Ryerson University

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

Course Title:	Signals and Systems II
Course Number:	ELE 632
Semester/Year (e.g.F2016)	W2023

Instructor:	Dimitri Androutsos

Assignment/Lab Number:	Lab 3
Assignment/Lab Title:	Discrete-Time Fourier Series

Submission Date:	March 12, 2023
Due Date:	March 12, 2023

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Saini	Harsanjam	501055402	10	Harsanjam

*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

Introduction:

In Lab 3, we learned about discrete-time Fourier series (DTFS). We implemented DTFS and its inverse, discrete-time Fourier series (IDFTS), using Python programming language. Through practical exercises, we gained a better understanding of the fundamental properties of DTFS. During the lab session, we were guided through the process of computing the DTFS coefficients of a given sequence. We also learned how to interpret the DTFS coefficients in terms of frequency components. Using the IDFTS implementation, we were able to reconstruct the original signal from the computed DTFS coefficients. In addition, the lab session covered some of the fundamental properties of DTFS, such as symmetry and periodicity. We explored the impact of changes to the input sequence on the resulting DTFS coefficients, and observed how these changes affected the reconstructed signal. Overall, the lab session on DTFS provided us with a solid foundation in the mathematical principles underlying this important signal processing technique. By the end of the session, we were able to apply our newfound knowledge to a wide range of practical applications, such as audio and image processing, communications, and control systems.

A) Discrete Time Fourier Series

1) The fundamental period N_0 is 5, and the fundamental frequency Ω_0 is 0.4π

2) Python code for Part 2:

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

N_0 = 5
n = np.arange(0, N_0)
omega_0 = 2*np.pi/N_0
X_n = 4*np.cos(2.4*np.pi*n) + 2*np.sin(3.2*np.pi*n)

X_r = np.zeros(N_0, dtype=np.complex_)
for r in range(N_0):
    X_r[r] = np.sum(X_n*np.exp(-1j*omega_0*n*r))/N_0

r = n
plt.figure(1)
plt.subplot(311)
plt.stem(n, X_n)
plt.xlabel('n')
plt.ylabel('x[n]')
plt.axis([-0.5, 4.5, -6, 5])
plt.grid()

plt.subplot(312)
plt.stem(r, np.abs(X_r))
plt.xlabel('r')
plt.ylabel('|X_r|')
plt.axis([-0.5, 4.5, -6, 5])
plt.grid()

plt.subplot(313)
plt.stem(r, np.angle(X_r))
plt.xlabel('r')
plt.ylabel('<X_r')
plt.axis([-0.5, 4.5, -6, 5])
plt.grid()

plt.show()
```

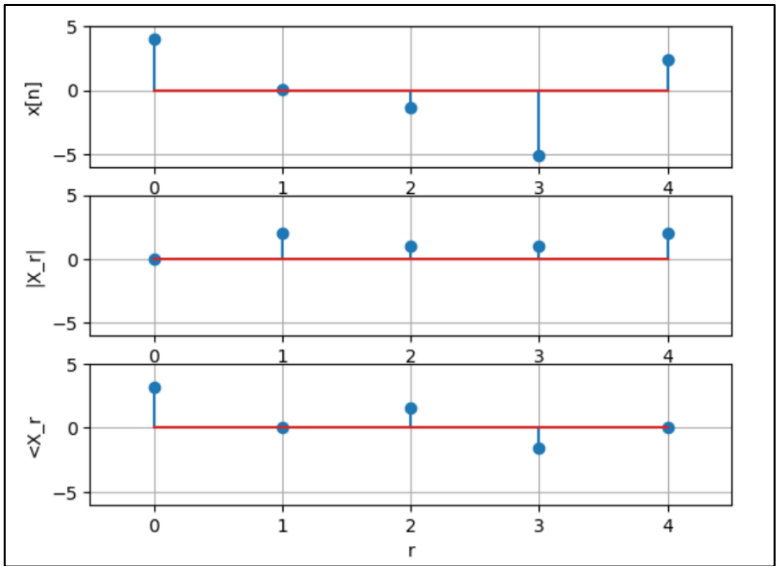

Figure 1: Plot for $x[n]$, the magnitude $|D_r|$ and phase $\angle D_r$ spectra with respect to r.

3) Python code for Part 3:

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

N_0 = 6
n = np.arange(0, N_0)
omega_0 = 2*np.pi/N_0
y_n = np.array([3, 2, 1, 0, 1, 2])
Y_r = np.zeros(N_0, dtype=np.complex_)

for r in range(N_0):
    Y_r[r] = np.sum(y_n*np.exp(-1j*omega_0*n*r))/N_0

r = n
plt.figure(2)
plt.subplot(311)
plt.stem(n, y_n)
plt.xlabel('n')
plt.ylabel('y[n]')
plt.axis([-1, 6, -1, 4])
plt.grid()

plt.subplot(312)
plt.stem(r, np.abs(Y_r))
plt.xlabel('r')
plt.ylabel('|Y_r|')
plt.axis([-1, 6, -1, 3])
plt.grid()

plt.subplot(313)
plt.stem(r, np.angle(Y_r))
plt.xlabel('r')
plt.ylabel('<Y_r')
plt.axis([-1, 6, -4, 2])
plt.grid()

plt.show()
```

