Ryerson University

Department of Electrical, Computer, & Biomedical Engineering
Faculty of Engineering & Architectural Science

Course Title:	Signals and Systems II
Course Number:	ELE632
Semester/Year (e.g.F2016)	W2023

Instructor:	Dr. Androutsos

Assignment/Lab Number:	Lab 1
Assignment/Lab Title:	Time-Domain Analysis of Discrete-Time Systems -Part 1

Submission Date:	2023-02-04
Due Date:	2023-02-05

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Zelenovic	Danilo	501032542	08	DZ

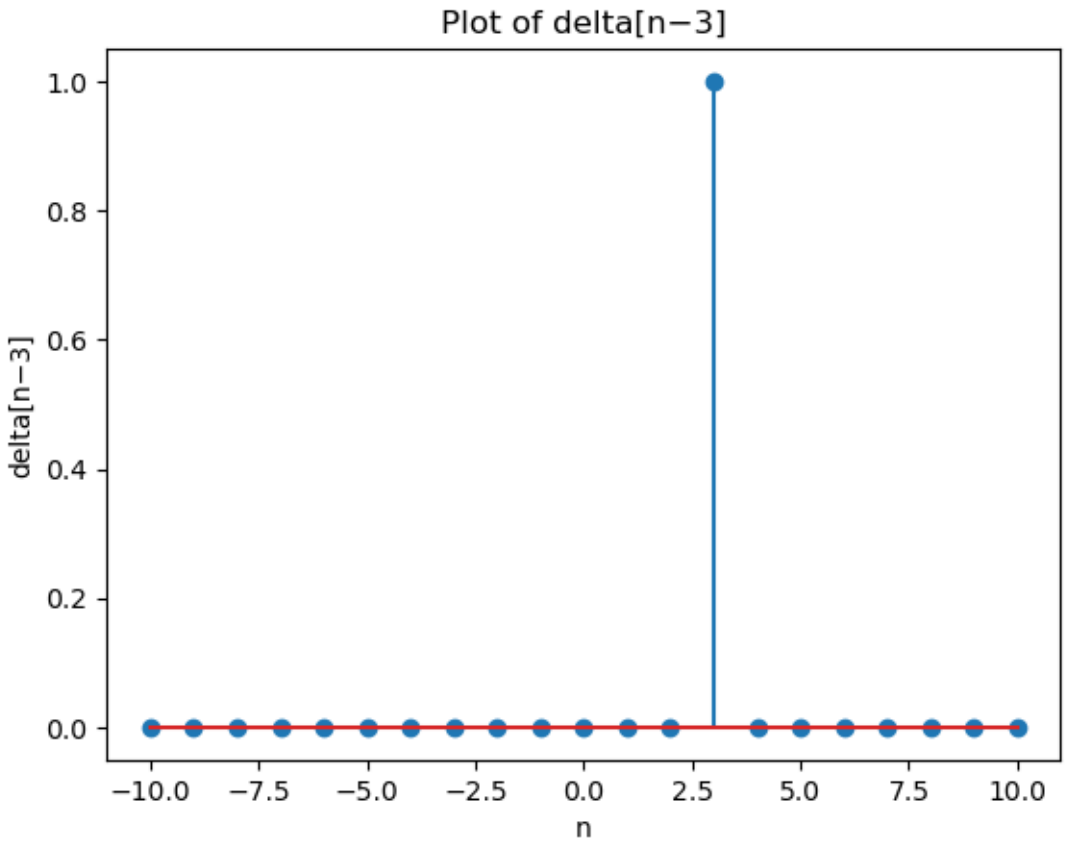
*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a “0” on the work, an “F” in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/senate/current/pol60.pdf

