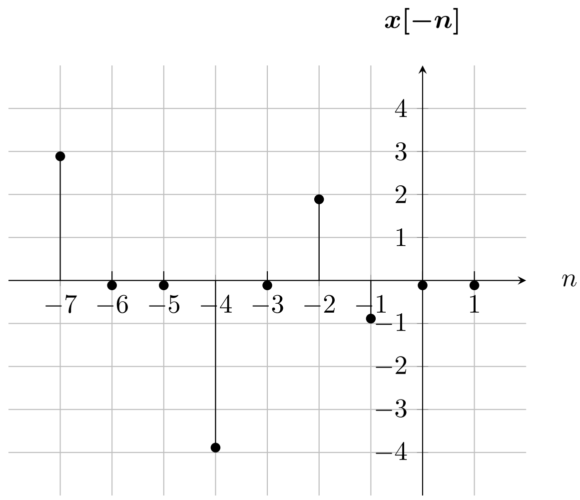
z = $e^{j\frac{11\pi}{6}}$

(d) We can convert j to polar form: $r = 1, \theta = \frac{\pi}{2}$.
$j = e^{j\frac{\pi}{2}}$. $z = e^{j\frac{\pi}{2}} e^{-j\frac{\pi}{2}} = e^0 = 1$.
1 is already in polar form where r = 1 and $\theta = 0$.

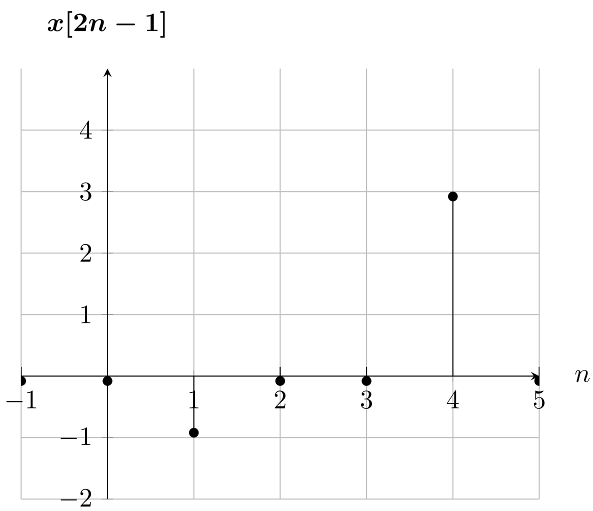


2.

3. (a) Graph for x[-n]:

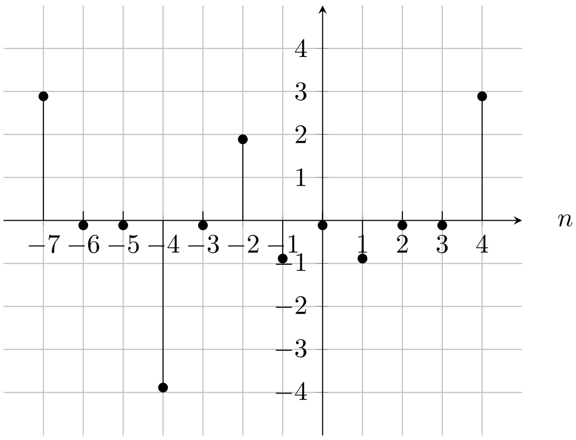


Graph for x[2n-1]:



Finally:

$\boldsymbol{x[-n] + x[2n-1]}$



(b) y[n] = x[-n] + x[2n-1] in terms of δ function:

$$y[n] = 3\delta[n+7] - 4\delta[n+4] + 2\delta[n+2] - \delta[n+1] - \delta[n-1] + 3\delta[n-4]$$

4. To compute fundemental period of a CT signal we can use following: $T_0 = \frac{2\pi}{w}$ where w is angular frequency. For DT case we can use this equality: $\omega N = 2\pi m$ where m and N integers, and m should be smallest possible integer which makes N an integer. This part will be used for solving the problems.

   (a) We have CT signal with w = 3:

   $$T_0 = \frac{2\pi}{w}$$
   $$T_0 = \frac{2\pi}{3}$$

   Since T_0 is real, this signal is periodic.