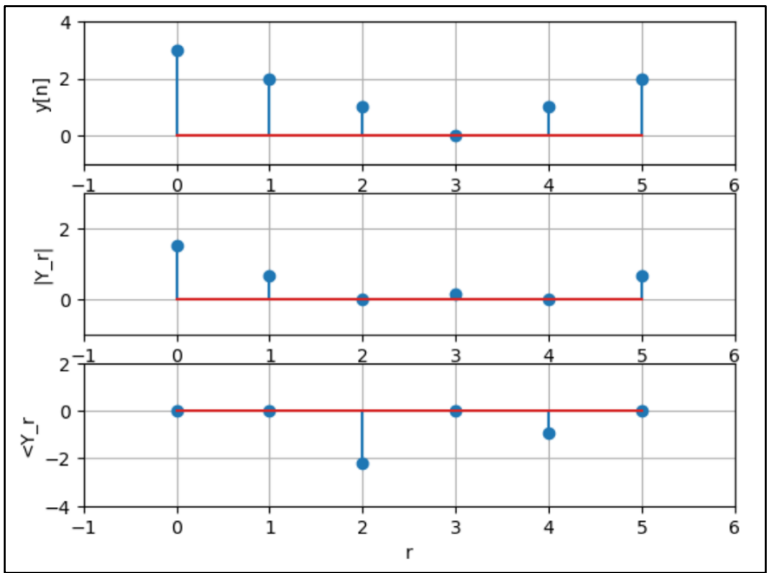

Figure 2: Plot for y[n], the magnitude $|D_r|$ and phase $\angle D_r$ spectra with respect to r.

B) Inverse DTFS and time shifting property

1) Python code for Part 1:

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

N_0 = 32
n = np.arange(0, N_0)
omega_0 = 2*np.pi/N_0
X_r = np.concatenate((np.ones(5), np.zeros(23), np.ones(4)))

x = np.zeros(N_0, dtype=np.complex_)
for r in range(N_0):
    x[r] = np.sum(X_r*np.exp(1j*r*omega_0*n))

