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

Çavuşoğlu, Arda
e2448249@ceng.metu.edu.tr

Çolak, Eren
e2587921@ceng.metu.edu.tr

April 2, 2023

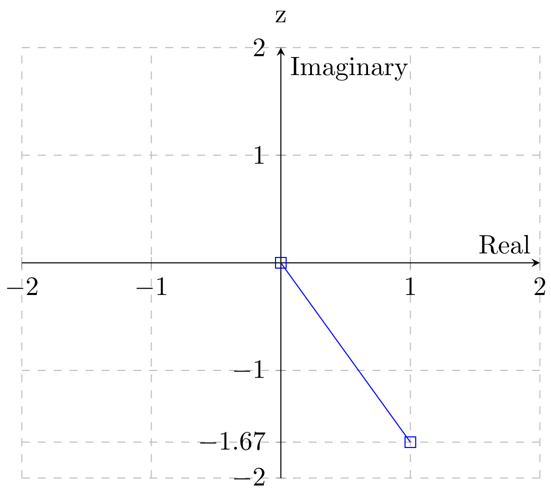
---

1. (a) $z = x + yj \qquad 2z + 5 = j - \bar{z}$

   $2x + 2yj + 5 = j - (x - yj)$

   $3x + 5 + yj = j \qquad y = 1, x = \frac{-5}{3}, z = \frac{-5}{3} + j$

   $|z|^2 = \frac{34}{9}$



   (b) $z = r.e^{j\Theta} \qquad z^5 = 32j$

   $j = e^{\frac{\pi}{2}j} \qquad r^5 e^{5\Theta j} = 32e^{\frac{\pi}{2}j} \qquad r = 2, \Theta = \frac{\pi}{10}, z = 2e^{\frac{\pi}{10}j}$

   (c) $(1 + j) = \sqrt{2}.e^{\frac{\pi}{4}j}$

   $(\frac{1}{2} + \frac{\sqrt{3}}{2}j) = e^{\frac{\pi}{3}j}$

   $(-1 + j) = \sqrt{2}.e^{\frac{3\pi}{4}j}$

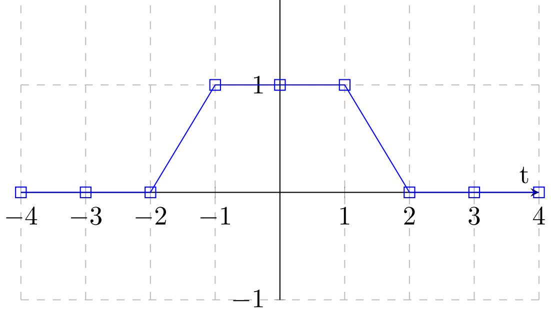
   $\frac{\sqrt{2}.e^{\frac{\pi}{4}j}e^{\frac{\pi}{3}j}}{\sqrt{2}.e^{\frac{3\pi}{4}}} = e^{\frac{-2\pi}{3}j}$

   $r = 1, \Theta = \frac{-2\pi}{3}$

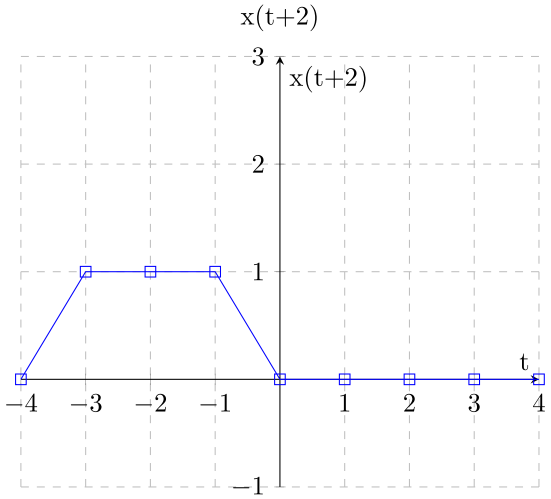
   (d) $z = j.e^{-j.\frac{\pi}{2}} \qquad j = e^{\frac{\pi}{2}.j}$

   $z = e^{\frac{\pi}{2}.j}.e^{\frac{-\pi}{2}.j} = 1 = e^{2\pi j}$

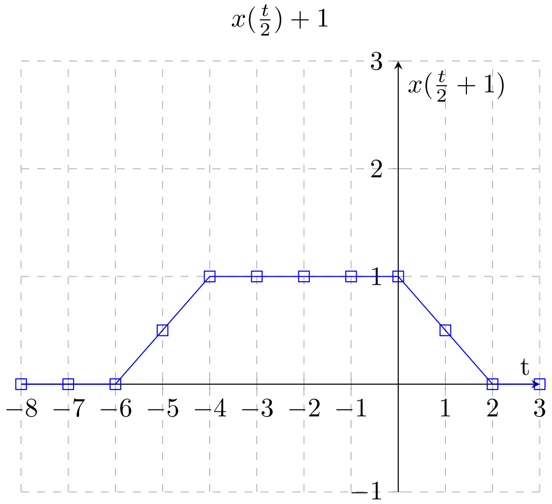
2. $x(t) \longrightarrow$



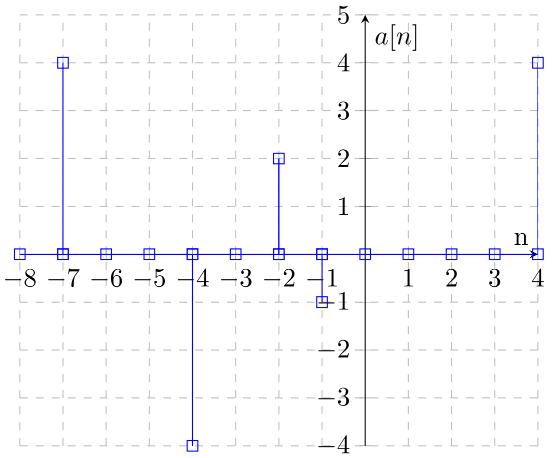
$x(t+2) \longrightarrow$



$x(\frac{t+2}{2}) = x(\frac{t}{2}+1) \longrightarrow$



$a[n] = x[-n] + x[2n-1]$

3. (a) 

(b) $a[n] = 4\delta[n+7] - 4\delta[n+4] + 2\delta[n+2] - \delta[n+1] + 4\delta[n-4]$

4. (a) $x(t) = 5sin(3t - \frac{\pi}{4}) \longrightarrow$ periodic, fundamental period $= \frac{2\pi}{3}$