ELE632_Lab1_DaniloZelenovic_501032542_Section08

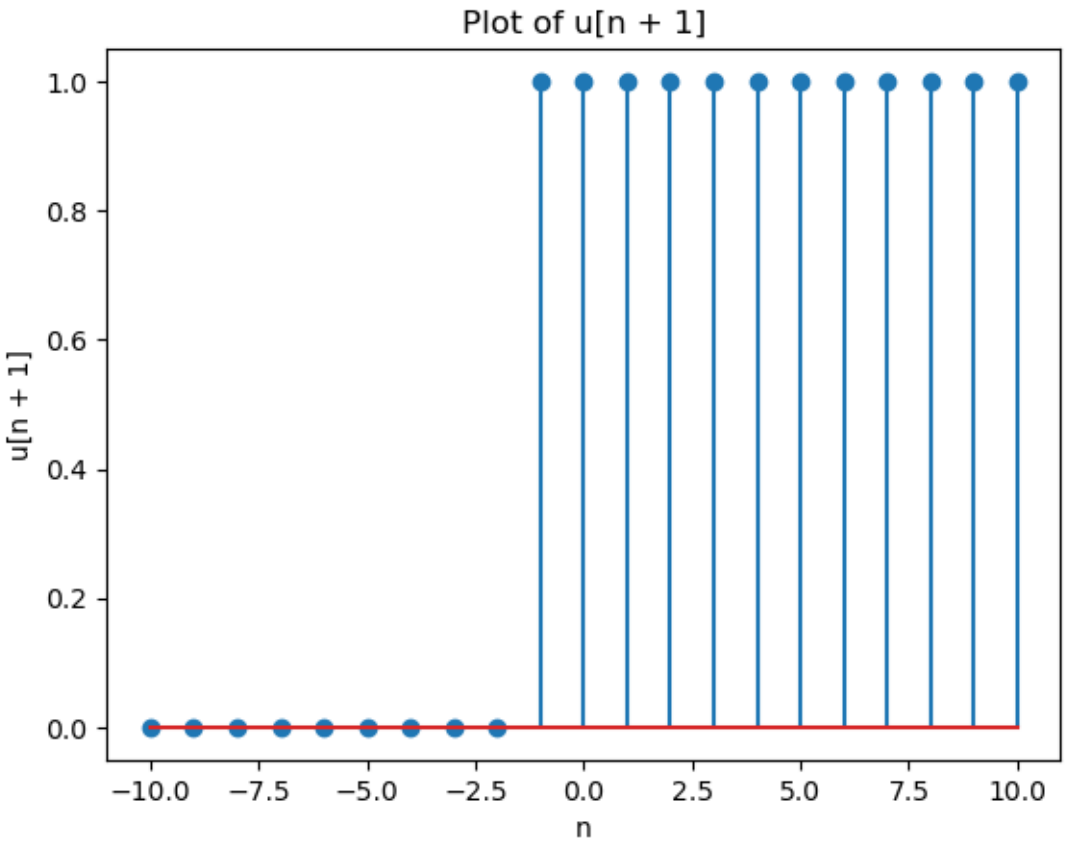
February 4, 2023

1 Part A

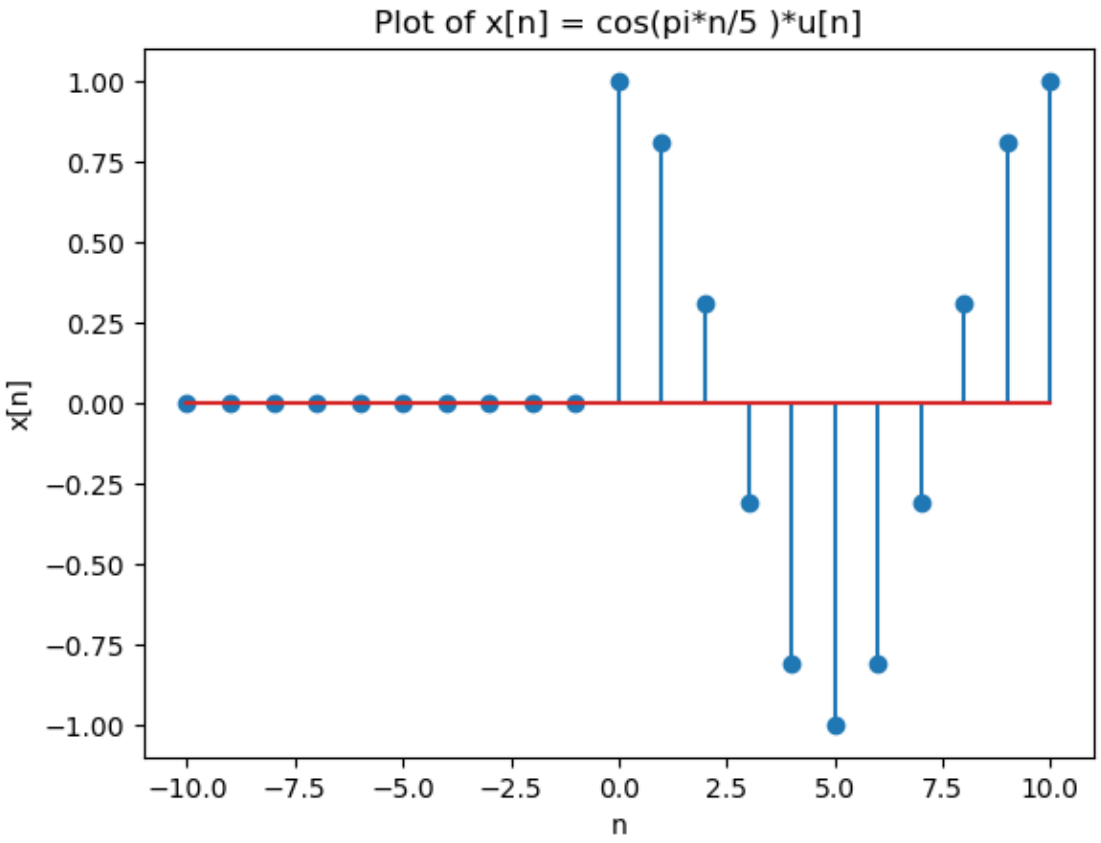
[5]:
```
#PART 1
# I)
import numpy as np
import matplotlib.pyplot as plt

n = np.arange(-10, 11)
delta = np.zeros(len(n))
delta[n==3] = 1

plt.stem(n, delta, use_line_collection=True)
plt.xlabel("n")
plt.ylabel("delta[n-3]")
plt.title("Plot of delta[n-3]")
plt.show()
```



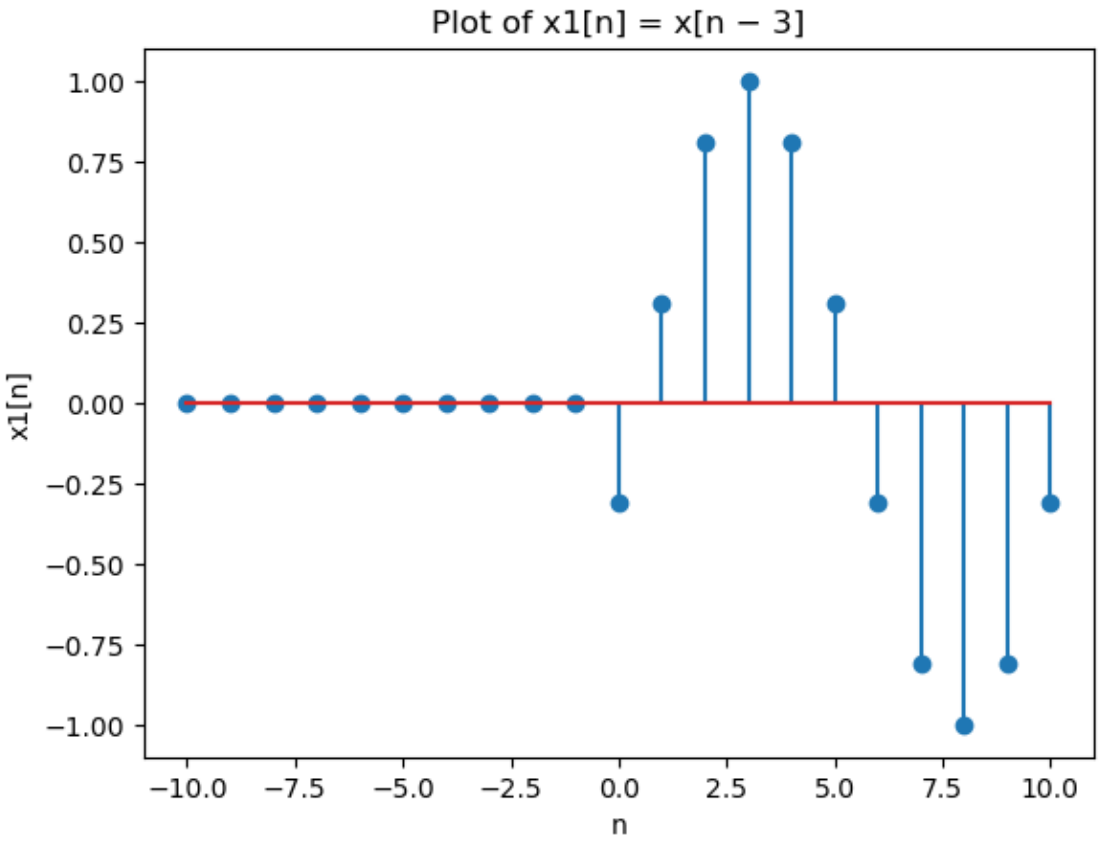
[6]:
```
# II)

import numpy as np
import matplotlib.pyplot as plt

n = np.arange(-10, 11)
u = np.zeros(len(n))
u[n >= -1] = 1

plt.stem(n, u, use_line_collection=True)
plt.xlabel("n")
plt.ylabel("u[n + 1]")
plt.title("Plot of u[n + 1]")
plt.show()
```