   (b) Two elements can be evaluated seperately. First, let's conider cos.

   $$\omega N = 2\pi m$$
   $$\frac{13\pi}{10}N = 2\pi m$$
   $$N = \frac{20m}{13}$$

   Smallest m is 13 and N_0 is 20. Now let's consider sin:

   $$\omega N = 2\pi m$$
   $$\frac{7\pi}{10}N = 2\pi m$$
   $$N = \frac{20}{7}m$$

   Smallest m is 7 and N_0 is 20. If we take lcm of both functions:

   $$lcm(20, 20) = 20$$

   Therefore this signal is periodic with fundamental period 20.

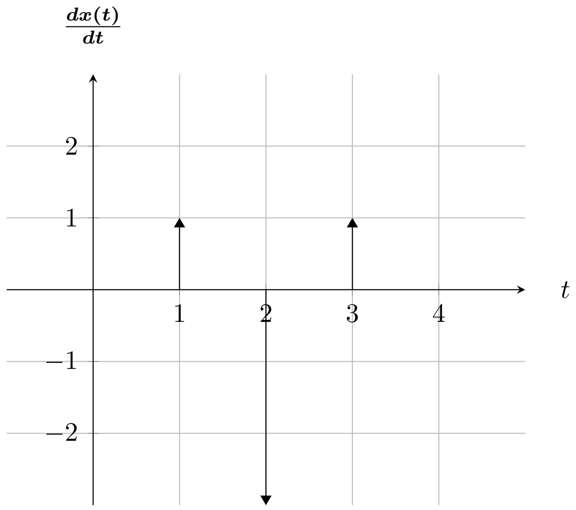
   (c) We have DT signal:

   $$\omega N = 2\pi m$$
   $$7N = 2\pi m$$
   $$N = \frac{2\pi}{7}m$$

   There is no m that makes N an integer. Therefore this signal is aperiodic.

5. (a) Expression of x(t) in terms of unit step function:

   $$x(t) = u(t-1) - 3u(t-3) + u(t-4)$$



(b)

6. (a) Memory: Since this system's output depends on past and future input, it has memory.
Stability: If we consider input as a constant, output still have t and not constant. Therefore this system is not stable, i.e tC is not constant.

Causality: For instance for t = 0, we need x(3) as input, so we need future values. System is not causal.
Linearity:
$y_1(t) = tx_1(2t+3)$, $y_2(t) = tx_2(2t+3)$
$x_3(t) = \alpha x_1(2t+3) + \beta x_2(2t+3)$
$y_3(t) = tx_3(2t+3)$
$y_3(t) = t\alpha x_1(2t+3) + t\beta x_2(2t+3)$
$y_3(t) = \alpha y_1(t) + \beta y_2(t)$
Linearity conditions are hold. Linear.
Invertibility: We can compute inverse of signal:

$$x(t) = y^{-1}(n)$$

$$x(t) = \frac{1}{t}y(\frac{n-3}{2})$$

Invertible.
Time-invariance: When we shift the input, output will not shift because of t. Not time invariant.

$$y(t-t_0) = tx(2(t-t_0+3)) \neq y(t)$$

(b) Memory: System's output needs past inputs, it has memory.
Stability: If we consider input as a constant, the result will be a constant as well. Therefore this system is stable.
Causality: System's output depends on only past inputs. Therefore this system is causal.
Linearity: There is no multiplication or division in the system. We are only summing up past values. If we multiply x_1 with α and x_2 with β, and sum them up, since there is no other element in the system, it will be equal to $\alpha y_1 + \beta y_2$. Therefore system is linear.
Invertibility:

$$x[n] = y^{-1}[n]$$

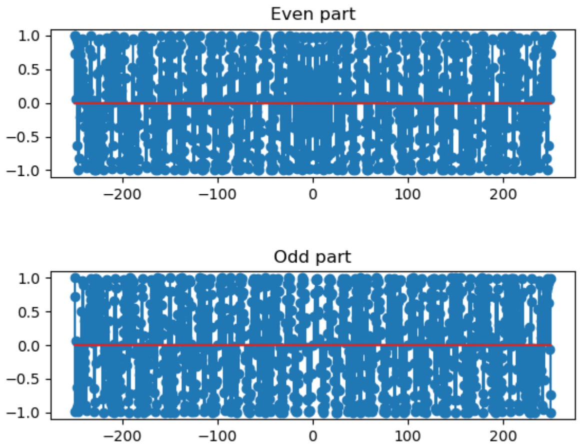
$$x[n] = \sum y[n+k]$$

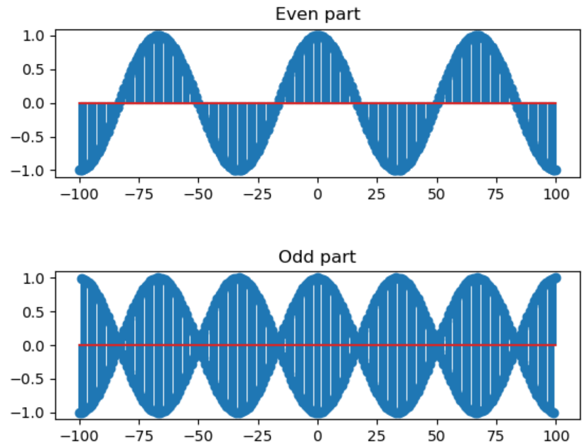
Since inverse exists, system is invertible.
Time-invariance: IF we shift the input, the output will be shifted as well, because there is no other element than input signal. Therefore system is time invariant.

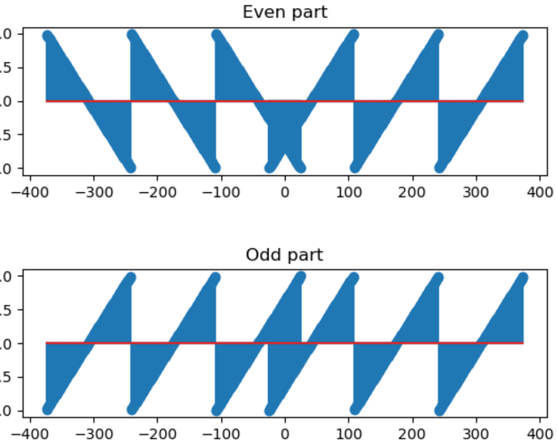
7. (a) import matplotlib.pyplot as plt

f = open('chirp_part_a.csv', 'r') # I couldn't find a way for parametric filename
floatInputs = []
inputs = f.readline()
inputsArray = inputs.split(",")
for i in range(0, len(inputsArray)):
floatInputs.append(float(inputsArray[i]))

```
length = len(floatInputs) - 1

xAxis = []
for i in range(0,length-1):
xAxis.append(floatInputs[0] + i)

firstShifted = []
for i in range(0,length-1):
firstShifted.append(xAxis[i]*(-1))

yAxis = floatInputs[1:length]

minus = []
for i in range(0,length-1):
minus.append(yAxis[i]*(-1))

even = plt.subplot2grid((5,2),(0,0), colspan=2, rowspan=2)
odd = plt.subplot2grid((5,2), (3,0), colspan=2, rowspan=2)

even.stem(xAxis, yAxis)
even.stem(firstShifted, yAxis)
even.set_title(”Even part”)

odd.stem(xAxis, yAxis)
odd.stem(firstShifted, minus)
odd.set_title(”Odd part”)

plt.show()

f.close()
```







(b)
```
import matplotlib.pyplot as plt
import numpy as np
```

```
f = open(’chirp_part_b.csv’, ’r’)
floatInputs = []
inputs = f.readline()
inputsArray = inputs.split(”,”)
for i in range(0, len(inputsArray)):
floatInputs.append(float(inputsArray[i]))

length = len(floatInputs) - 3

xAxis = []
for i in range(0,length-3):
xAxis.append(floatInputs[0] + i)

firstShifted = []
for i in range(0,length-3):
firstShifted.append(xAxis[i]/floatInputs[1])

newB = floatInputs[2]*(-1)/floatInputs[1]

secondShifted = []
for i in range(0,length-3):
secondShifted.append(firstShifted[i] + newB)

yAxis = floatInputs[3:length]
plt.stem(secondShifted, yAxis)
plt.show()
f.close()
```