(b) $x[n] = cos[\frac{13\pi}{10}n] + sin[\frac{7\pi}{10}n]$

$\frac{\frac{13\pi}{10}}{2\pi} = \frac{13}{20}, N_1 = 20$

$\frac{\frac{7\pi}{10}}{2\pi} = \frac{7}{20}, N_2 = 20$

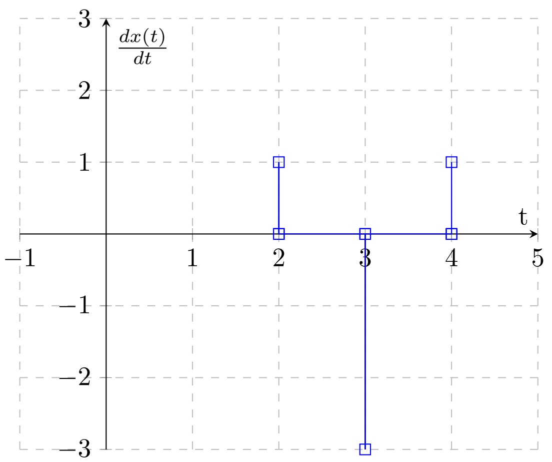
$N = LCM(N_1, N_2) = 20 \longrightarrow$ periodic, fundamental period $= 20$

(c) $x[n] = \frac{1}{2}cos[7n-5] \longrightarrow$ aperiodic, as $\frac{7}{2\pi}$ is irrational

5. (a) $x(t) = -3u(t-3) + u(t-2) + u(t-4)$

$\frac{dx(t)}{dt} = -3\delta(t-3) + \delta(t-2 + \delta(t-4)$

$\frac{dx(t)}{dt}$

(b) 

6. (a) $y(t) = tx(2t+3)$

-Not causal, as the output depends on inputs after time t. $(t+3)$

-With memory, as the output depends on inputs other than at time t. $(t+3)$

-Invertible, as the inverse of the system is $h^{-1}(y(t)) = \frac{y(\frac{t-3}{2})}{t} = \frac{tx(2(\frac{t-3}{2})+3)}{t} = x(t)$.

-Not stable, due to the coefficient t the system will diverge even if we provide a bounded input.

-Time variant, since shifting t by t_0 does not shift the system by t_0

$x_1(2t+3) = x(2(t-t_0)+3) \longrightarrow y_1(t) = tx_1(2t+3) = tx(2(t-t_0)+3) \neq y(t-t_0) = (t-t_0)x(2(t-t_0)+3)$

-Linear, $x_1(2t+3) \longrightarrow y_1(t) = tx_1(2t+3)$

$x_2(2t+3) \longrightarrow y_2(t) = tx_2(2t+3)$

$x_3(2t+3) = \alpha x_1(2t+3) + \beta x_2(2t+3)$ and

$y_3 = t(\alpha x_1(2t+3) + \beta x_2(2t+3)) = \alpha y_1 + \beta y_2$

(b) $y[n] = \Sigma_{k=1}^{\infty} x[n-k]$

-Causal, as k only takes positive values, the system will only depend on past values

-With memory, as the system depends on past values

-Not invertible

-Unstable, as the system will diverge to infinity if we put constant values for the input

-Time invariant, since shifting n by n_0 will shift the system by n_0

$$x_1[n-k] = x[n-n_0-k] \longrightarrow y_1[n] = \Sigma_{k=1}^{\infty} x[n-n_0-k] = y[n-n_0] = \Sigma_{k=1}^{\infty} x[n-n_0-k]$$

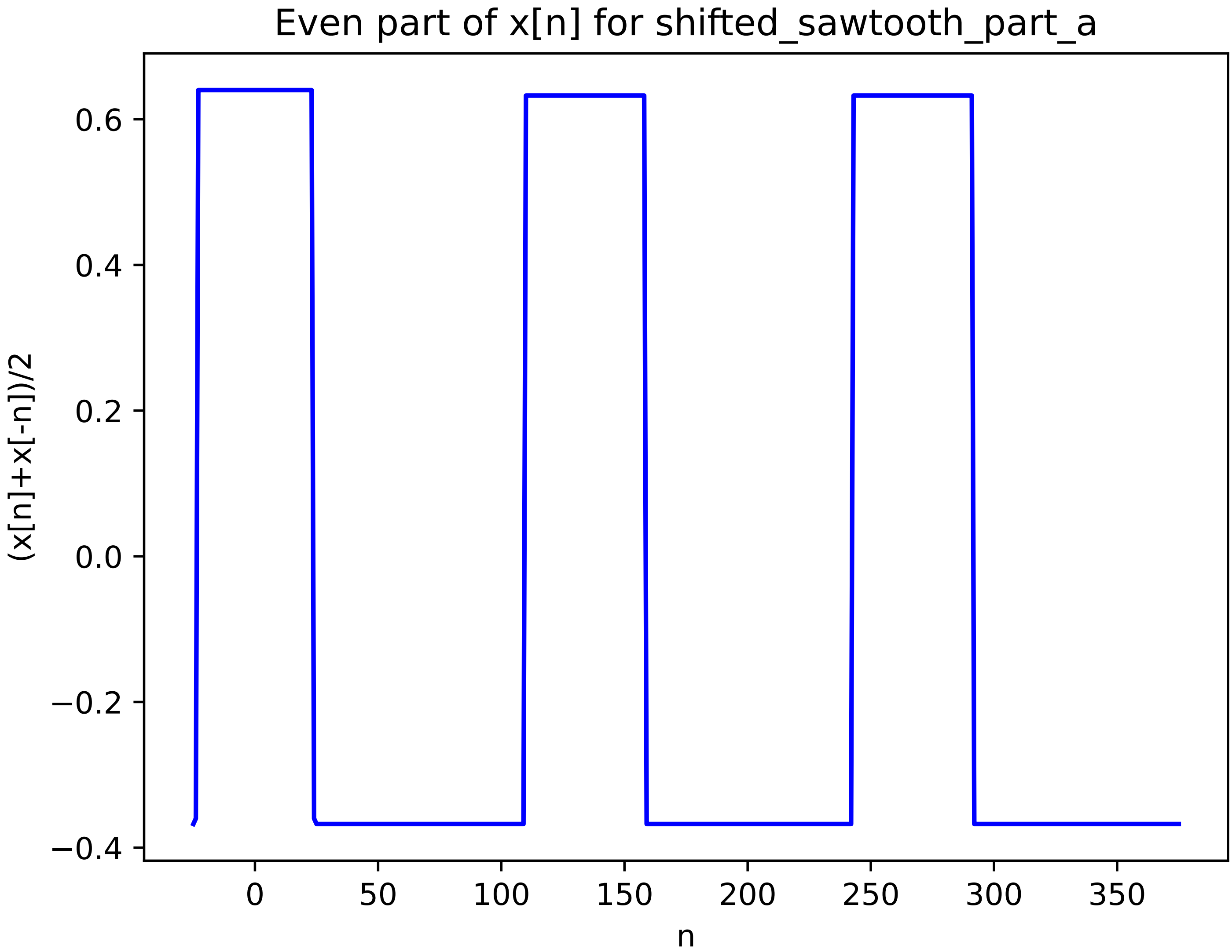
-Linear, $x_1[n-k] \longrightarrow y_1[n] = \Sigma_{k=1}^{\infty} x_1[n-k]$