plt.figure(3)
plt.stem(n, x)
plt.xlabel('n')
plt.ylabel('x[n]')
plt.axis([0, 32, -4, 10])
plt.grid()
plt.show()
```

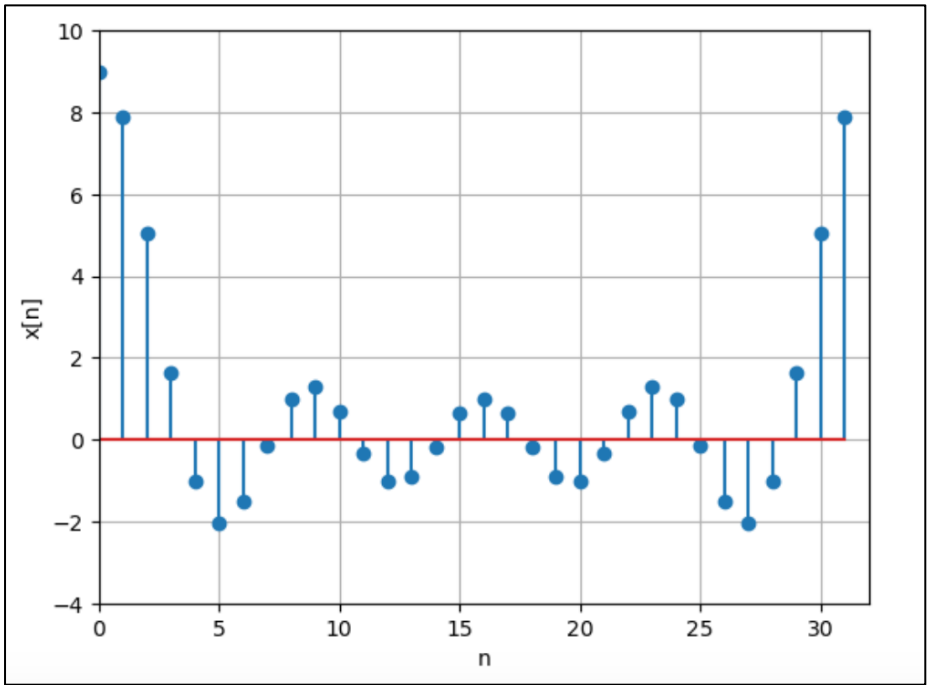

Figure 3: Plot for inverse DTFS $x[n]$ with respect to n

2) Python code for Part 2:

D=0 from my student number. However, that would create the same waveform as part 1. Therefore, to better visualize the differences between the two parts, I chose **D=5**, like the MATLAB report.

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

N_0 = 32
n = np.arange(0, N_0)
omega_0 = 2*np.pi/N_0
X_r = np.concatenate((np.ones(5), np.zeros(23), np.ones(4)))
X_r = X_r * np.exp(-1j*5*omega_0*n)

x = np.zeros(N_0, dtype=np.complex_)
for r in range(N_0):
    x[r] = np.sum(X_r * np.exp(1j*r*omega_0*n))

# x = np.real(np.fft.ifft(X_r)*N_0)

plt.figure(3)
plt.stem(n, x)
plt.xlabel('n')
plt.ylabel('x[n]')
plt.axis([0, 32, -4, 10])
plt.grid()
plt.show()
```