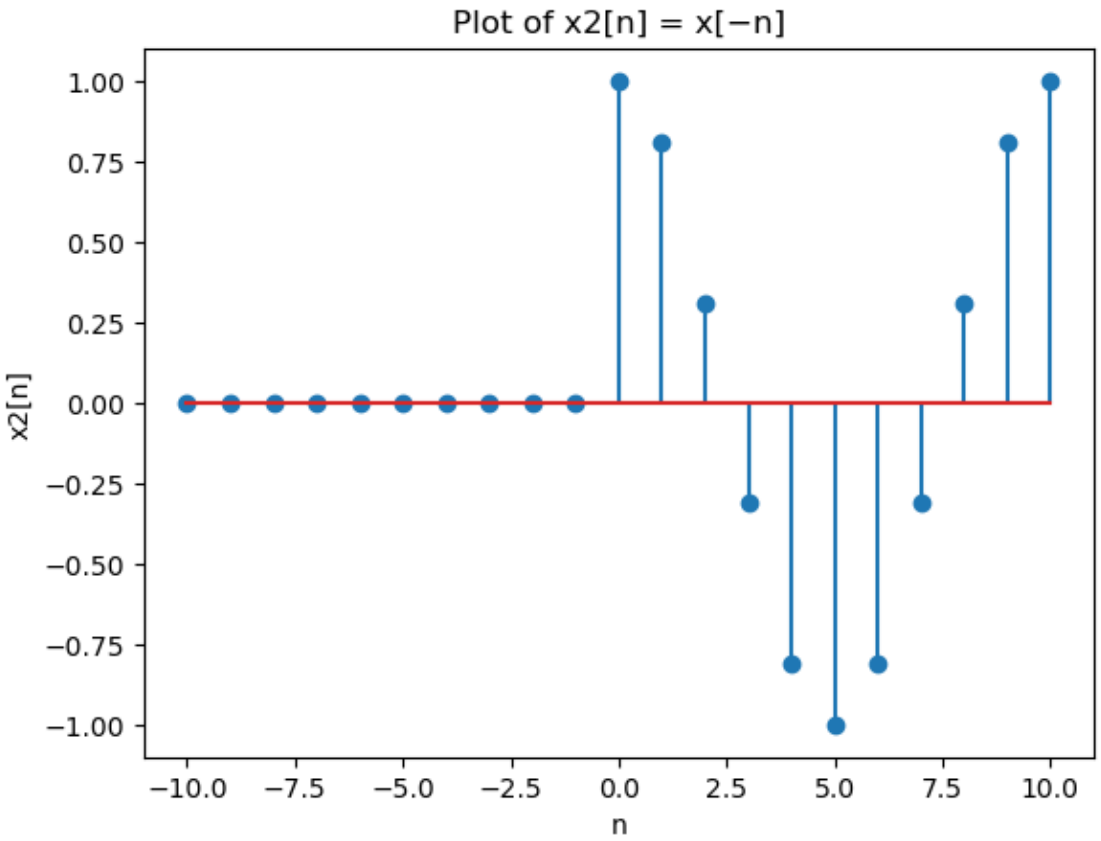
[8]:
```
# III)

import numpy as np
import matplotlib.pyplot as plt

n = np.arange(-10, 11)
u = np.zeros(len(n))
u[n >= 0] = 1
x = np.cos(np.pi * n / 5) * u

plt.stem(n, x, use_line_collection = True)
plt.xlabel("n")
plt.ylabel("x[n]")
plt.title("Plot of x[n] = cos(pi*n/5 )*u[n]")
plt.show()
```



[9]:
```python
# IV)

import numpy as np
import matplotlib.pyplot as plt

n = np.arange(-10, 11)
u = np.zeros(len(n))
u[n >= 0] = 1
x = np.cos(np.pi * (n - 3) / 5) * u

plt.stem(n, x, use_line_collection=True)
plt.xlabel("n")
plt.ylabel("x1[n]")
plt.title("Plot of x1[n] = x[n - 3]")
plt.show()
```