$x_2[n-k] \longrightarrow y_2[n] = \Sigma_{k=1}^{\infty} x_2[n-k]$

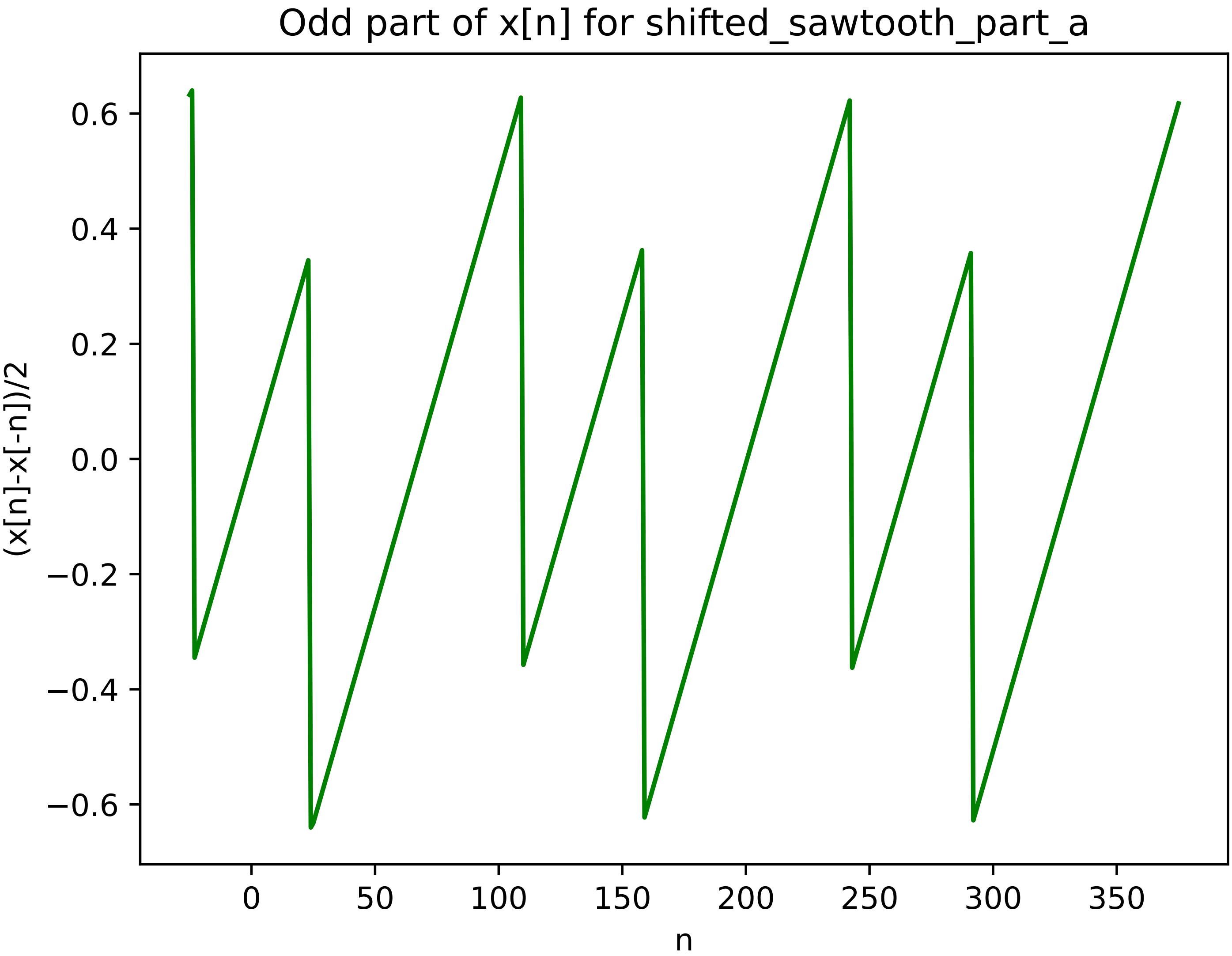
$x_3[n-k] = \alpha x_1[n-k] + \beta x_2[n-k]$ and

$y_3 = \Sigma_{k=1}^{\infty} x_3[n-k] = \Sigma_{k=1}^{\infty} \alpha x_1[n-k] + \beta x_2[n-k] = \alpha y_1[n] + \beta y_2[n]$

