Digital Signal Processing Assignment 1

Sri Varshitha Manduri

Contents

***Abstract*—This manual provides a simple introduction to digital signal processing.**

1 Software Installation

Run the following commands

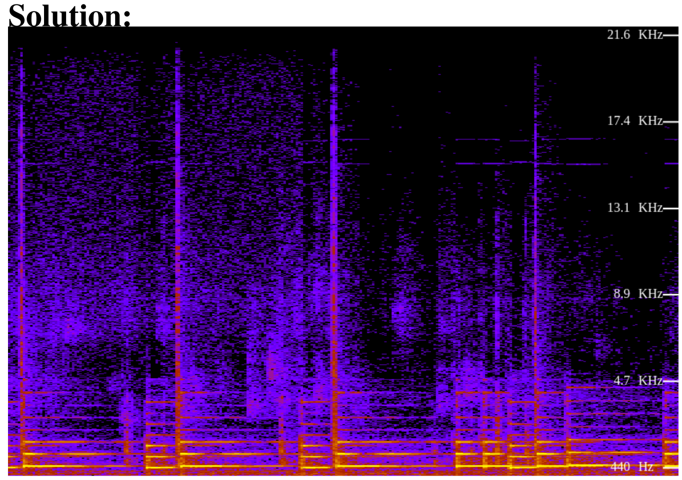
```
sudo apt−get update
sudo apt−get install libffi−dev libsndfile1 python3
    −scipy python3−numpy python3−matplotlib
sudo pip install cffi pysoundfile
```

2 Digital Filter

2.1 Download the sound file from

```
wget https://raw.githubusercontent.com/
    gadepall/
EE1310/master/filter/codes/Sound_Noise.wav
```

2.2 You will find a spectrogram at https://academo.org/demos/spectrum-analyzer. Upload the sound file that you downloaded in Problem 2.1 in the spectrogram and play. Observe the spectrogram. What do you find?
Solution:



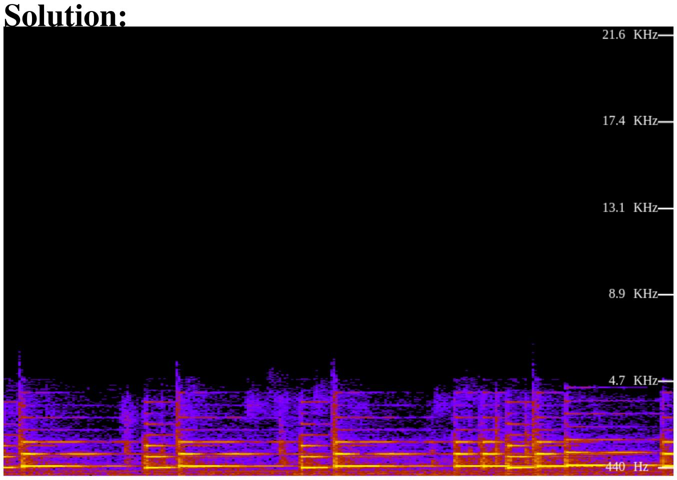
There are a lot of yellow lines between 440 Hz to 5.1 KHz. These represent the synthesizer key tones. Also, the key strokes are audible along with background noise.

2.3 Write the python code for removal of out of band noise and execute the code.
Solution:

```
import soundfile as sf
from scipy import signal

#read .wav file
input_signal,fs = sf.read('Sound_Noise.wav
    ')

#sampling frequency of Input signal
sampl_freq=fs

#order of the filter
order=4

#cutoff frquency 4kHz
cutoff_freq=4000.0

#digital frequency
Wn=2∗cutoff_freq/sampl_freq
```

```
# b and a are numerator and denominator
    polynomials respectively
b, a = signal.butter(order,Wn, 'low')

#filter the input signal with butterworth filter
output_signal = signal.filtfilt(b, a,
    input_signal)
#output_signal = signal.lfilter(b, a,
    input_signal)

#write the output signal into .wav file
sf.write('Sound_With_ReducedNoise.wav',
    output_signal, fs)
```

2.4 The output of the python script in Problem 2.3 is the audio file Sound_With_ReducedNoise.wav. Play the file in the spectrogram in Problem 2.2. What do you observe?
Solution:



The key strokes as well as background noise is subdued in the audio. Also, the signal is blank for frequencies above 5.1 kHz.