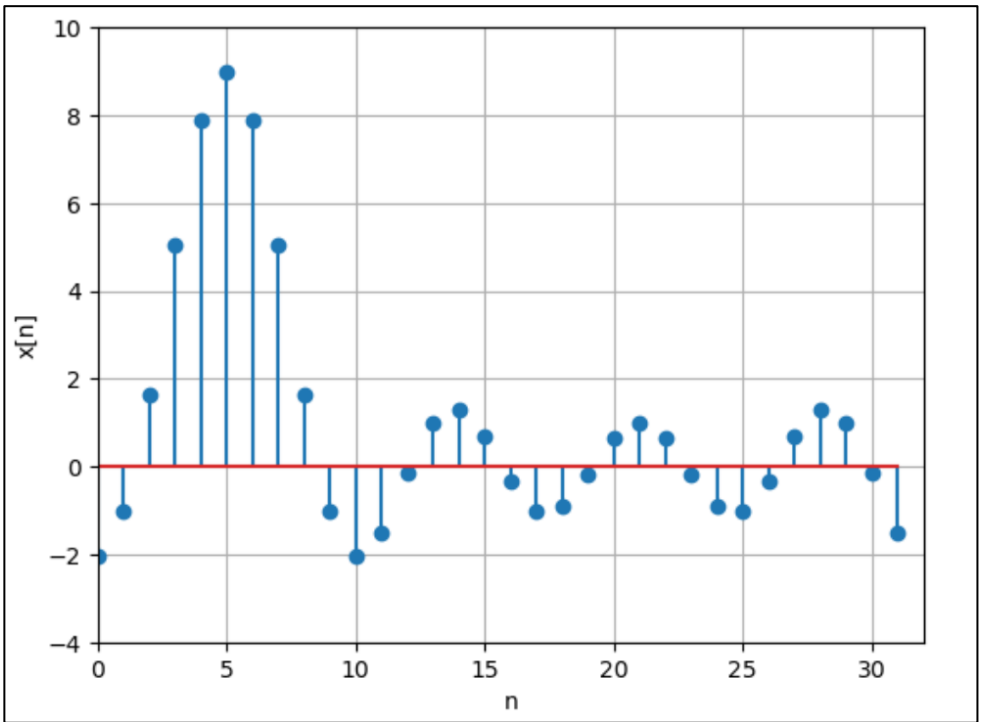

Figure 4: Plot for inverse DTFS $x[n]$ with respect to n by multiplying $e^{-j5\Omega r}$

3). The result in part 2 differs from the x[n] of part 1 by a time shift of 5 to the right.

C) System Response

1) Python code for Part 1:

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

def u(a, n):
    unit = np.ones_like(n)
    unit[n < a] = 0
    return unit

N0 = 32
n = np.arange(N0)
Omega = 2 * np.pi / N0
Hr = (u(0, n) - u(5, n)) + (u(28, n) - u(33, n))

plt.figure(1)
plt.stem(n*Omega, Hr)
plt.xlabel("Omega")
plt.ylabel("H[r]")
plt.title("H[r] with respect to Omega")
plt.grid()

X1 = 4 * np.cos(np.pi * n / 8)
X2 = 4 * np.cos(np.pi * n / 2)
XIr = np.zeros(N0, complex)
X2r = np.zeros(N0, complex)
YIr = np.zeros(N0, complex)
Y2r = np.zeros(N0, complex)

for r in range(N0):
    XIr[r] = np.sum(X1 * np.exp(-1j * Omega *r* n)) / N0
    X2r[r] = np.sum(X2 * np.exp(-1j * Omega *r* n)) / N0
    YIr[r] = Hr[r] * XIr[r]
    Y2r[r] = Hr[r] * X2r[r]

y1 = np.zeros(N0)
y2 = np.zeros(N0)

for i in range(N0):
    y1[i] = np.sum(YIr * np.exp(1j * Omega * i * n))
    y2[i] = np.sum(Y2r * np.exp(1j * Omega * i * n))

plt.figure(2)
plt.stem(n, y1.real)
plt.xlabel("n")
plt.ylabel("Amplitude")
plt.title("Output y[n] for x[n] = 4*cos(pi*n/8)")
plt.grid()
```

```
plt.figure(3)
plt.stem(n, y2.real)
plt.xlabel("n")
plt.ylabel("Amplitude")
plt.title("Output y[n] for x[n] = 4*cos(pi*n/2)")
plt.grid()

plt.figure(4)
plt.stem(n, XIr.real)
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.title("XIr")
plt.grid()

plt.figure(5)
plt.stem(n, X2r.real)
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.title("X2r")
plt.grid()

plt.figure(6)
plt.stem(n, YIr.real)
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.title("Output YIr")
plt.grid()

plt.figure(7)
plt.stem(n, Y2r.real)
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.title("Output Y2r")
plt.grid()