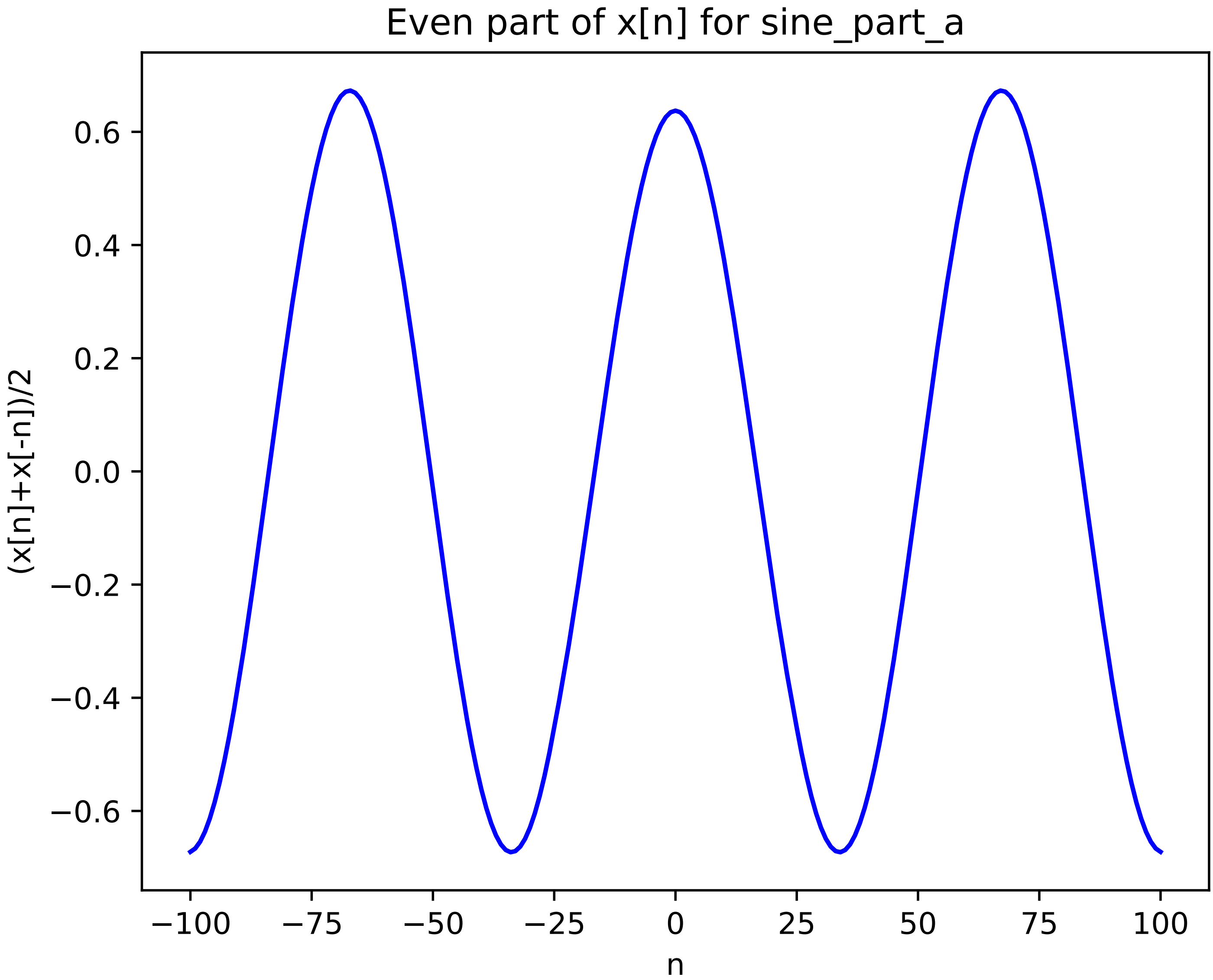
7. (a)



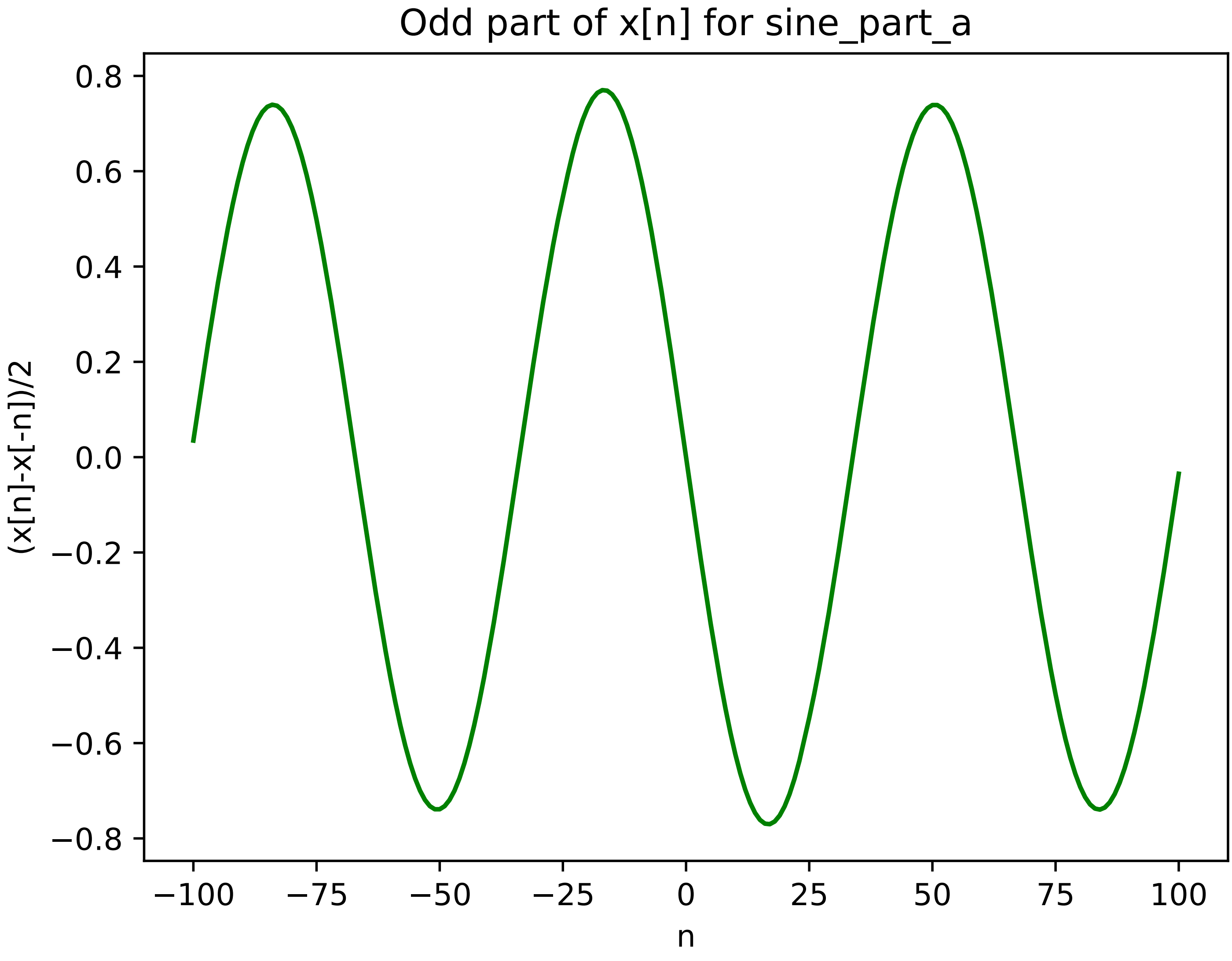
Even part of x[n] for shifted_sawtooth_part_a
0.6
0.4
0.2
0.0
−0.2
−0.4
(x[n]+x[-n])/2
0
50
100
150
200
250
300
350
n