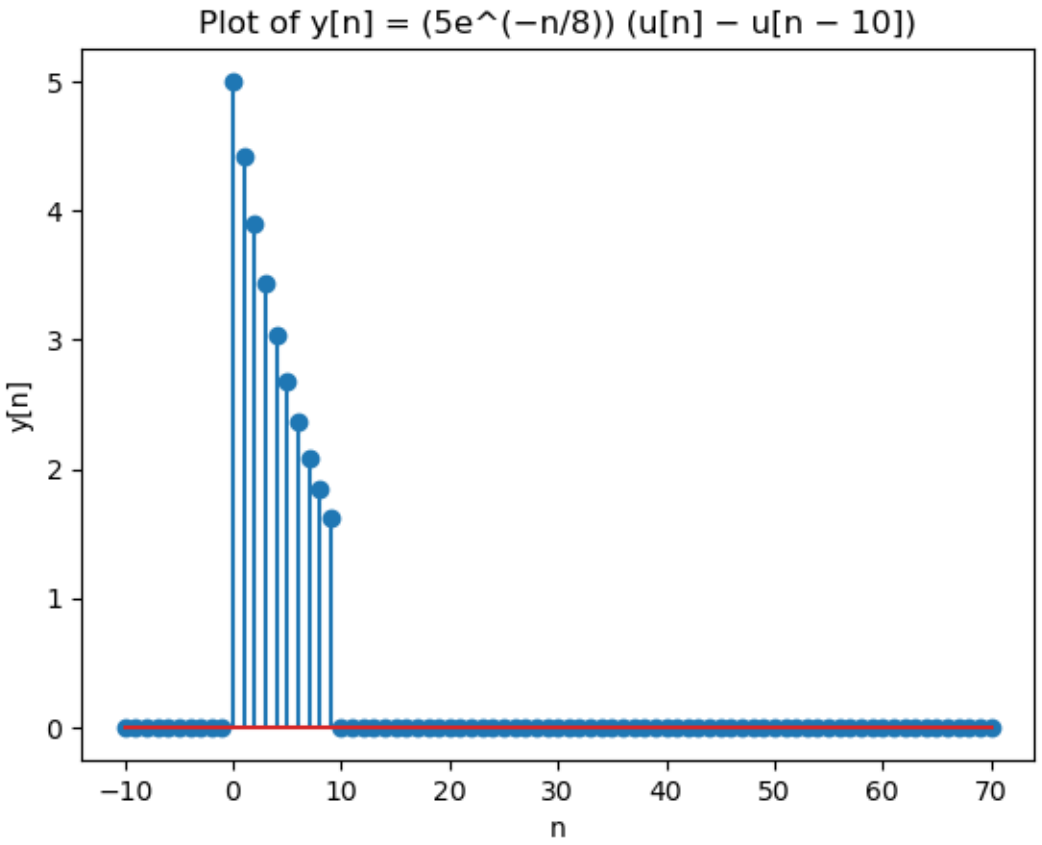


[17]:
```python
# V)

import numpy as np
import matplotlib.pyplot as plt

n = np.arange(-10, 11)
u = np.zeros(len(n))
u[n >= 0] = 1
x = np.cos(np.pi * (-n) / 5) * u

plt.stem(n, x, use_line_collection=True)
plt.xlabel("n")
plt.ylabel("x2[n]")
plt.title("Plot of x2[n] = x[-n]")
plt.show()
```

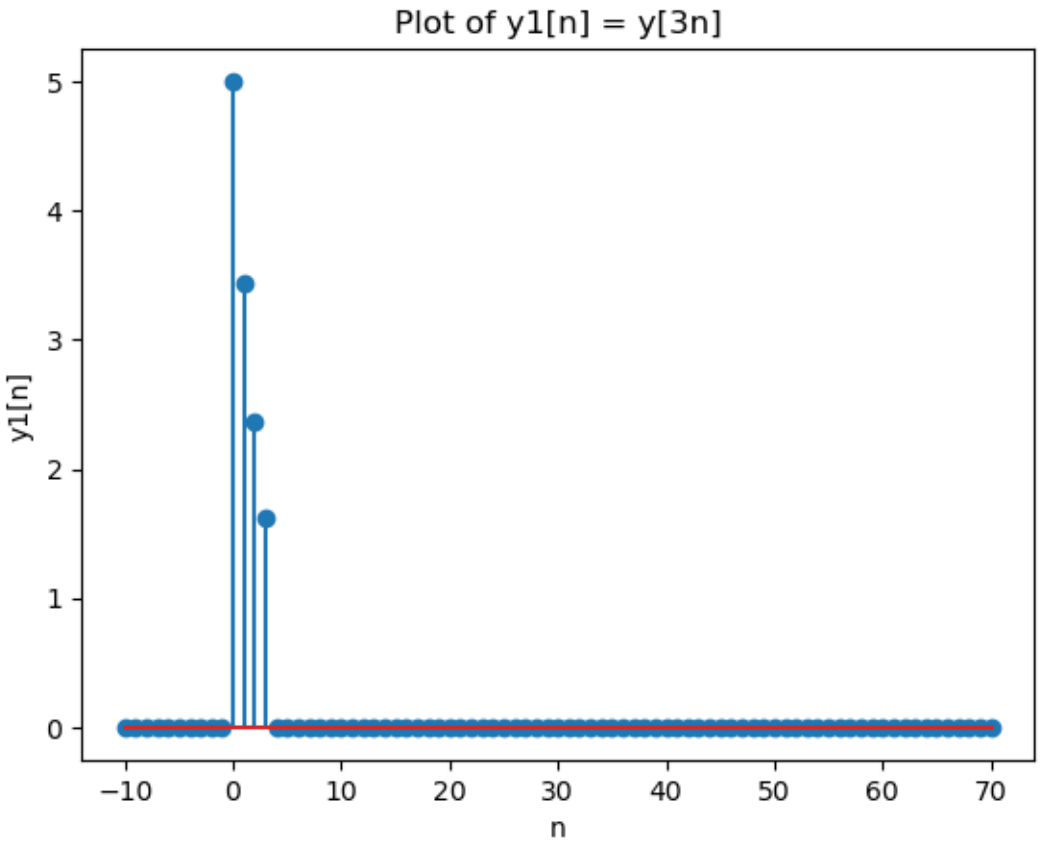


x1 is the same graph as x, but it is shifted by 3 to the right. x2 is similar to x, however it is flipped across the y-axis.