plt.show()
```

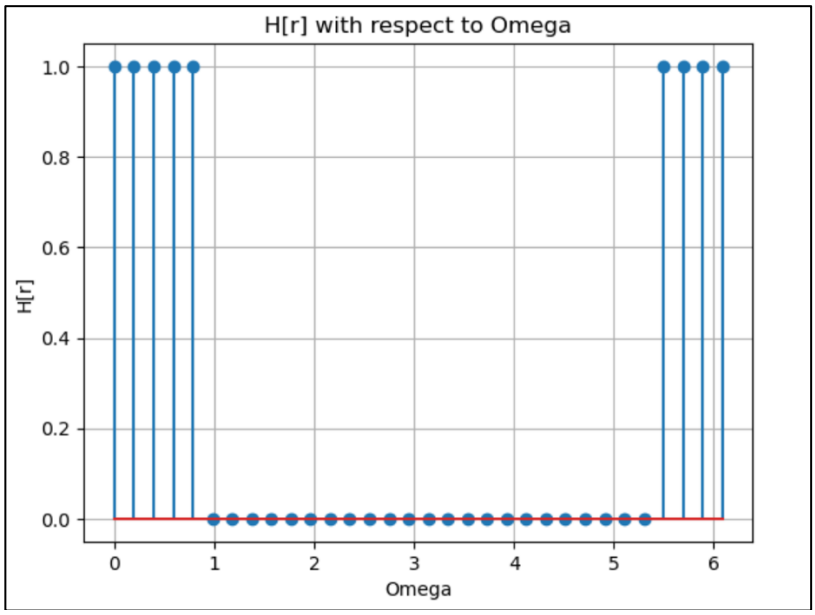

Figure 5: plot of $H[r]$ with respect to Ω_0

2 & 3) $x_1[n] = 4\cos\left(\frac{\pi n}{8}\right)$ *and* $x_2[n] = 4\cos\left(\frac{\pi n}{2}\right)$

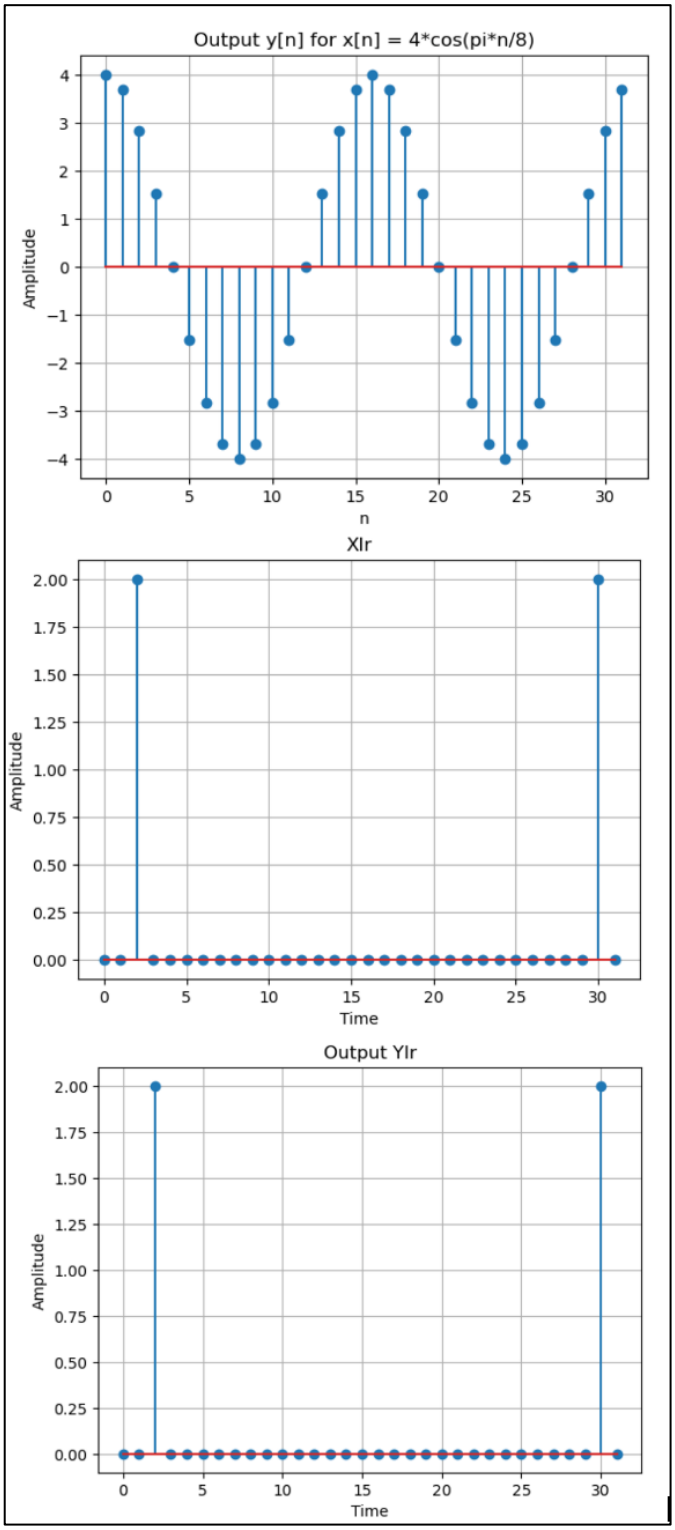

Figure 6: Plot for output y1[n] if input x1[n] is applied to the system. Y1[r] = X1[r]H[r]

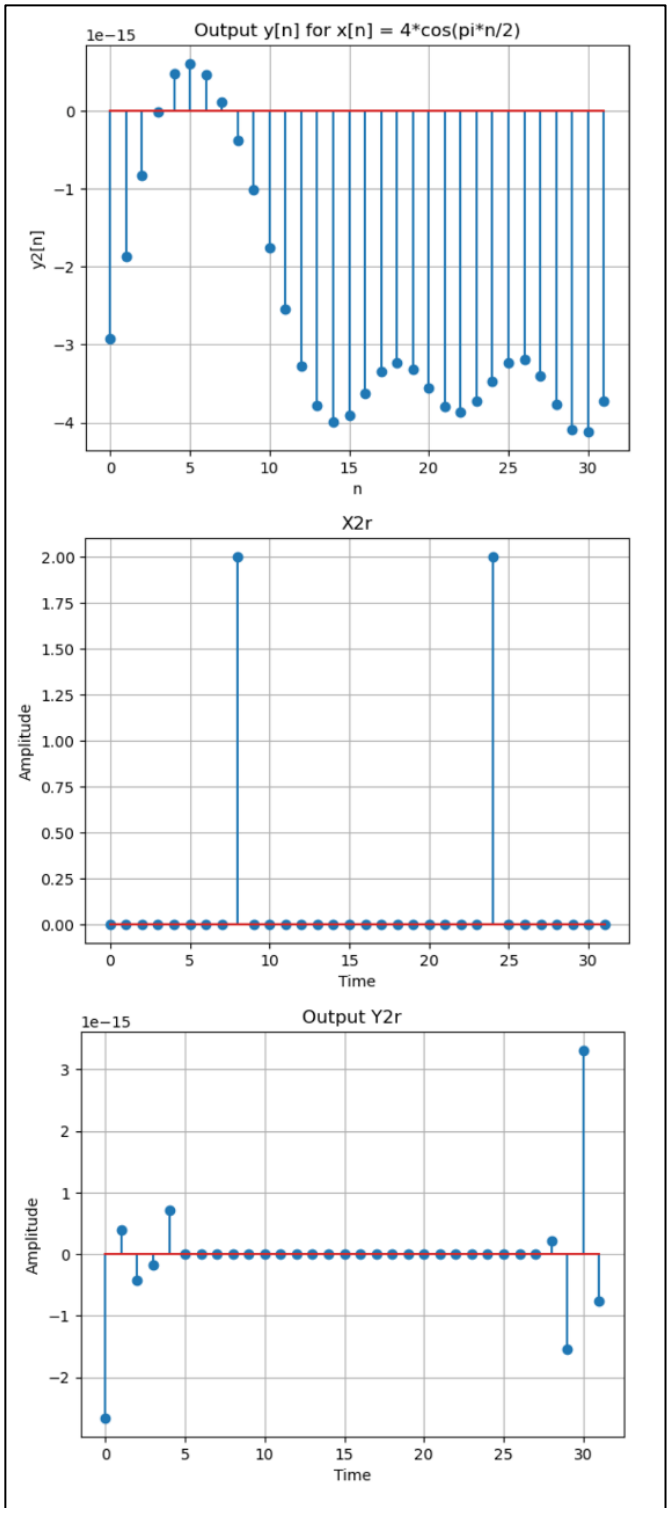

Figure 7: Plot for output y2[n] if input x2[n] is applied to the system. Y2[r] = X2[r]H[r]

4). The results in part 2 and part 3 are different because of their different frequencies. Their different frequencies affect the filter they pass through and the values that are found as a result. $x_1[n]$ falls within the range of H[r] but $x_2[n]$ was surpassed since it is outside the range of the system.