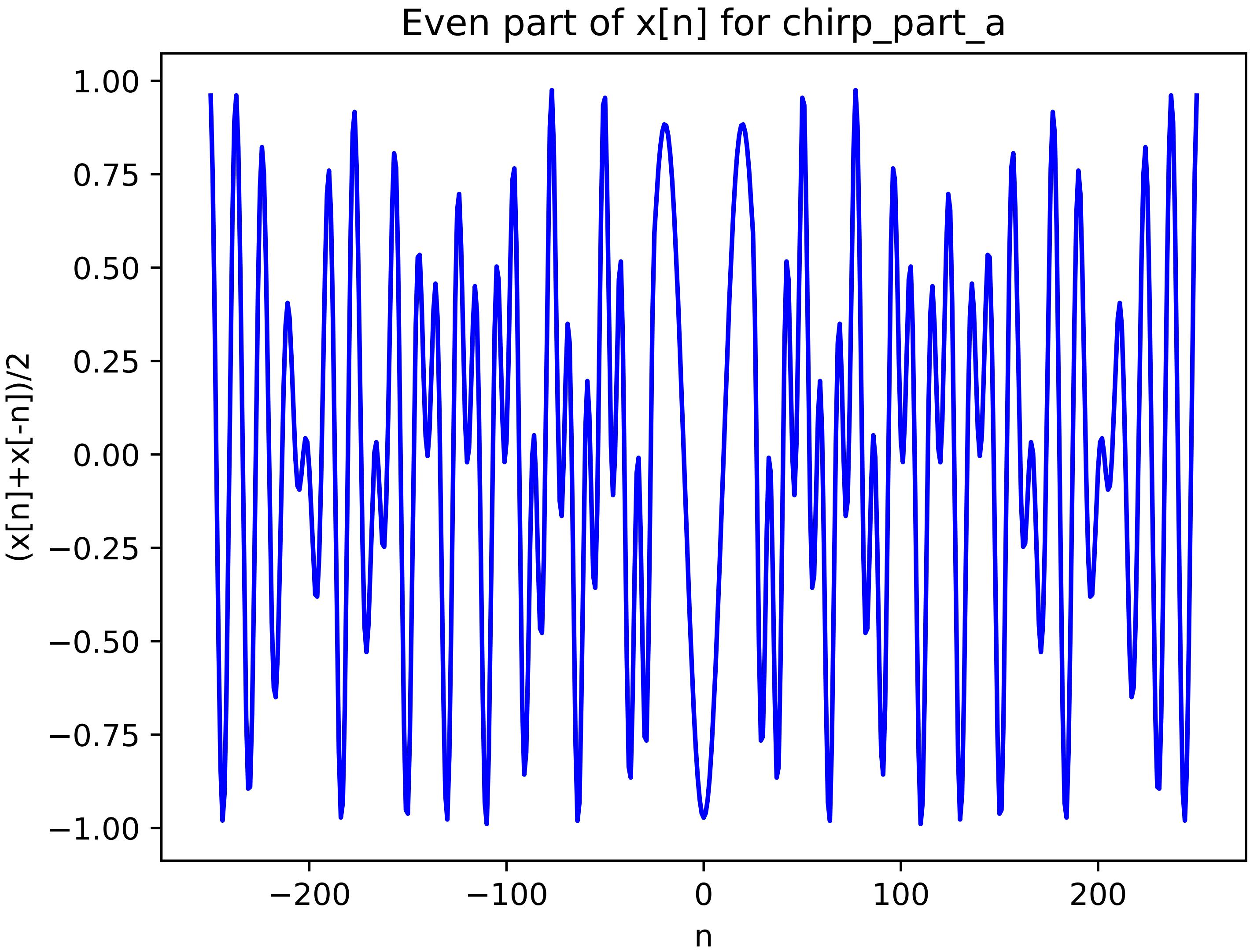
Odd part of x[n] for shifted_sawtooth_part_a
(x[n]-x[-n])/2
n
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
50
100
150
200
250
300
350



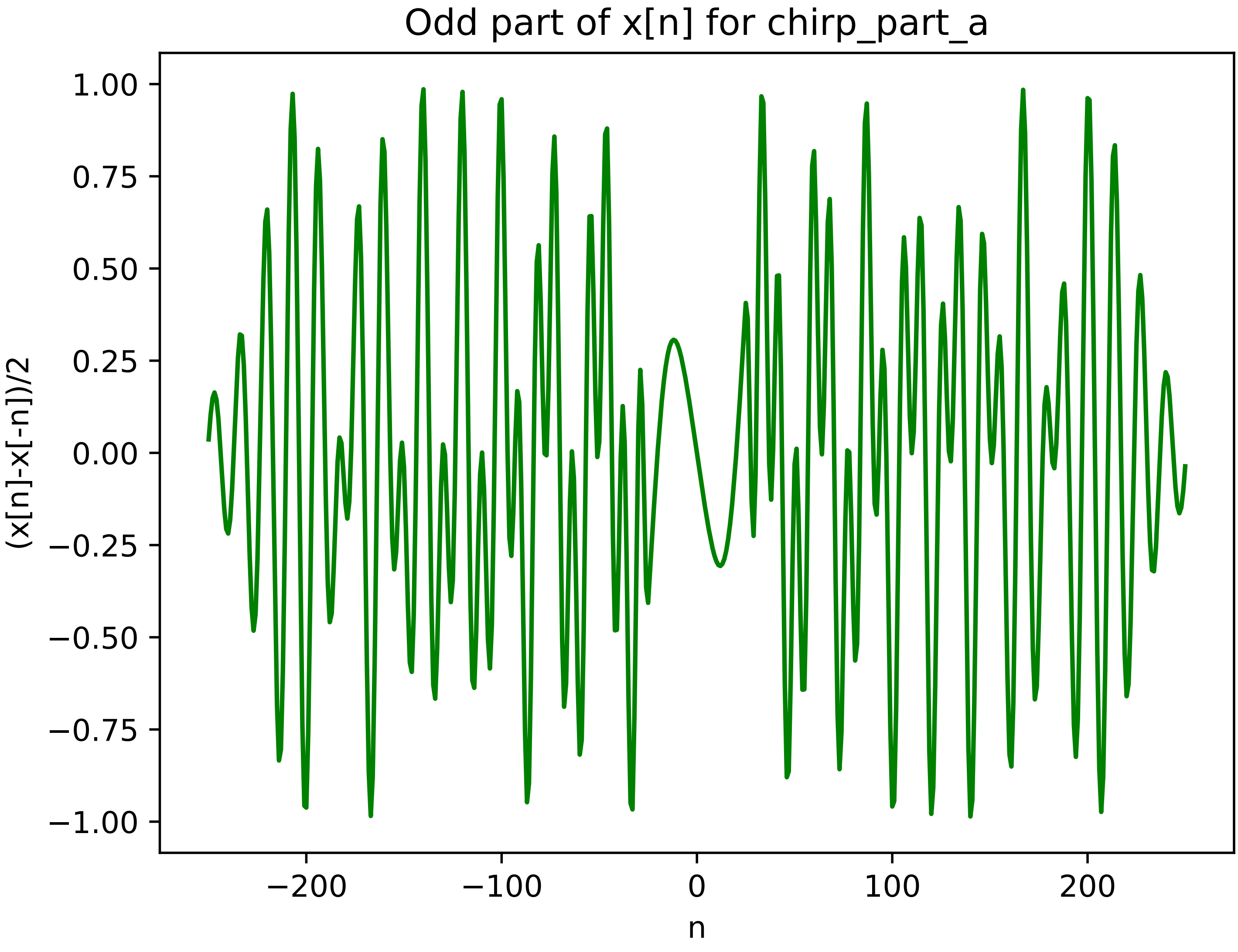
Even part of x[n] for sine_part_a
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
(x[n]+x[-n])/2
−100
−75
−50
−25
0
25
50
75
100
n