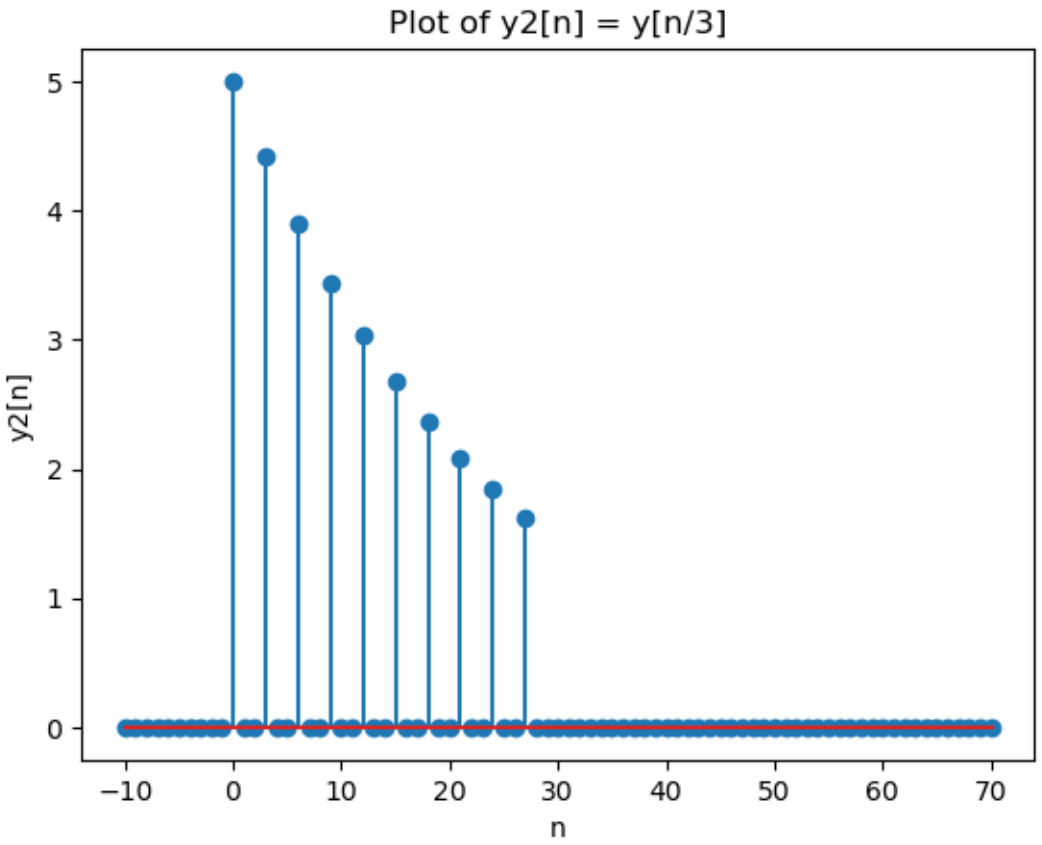
[24]:
```
#Part 2
# I)

import numpy as np
import matplotlib.pyplot as plt

x = np.arange(-10, 71)
u = (x % 1 == 0) * (x >= 0).astype(float)
u2 = (x % 1 == 0) * (x >= 10).astype(float)
y = 5 * np.exp(-x / 8) * (u - u2)

plt.stem(x, y, use_line_collection=True)
plt.xlabel("n")
plt.ylabel("y[n]")
plt.title("Plot of y[n] = (5e^(-n/8)) (u[n] - u[n - 10])")
plt.show()
```



[11]:
```
# II)

import numpy as np
import matplotlib.pyplot as plt

x = np.arange(-10, 71)
z = 3 * x
u = (z % 1 == 0) * (z >= 0).astype(float)
u2 = (z % 1 == 0) * (z >= 10).astype(float)
y = 5 * np.exp(-3 * x / 8) * (u - u2)

plt.stem(x, y, use_line_collection=True)
plt.xlabel("n")
plt.ylabel("y1[n]")
plt.title("Plot of y1[n] = y[3n]")
plt.show()
```



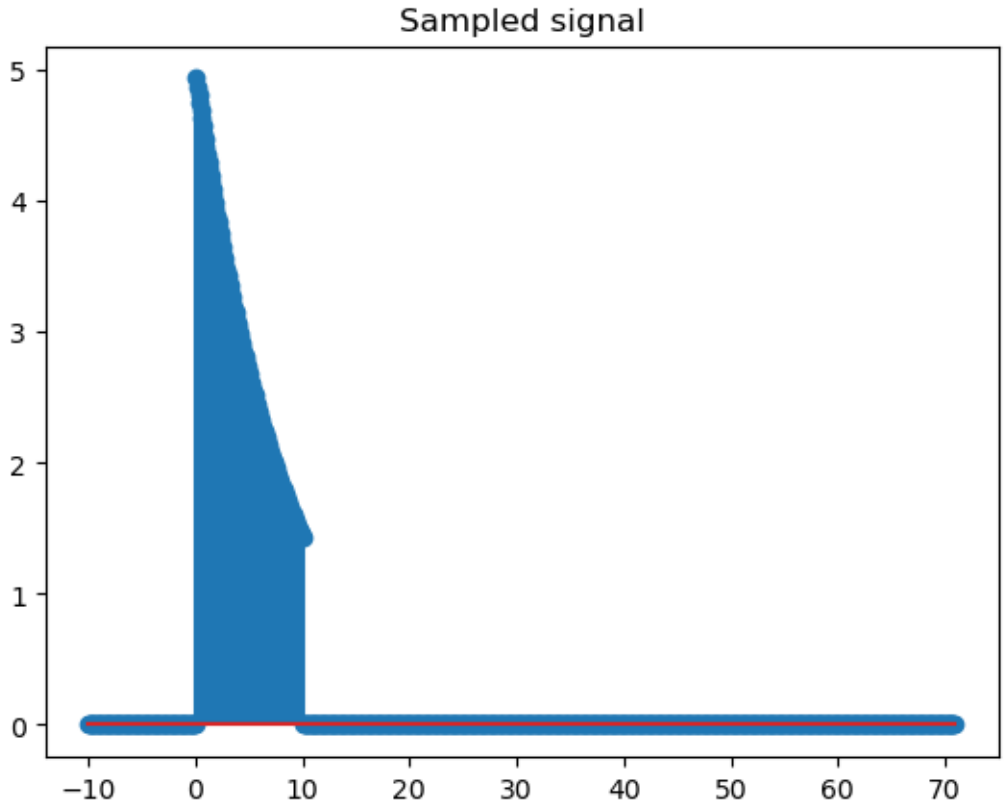
[10]:
```
# III)

import numpy as np
import matplotlib.pyplot as plt

x = np.arange(-10, 71)
z = x / 3
u = (z % 1 == 0) * (z >= 0).astype(float)
u2 = (z % 1 == 0) * (z >= 10).astype(float)
y = 5 * np.exp(-x / (8 * 3)) * (u - u2)

plt.stem(x, y, use_line_collection=True)
plt.xlabel("n")
plt.ylabel("y2[n]")
plt.title("Plot of y2[n] = y[n/3]")
plt.show()
```