3 Difference Equation

3.1 Let

$$x(n) = \left\{ \underset{\uparrow}{1}, 2, 3, 4, 2, 1 \right\} \tag{3.1}$$

Sketch $x(n)$.
Solution:

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

x=np.array([1.0,2.0,3.0,4.0,2.0,1.0])

plt.subplot(2, 1, 1)
plt.stem(range(0,6),x)
plt.title('Digital Filter Input−Output')
plt.ylabel('$x(n)$')
plt.grid()
```

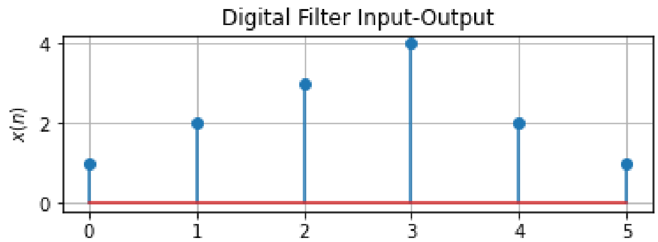


Fig. 3.1

3.2 Let

$$y(n) + \frac{1}{2}y(n-1) = x(n) + x(n-2),$$
$$y(n) = 0, n < 0 \tag{3.2}$$

Sketch $y(n)$.
Solution: The following code yields Fig. 3.2.

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

x=np.array([1.0,2.0,3.0,4.0,2.0,1.0])
k = 20
y = np.zeros(20)


y[0] = x[0]
y[1] = −0.5∗y[0]+x[1]

for n in range(2,k−1):
        if n < 6:
                y[n] = −0.5∗y[n−1]+x[n]+x
                    [n−2]
        elif n > 5 and n < 8:
                y[n] = −0.5∗y[n−1]+x[n−2]
        else:
                y[n] = −0.5∗y[n−1]
print(y)

#subplots
plt.subplot(2, 1, 1)
plt.stem(range(0,6),x)
plt.title('Digital Filter Input−Output')
plt.ylabel('$x(n)$')
plt.grid()# minor
```

```
plt.subplot(2, 1, 2)
plt.stem(range(0,k),y)
plt.xlabel('$n$')
plt.ylabel('$y(n)$')
plt.grid()# minor
```

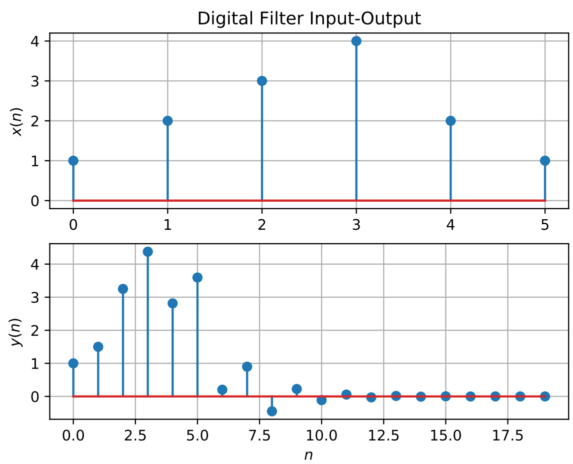


Fig. 3.2

3.3 Repeat the above exercise using a C code.
Solution:

```
#include <stdio.h>
float yn(int∗ x,int n){
    int y=0;
    if(n==0){y =∗p1;}
    else if(n==1){
    y=∗(x+1)+yn(x,0);
    }
    else if(n<6){
    y = ∗(x+n−1) + ∗(x+n−3) − yn(x,n−1)
        ∗0.5;
    }
    else {
    y = −yn(x,n−1)∗0.5;
    }
    return y;
}

int main(){
FILE∗ fp;
const int Points;
int x[];
float y[Points];
int i;
x={1,2,3,4,2,1};
Points=20;

for(int i=0;i<number_Of_Points;i++){
y[i]=yn(x,i);
}

fp = fopen("y(n).dat","w");

for(i=0;i<number_Of_Points;i++){
fprintf(fp,"%f\n",y[i]);
}

fclose(fp);
return 0;
}
```

For plotting:

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


y = np.loadtxt("y(n).dat",dtype = "double")
x = np.array([1,2,3,4,2,1])

# ploting graphs
plt.subplot(211)
plt.stem(np.arange(len(x)),x)
plt.xlabel("n")
plt.ylabel("$x(n)$")
plt.grid()

plt.subplot(212)
plt.stem(np.arange(len(y)),y)
plt.xlabel("n")
plt.ylabel("$y(n)$")
plt.grid()

plt.show()
```

4 Z-transform

4.1 The Z-transform of $x(n)$ is defined as

$$X(z) = \mathcal{Z}\{x(n)\} = \sum_{n=-\infty}^{\infty} x(n)z^{-n} \tag{4.1}$$

Show that

$$\mathcal{Z}\{x(n-1)\} = z^{-1}X(z) \tag{4.2}$$

and find

$$\mathcal{Z}\{x(n-k)\} \tag{4.3}$$

Solution: From (4.1),

$$\mathcal{Z}\{x(n-k)\} = \sum_{n=-\infty}^{\infty} x(n-1)z^{-n} \tag{4.4}$$

$$= \sum_{n=-\infty}^{\infty} x(n)z^{-n-1} = z^{-1}\sum_{n=-\infty}^{\infty} x(n)z^{-n} \tag{4.5}$$

resulting in (4.2). Similarly, it can be shown that

$$\mathcal{Z}\{x(n-k)\} = z^{-k}X(z) \tag{4.6}$$

4.2 Obtain $X(z)$ for $x(n)$ defined in problem 3.1.

Solution:

$$x(n) = \{1, 2, 3, 4, 2, 1\} \tag{4.7}$$

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n} \tag{4.8}$$

$$= z^{-1} + 2z^{-2} + 3z^{-3} + 4z^{-4} + 2z^{-2} + 1z^{-1} \tag{4.9}$$

4.3 Find

$$H(z) = \frac{Y(z)}{X(z)} \tag{4.10}$$

from (3.2) assuming that the Z-transform is a linear operation.
Solution: Applying (4.6) in (3.2),

$$Y(z) + \frac{1}{2}z^{-1}Y(z) = X(z) + z^{-2}X(z) \tag{4.11}$$

$$\implies \frac{Y(z)}{X(z)} = \frac{1+z^{-2}}{1+\frac{1}{2}z^{-1}} \tag{4.12}$$

4.4 Find the Z transform of

$$\delta(n) = \begin{cases} 1 & n = 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.13}$$

and show that the Z-transform of

$$u(n) = \begin{cases} 1 & n \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.14}$$

is

$$U(z) = \frac{1}{1-z^{-1}}, \quad |z| > 1 \tag{4.15}$$

Solution: It is easy to show that

$$\delta(n) \overset{\mathcal{Z}}{\rightleftharpoons} 1 \tag{4.16}$$

and from (4.20),

$$U(z) = \sum_{n=0}^{\infty} z^{-n} \tag{4.17}$$

$$= \frac{1}{1-z^{-1}}, \quad |z| > 1 \tag{4.18}$$

4.5 Show that

$$a^n u(n) \overset{\mathcal{Z}}{\rightleftharpoons} \frac{1}{1-az^{-1}} \quad |z| > |a| \tag{4.19}$$

Solution:

$$Since, u(n) = \begin{cases} 1 & n \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.20}$$

$$a^n u(n) = \left\{1, a^1, a^2..\right\} \tag{4.21}$$

Z transform ,

$$= \sum_{n=-\infty}^{\infty} a^n z^{-n} \tag{4.22}$$

$$= 1 + az^{-1} + a^2z^{-2} + \dots \tag{4.23}$$

$$a^n u(n) \overset{\mathcal{Z}}{\rightleftharpoons} \frac{1}{1-az^{-1}}(|z| > |a|) \tag{4.24}$$

using the formula for the sum of an infinite geometric progression.

4.6 Let

$$H\left(e^{j\omega}\right) = H\left(z = e^{j\omega}\right). \tag{4.25}$$

Plot $|H\left(e^{j\omega}\right)|$. Is it periodic? If so, find the period. $H(e^{j\omega})$ is known as the *Discret Time Fourier Transform* (DTFT) of $h(n)$.
Solution: The following code plots Fig. 4.6.

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



#DTFT
def H(z):
        num = np.polyval([1,0,1],z**(-1))
        den = np.polyval([0.5,1],z**(-1))
        H = num/den
        return H
```

```
#Input and Output
omega = np.linspace(−3*np.pi,3*np.pi,1e2)

#subplots
plt.plot(omega, abs(H(np.exp(1j*omega))))
plt.title('Filter Frequency Response')
plt.xlabel('$\omega$')
plt.ylabel('$|H(e^{\jmath\omega})| $')
plt.grid()# minor

plt.show()
```