Odd part of x[n] for sine_part_a
(x[n]-x[-n])/2
n
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−100
−75
−50
−25
0
25
50
75
100



Even part of x[n] for chirp_part_a
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
(x[n]+x[-n])/2
−200
−100
0
100
200
n

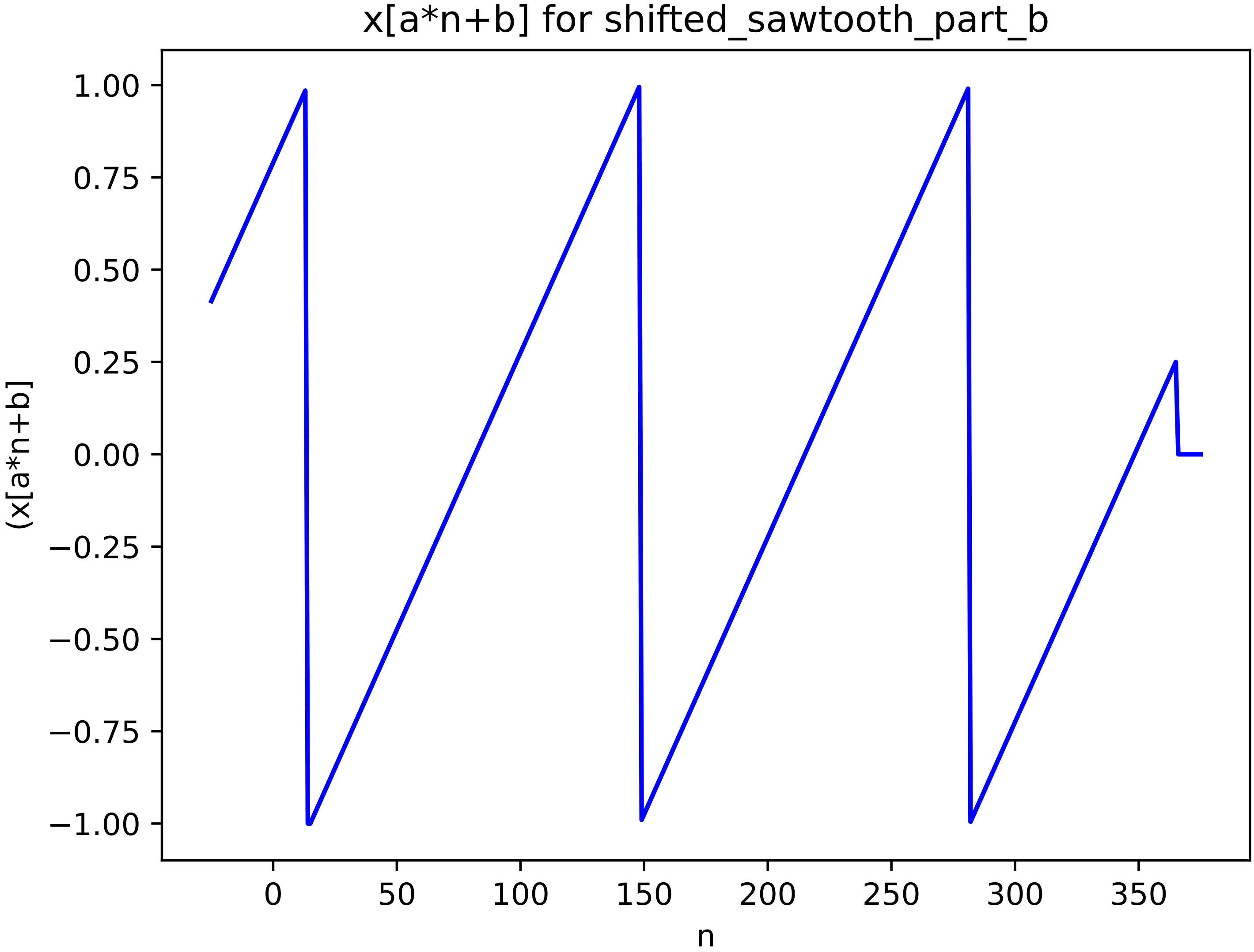

Odd part of x[n] for chirp_part_a
(x[n]-x[-n])/2
n
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−200
−100
0
100
200