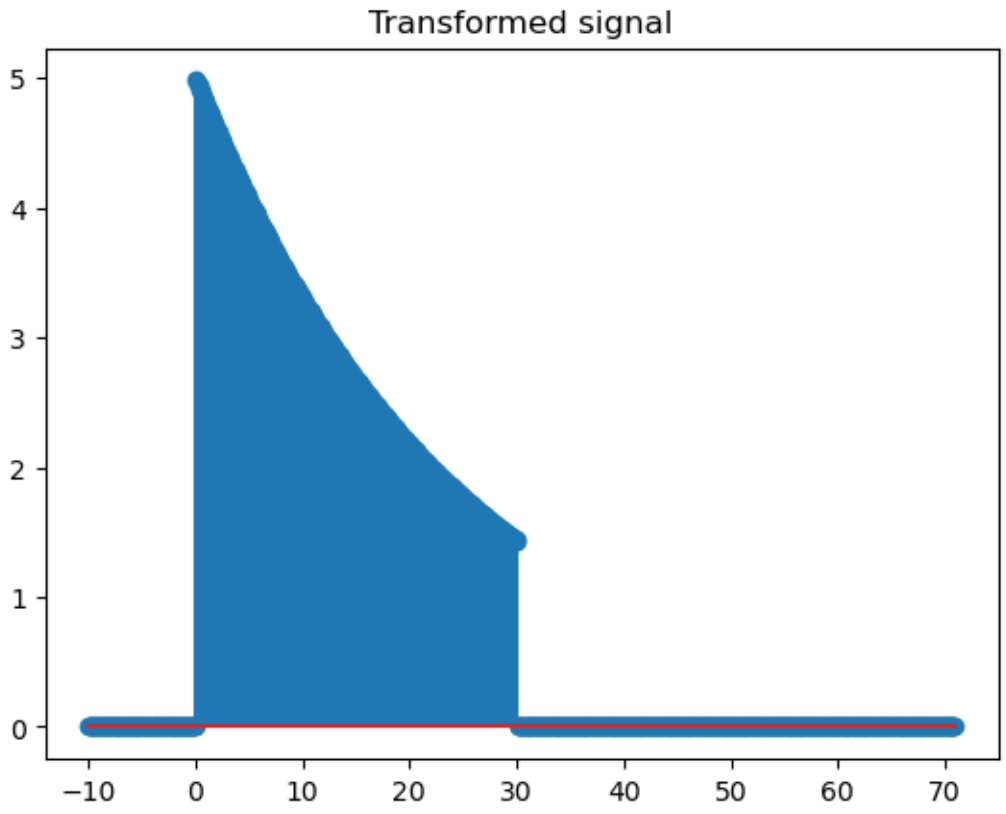


The transformation in y1 is a horizontal compression, and in y2 it is a horizontal expansion (both are by a factor of 3).

[16]:
```
# PART 3
# I)

import numpy as np
import matplotlib.pyplot as plt

x = np.arange(-10,71,0.1)
u = np.where(x >= 0, 1, 0)
u2 = np.where(x >= 10, 1, 0)
z =  5 * np.exp(-x/(8)) * (u-u2)

plt.stem(x, z)
plt.title('Sampled signal')
plt.show()

y = x / 3
u = np.where(y >= 0, 1, 0)
```

```
u2 = np.where(y >= 10, 1, 0)
z =  5 * np.exp(-y/(8)) * (u-u2)

plt.stem(x, z)
plt.title('Transformed signal')
plt.show()
```