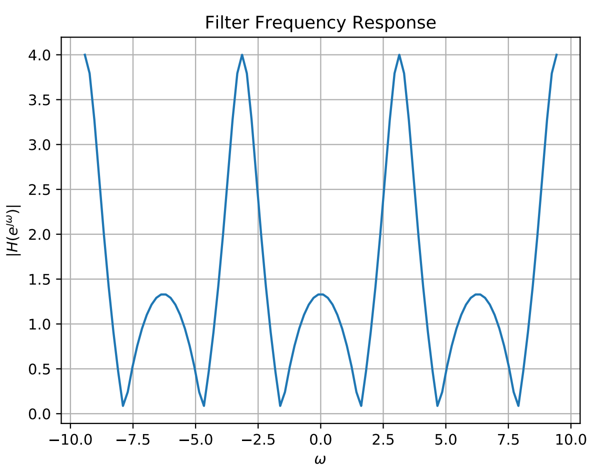


Fig. 4.6: $|H(e^{\jmath\omega})|$

Proving its periodic,

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1 + z^{-2}}{1 + \frac{1}{2}z^{-1}} \tag{4.26}$$

for

$$z = e^{jw} \tag{4.27}$$

$$H(z) = H(e^{jw}) = \frac{1 + e^{-2jw}}{1 + \frac{1}{2}e^{-jw}} \tag{4.28}$$

$$= \frac{1 + e^{2jw}}{e^{2jw} + \frac{1}{2}e^{jw}} \tag{4.29}$$

$$\implies \left|H(e^{jw})\right| = \left|\frac{1 + e^{2jw}}{e^{2jw} + \frac{1}{2}e^{jw}}\right| \tag{4.30}$$

$$= \frac{\left|1 + e^{2jw}\right|}{\left|e^{2jw} + \frac{1}{2}e^{jw}\right|} \tag{4.31}$$

$$= \frac{|1 + cos2w + 2jsinw| * 2}{|e^{jw} + 1|} \tag{4.32}$$

$$= \frac{|cosw + jsinw| * 4cosw}{|2cosw + 1 + 2jsinw|} \tag{4.33}$$

$$= \frac{4cosw}{\sqrt{(2cosw + 1)^2 + 4sin^2w}} \tag{4.34}$$

$$= \frac{4cosw}{\sqrt{5 + 4cosw}} \tag{4.35}$$

Therefore

$$H(e^{jw}) = \frac{4cosw}{\sqrt{5 + 4cosw}} \tag{4.36}$$

Consider

$$f(x) = \frac{4cosx}{\sqrt{5 + 4cosx}} \tag{4.37}$$

then

$$f(x + T) = \frac{4cos(x + T)}{\sqrt{5 + 4cos(x + T)}} \tag{4.38}$$

$$f(x + T) = \frac{4cos(x + T)}{\sqrt{5 + 4cos(x + T)}} \tag{4.39}$$

$$= \frac{4(cosxcosT - sinxsinT)}{\sqrt{5 + 4(cosxcosT - sinxsinT)}} \tag{4.40}$$

By comparing (4.37) and (4.38), we get cosT=1 and sinT=0

This is true for $t = 2k\pi$. This implies that the principal period of this function is 2π.