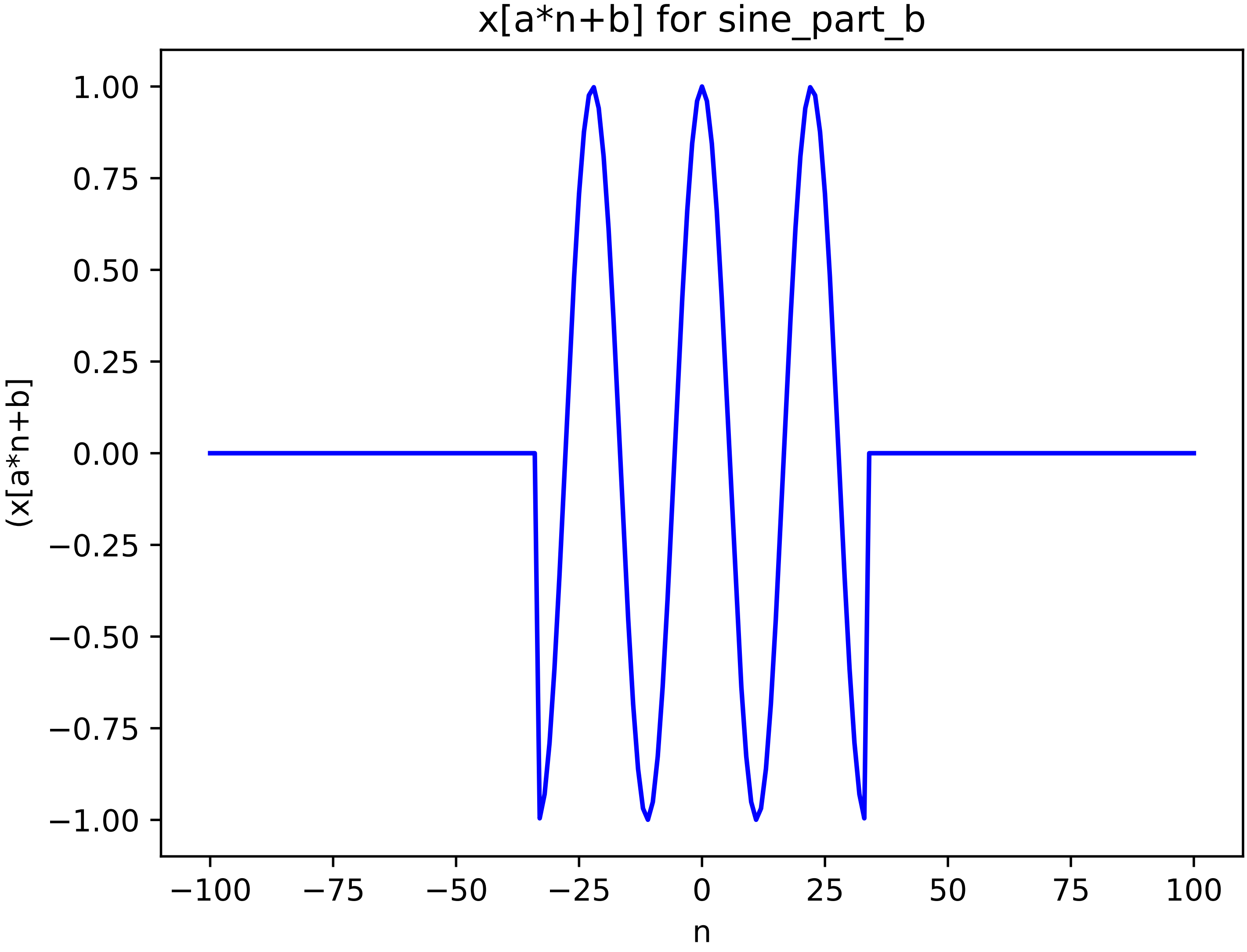
Code for 7-a:

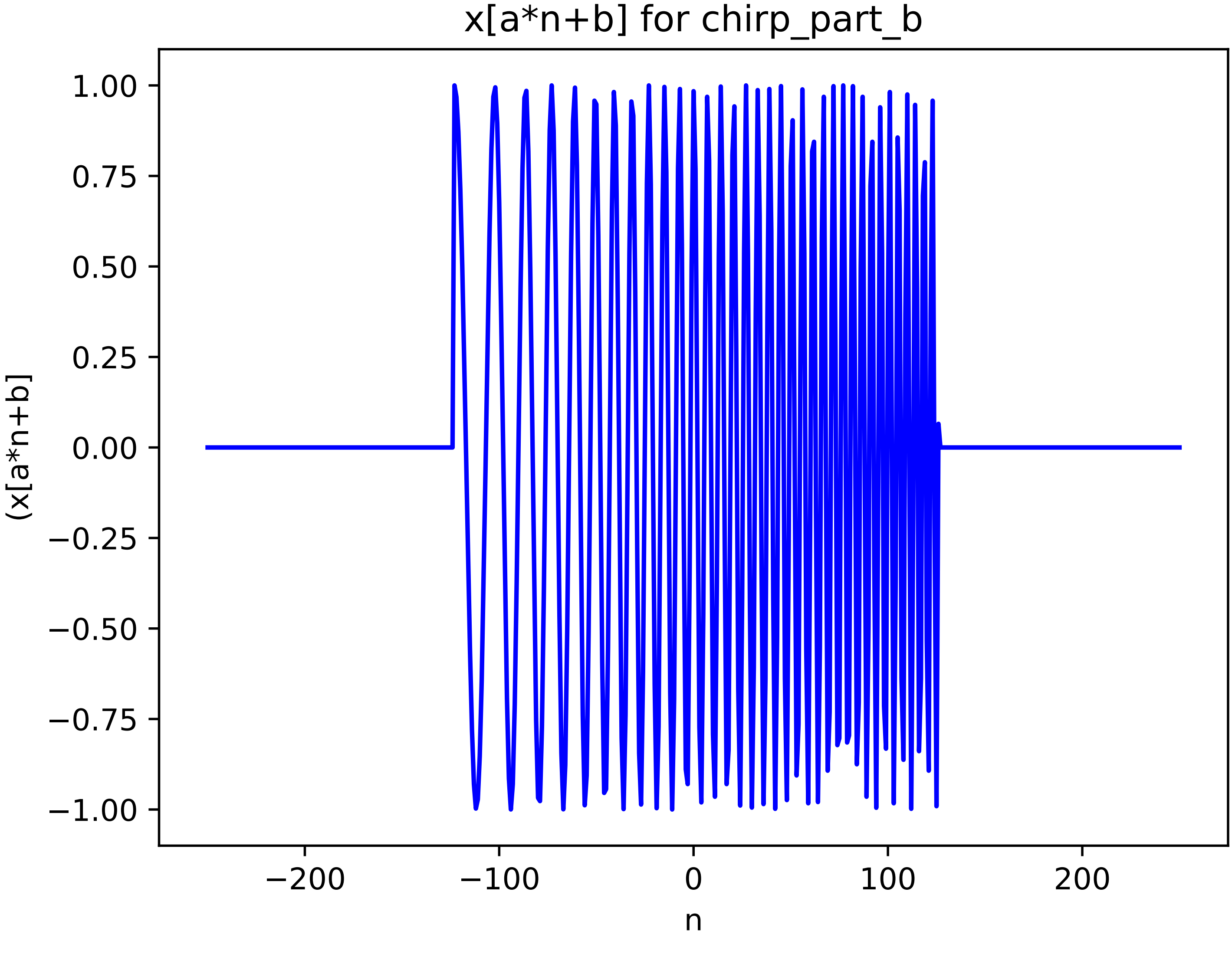
```
import matplotlib.pyplot as plt

# even = 0.5 * (x[n] + x[-n])
# odd  = 0.5 * (x[n] - x[-n])


# Plots for shifted_sawtooth_part_a.csv

path_saw = "shifted_sawtooth_part_a.csv"
file_saw = open(path_saw, "r")
text_saw = file_saw.read()
splitted_saw = text_saw.split(",")

si_saw = int(splitted_saw[0])
signal_saw = [float(i) for i in splitted_saw[1:]]


def x(n, signal):
    return signal[n + si_saw]

def x_odd(n, signal):
    return 0.5 * (x(n, signal) - x(-n, signal))

def x_even(n, signal):
    return 0.5 * (x(n, signal) + x(-n, signal))

range_arr_saw = range(si_saw, si_saw+len(signal_saw))
x_odd_arr_saw = [x_odd(n, signal_saw) for n in range_arr_saw]
x_even_arr_saw = [x_even(n, signal_saw) for n in range_arr_saw]


plt.plot(range_arr_saw, x_even_arr_saw, "b")
plt.xlabel("n")
plt.ylabel("(x[n]+x[-n])/2")
plt.title("Even part of x[n] for shifted_sawtooth_part_a")
plt.show()
plt.plot(range_arr_saw, x_odd_arr_saw, "g")
plt.xlabel("n")
plt.ylabel("(x[n]-x[-n])/2")
plt.title("Odd part of x[n] for shifted_sawtooth_part_a")
plt.show()



# Plots for sine_part_a.csv

path_sine = "sine_part_a.csv"
file_sine = open(path_sine, "r")
text_sine = file_sine.read()
splitted_sine = text_sine.split(",")

si_sine = int(splitted_sine[0])
signal_sine = [float(i) for i in splitted_sine[1:]]

range_arr_sine = range(si_sine, si_sine+len(signal_sine))
x_odd_arr_sine = [x_odd(n, signal_sine) for n in range_arr_sine]
x_even_arr_sine = [x_even(n, signal_sine) for n in range_arr_sine]


plt.plot(range_arr_sine, x_even_arr_sine, "b")
plt.xlabel("n")
plt.ylabel("(x[n]+x[-n])/2")
plt.title("Even part of x[n] for sine_part_a")
```

```python
plt.show()
plt.plot(range_arr_sine, x_odd_arr_sine, "g")
plt.xlabel("n")
plt.ylabel("(x[n]-x[-n])/2")
plt.title("Odd part of x[n] for sine_part_a")
plt.show()



# Plots for chirp_part_a.csv

path_chirp = "chirp_part_a.csv"
file_chirp = open(path_chirp, "r")
text_chirp = file_chirp.read()
splitted_chirp = text_chirp.split(",")

si_chirp = int(splitted_chirp[0])
signal_chirp = [float(i) for i in splitted_chirp[1:]]

range_arr_chirp = range(si_chirp, si_chirp+len(signal_chirp))
x_odd_arr_chirp = [x_odd(n, signal_chirp) for n in range_arr_chirp]
x_even_arr_chirp = [x_even(n, signal_chirp) for n in range_arr_chirp]


plt.plot(range_arr_chirp, x_even_arr_chirp, "b")
plt.xlabel("n")
plt.ylabel("(x[n]+x[-n])/2")
plt.title("Even part of x[n] for chirp_part_a")
plt.show()
plt.plot(range_arr_chirp, x_odd_arr_chirp, "g")
plt.xlabel("n")
plt.ylabel("(x[n]-x[-n])/2")
plt.title("Odd part of x[n] for chirp_part_a")
plt.show()
```