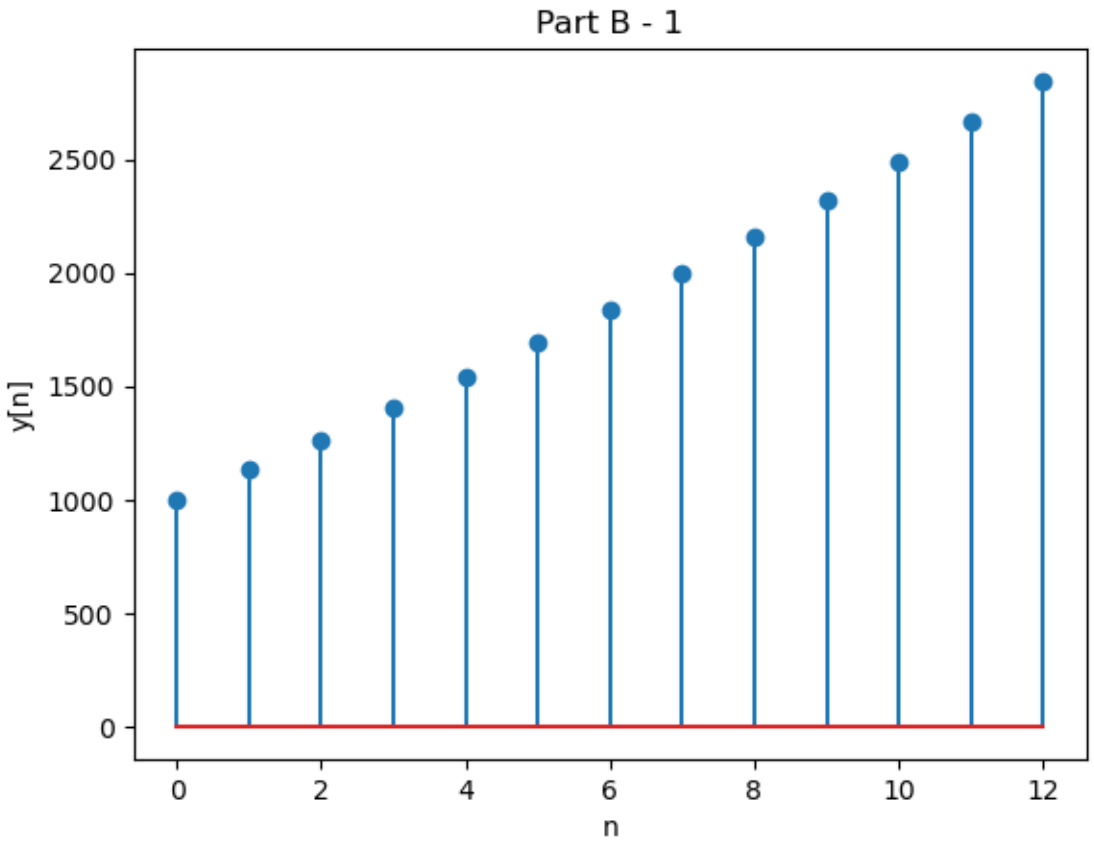


II) The signals here and in part 2(III) are not the same, as part 2 signal was sampled and then had a transformation applied, whereas in part 3, the signal was transformed, then sampled.

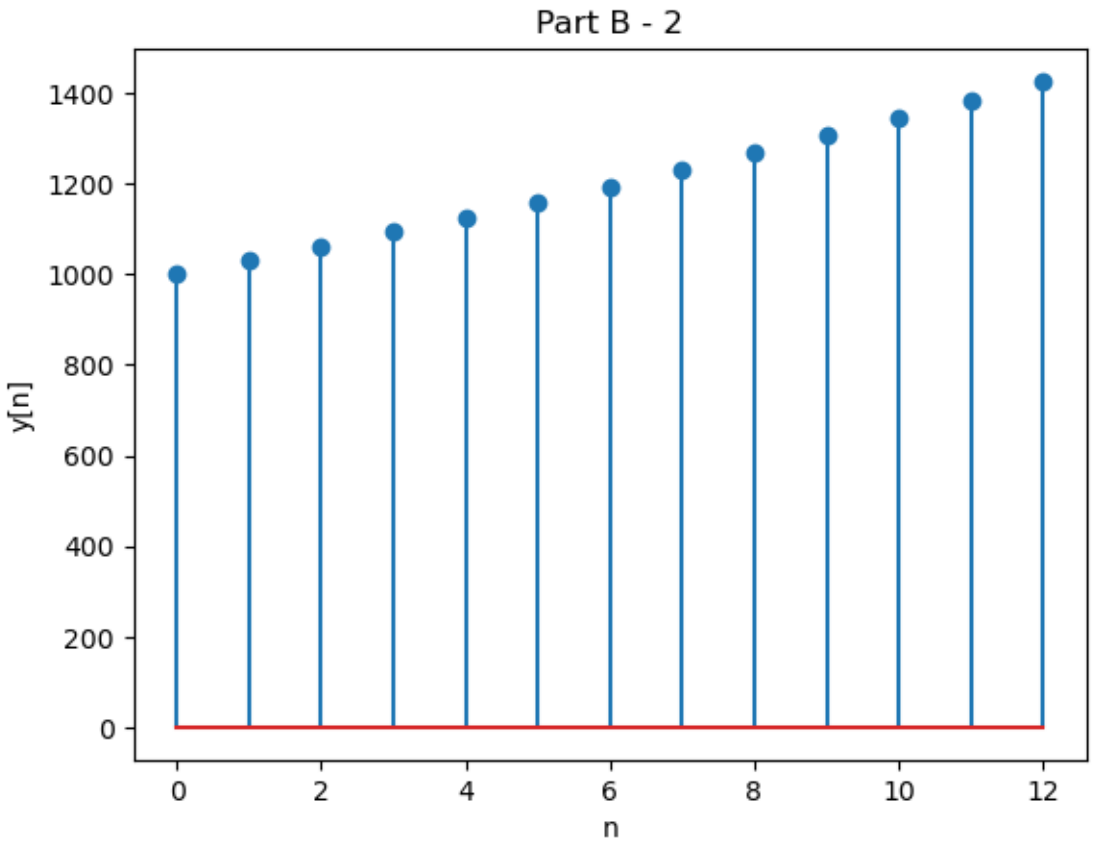
2 PART B

[30]:
```
#1) -> D = 0, I = 2

r = 0.03 # 0.03 = I + 1 = 1 + 2
n = range(0, 13)
y = [1000, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] # 1000 = D+1 = 0+1 * 1000
x = [0, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100] # the␣
 ↪deposits, arbitrary value

for k in range(1, len(n)):
    y[k] = (1 + r) * y[k-1] + x[k]

import matplotlib.pyplot as plt
plt.stem(n, y)
plt.title("Part B - 1")
plt.xlabel("n")
plt.ylabel("y[n]")
```