(b)


x[a*n+b] for shifted_sawtooth_part_b
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
(x[a*n+b]
0
50
100
150
200
250
300
350
n


x[a*n+b] for sine_part_b
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
(x[a*n+b]
−100
−75
−50
−25
0
25
50
75
100
n


x[a*n+b] for chirp_part_b
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
(x[a*n+b]
−200
−100
0
100
200
n

Code for 7-b:

```
import matplotlib.pyplot as plt


# Plots for shifted_sawtooth_part_b.csv

path_saw = "shifted_sawtooth_part_b.csv"
file_saw = open(path_saw, "r")
text_saw = file_saw.read()
splitted_saw = text_saw.split(",")

si_saw = int(splitted_saw[0])
a_saw = int(splitted_saw[1])
b_saw = int(splitted_saw[2])
signal_saw = [float(i) for i in splitted_saw[3:]]

def x(n, signal, si):
    if n < si or n >= si + len(signal):
        return 0
    return signal[n + si]

def axb(n, signal, si, a, b):
    return x(a*n+b, signal, si)

range_arr_saw = range(si_saw, si_saw+len(signal_saw))
axb_arr_saw = [axb(n, signal_saw, si_saw, a_saw, b_saw) for n in range_arr_saw]

plt.plot(range_arr_saw, axb_arr_saw, "b")
plt.xlabel("n")
plt.ylabel("(x[a*n+b]")
plt.title("x[a*n+b] for shifted_sawtooth_part_b")
plt.show()


# Plots for sine_part_b.csv

path_sine = "sine_part_b.csv"
file_sine = open(path_sine, "r")
text_sine = file_sine.read()
splitted_sine = text_sine.split(",")

si_sine = int(splitted_sine[0])
a_sine = int(splitted_sine[1])
b_sine = int(splitted_sine[2])
signal_sine = [float(i) for i in splitted_sine[3:]]

range_arr_sine = range(si_sine, si_sine+len(signal_sine))
axb_arr_sine = [axb(n, signal_sine, si_sine, a_sine, b_sine) for n in range_arr_sine]

plt.plot(range_arr_sine, axb_arr_sine, "b")
plt.xlabel("n")
plt.ylabel("(x[a*n+b]")
plt.title("x[a*n+b] for sine_part_b")
plt.show()



# Plots for chirp_part_b.csv

path_chirp = "chirp_part_b.csv"
file_chirp = open(path_chirp, "r")
text_chirp = file_chirp.read()
splitted_chirp = text_chirp.split(",")
```

```
si_chirp = int(splitted_chirp[0])
a_chirp = int(splitted_chirp[1])
b_chirp = int(splitted_chirp[2])
signal_chirp = [float(i) for i in splitted_chirp[3:]]

range_arr_chirp = range(si_chirp, si_chirp+len(signal_chirp))
axb_arr_chirp = [axb(n, signal_chirp, si_chirp, a_chirp, b_chirp) for n in range_arr_chi

plt.plot(range_arr_chirp, axb_arr_chirp, "b")
plt.xlabel("n")
plt.ylabel("(x[a*n+b]")
plt.title("x[a*n+b] for chirp_part_b")
plt.show()
```