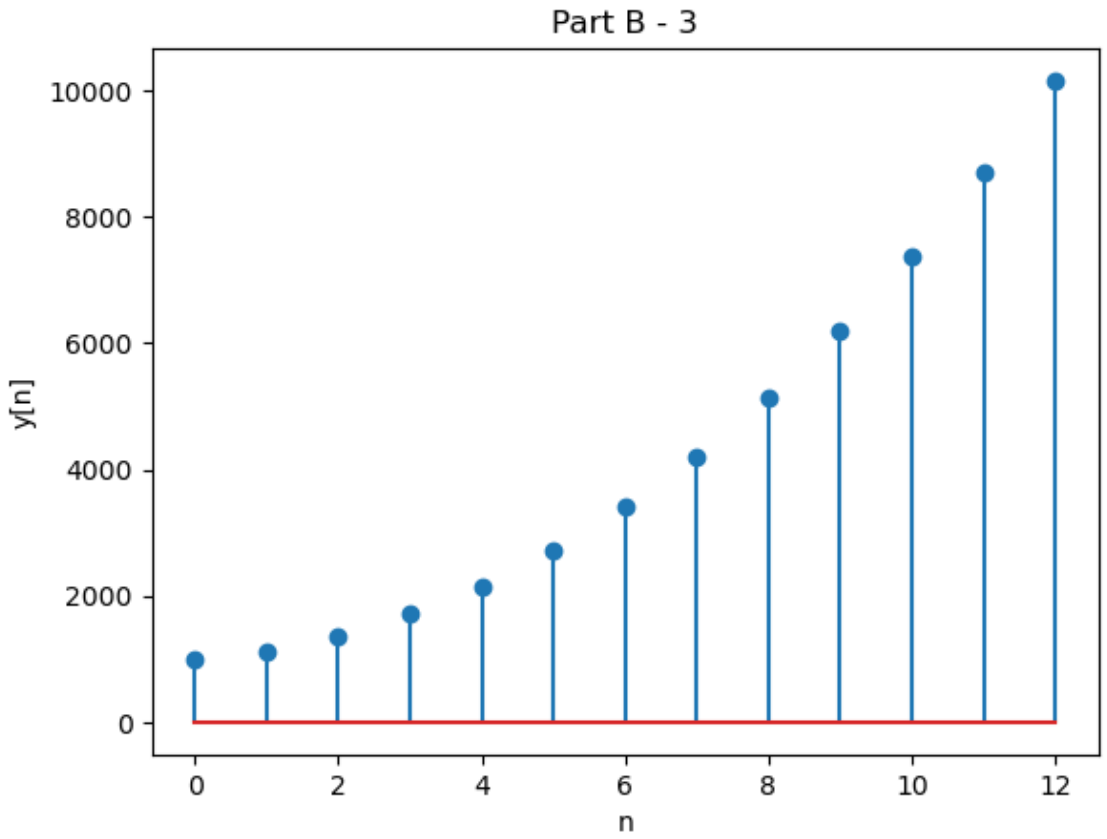
```
plt.show()
```



[29]:
```
# 2)

r = 0.03 # 0.03 = I + 1 = 1 + 2
n = range(0, 13)
y = [1000, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] # 1000 = D+1 = 0+1 * 1000
x = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] # no deposits

for k in range(1, len(n)):
    y[k] = (1 + r) * y[k-1] + x[k]

import matplotlib.pyplot as plt
plt.stem(n, y)
plt.title("Part B - 2")
plt.xlabel("n")
plt.ylabel("y[n]")
plt.show()
```



[31]:
```
#3)

r = 0.03
n = range(0,13)
y = [1000, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
x = [0, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200]

for k in range(1, len(n)):
    y[k] = (1 + r) * y[k-1] + x[k]

import matplotlib.pyplot as plt

plt.stem(n, y)
plt.title('Part B - 3')
plt.xlabel('n')
plt.ylabel('y[n]')
plt.show()
```



3 PART C

[48]:
```
# C - 1

import numpy as np

def maximum_filter(x, N):
    M = len(x)
    x_padded = np.concatenate((np.zeros(N-1), x))
    y = np.zeros(M)
    for k in range(M):
        temp = x_padded[k:k+N]
        y[k] = np.amax(temp)
    return y
```

[53]:
```
# C part 2

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

```
x = np.cos(np.pi*(np.arange(45)/(0+1))) + np.concatenate((np.zeros(44), [1])) -␣
 ↪np.concatenate((np.zeros(43), [1]))
N_values = [4, 8, 12]
for N in N_values:
    y = maximum_filter(x, N)
    n = np.arange(45)
    plt.stem(n, y)
    plt.title(f'N = {N}')
    plt.xlabel('n')
    plt.ylabel('y[n]')
    plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_13344\2834808356.py in <module>
      4 import matplotlib.pyplot as plt
      5
----> 6 x = np.cos(np.pi*(np.arange(45)/(0+1))) + np.concatenate((np.zeros(44),
 ↪[1])) - np.concatenate((np.zeros(43), [1]))
      7 N_values = [4, 8, 12]
      8 for N in N_values:

ValueError: operands could not be broadcast together with shapes (45,) (44,)
```

[ ]:
```
#C part 3:

# When N = 4, function has 4 of highest values, and same with 8 hvaing 8. For N␣
 ↪= 12, it is also only for 8 units, as the next peak is withing range. Then␣
 ↪for all, at x=20 an empty zone occurs for N units
```

4 Part D

[63]:
```
# 1)

import numpy as np

def energy_power(x):
    L = len(x)
    N = L
    E = np.sum(np.abs(x)**2)
    P = (np.linalg.norm(x)**2) / N
    return E, P

n = np.arange(0, 20001)
```

```
x = ((n % 1 == 0) * 1.0 * (n >= 10000))
E, P = energy_power(x)
```

[64]:
```
# Part 2

import numpy as np

def energy_power(x):
    x = np.array([0, -9, -6, -3, 0, 3, 6, 9, 0, 0])
    L = len(x)
    N = L
    E = np.sum(np.abs(x) ** 2)
    P = np.linalg.norm(x) ** 2 / N
    return E, P

n = np.arange(0, 20001)
x = ((n % 1 == 0) * 1.0 * (n >= 10000))
E, P = energy_power(x)
```

[ ]:
```
filepath = "C:\\Users\\zelen\\ELE632_Lab1_DaniloZelenovic_501032542_Section08.
 ↪ipynb"
!jupyter nbconvert --to pdf $filepath
```