4.7 Express $h(n)$ in terms of $H(e^{j\omega})$.

$$H\left(e^{j\omega}\right) = \sum_{n=-\infty}^{\infty} h(n)\, e^{-jn\omega} \tag{4.41}$$

$$\implies \left(e^{j\omega}\right) * e^{jk\omega} = \sum_{n=-\infty}^{\infty} h(n)\, e^{-jn\omega} e^{jk\omega} \tag{4.42}$$

$$\implies \int_{-\pi}^{\pi} H\left(e^{j\omega}\right) * e^{jk\omega} d\omega = \int_{-\pi}^{\pi} \sum_{n=-\infty}^{\infty} h(n)\, e^{-jn\omega} e^{jk\omega} \tag{4.43}$$

$$\implies \int_{-\pi}^{\pi} H\left(e^{j\omega}\right) * e^{jk\omega} d\omega = \sum_{n=-\infty}^{\infty} h(n) \int_{-\pi}^{\pi} e^{-jn\omega} e^{jk\omega} \tag{4.44}$$

NOTE: We know that,

$$\int_{-\pi}^{\pi} e^{-jn\omega} e^{jk\omega} = \begin{cases} 2\pi & n = k \\ 0 & \text{otherwise} \end{cases} \tag{4.45}$$

Hence,

$$\int_{-\pi}^{\pi} H\left(e^{j\omega}\right) * e^{jk\omega} d\omega = \int_{-\pi}^{\pi} \sum_{n=-\infty}^{\infty} h(n)\, e^{-jn\omega} e^{jk\omega} \tag{4.46}$$

$$\implies \int_{-\pi}^{\pi} H\left(e^{j\omega}\right) * e^{jk\omega} d\omega = 2\pi h(n) \tag{4.47}$$

$$\implies \frac{1}{2\pi} \int_{-\pi}^{\pi} H\left(e^{j\omega}\right) * e^{jk\omega} d\omega = h(n) \tag{4.48}$$

Therefore,

$$h(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H\left(e^{j\omega}\right) e^{j\omega n} d\omega \tag{4.49}$$

5 Impulse Response

5.1 Using long division, find

$$h(n), \quad n < 5 \tag{5.1}$$

for H(z) in (4.12).
Solution:

$$H(z) = \frac{1 + z^{-2}}{1 + \frac{1}{2} z^{-1}} \tag{5.2}$$

Let $z^{-1} = x$,then, by polynomial long division we get

$$\implies (1 + z^{-2}) = (\frac{1}{2} z^{-1} + 1)(2z^{-1} - 4) + 5 \tag{5.3}$$

$$\implies \frac{(1 + z^{-2})}{\frac{1}{2} z^{-1} + 1} = (2z^{-1} - 4) + \frac{5}{\frac{1}{2} z^{-1} + 1} \tag{5.4}$$

$$\implies H(z) = (2z^{-1} - 4) + \frac{5}{\frac{1}{2} z^{-1} + 1} \tag{5.5}$$

Now, consider $\frac{5}{\frac{1}{2}z^{-1}+1}$
The denominator $\frac{1}{2}z^{-1} + 1$ can be expressed as sum of an infinite geometric progression, which as its first term equal to 1 and common ratio $\frac{-1}{2}z^{-1}$
Therefore, we can write $\frac{5}{\frac{1}{2}z^{-1}+1}$ as $5\left(1 + \left(\frac{-1}{2}z^{-1}\right) + \left(\frac{-1}{2}z^{-1}\right)^2 + \left(\frac{-1}{2}z^{-1}\right)^3 + \left(\frac{-1}{2}z^{-1}\right)^4 + \dots\right)$
Therefore, H(z) can be given by,

$$H(z) = (2z^{-1} - 4) + \frac{5}{\frac{1}{2} z^{-1} + 1} \tag{5.6}$$

$$\tag{5.7}$$

$$= 2z^{-1} - 4 + 5 + \frac{-5}{2} z^{-1} + \frac{5}{4} z^{-2} + \frac{-5}{8} z^{-3} + \frac{5}{16} z^{-4} + \dots \tag{5.8}$$

$$\implies H(z) = 1z^0 + \frac{-1}{2} z^{-1} + \frac{5}{4} z^{-2} + \frac{-5}{8} z^{-3} + \frac{5}{16} z^{-4} + \dots \tag{5.9}$$

Comparing the above expression to (4.1) we get h(n) for n<5 as,

$$h(0) = 1 \tag{5.10}$$

$$h(1) = \frac{-1}{2} \tag{5.11}$$

$$h(2) = \frac{5}{4} \tag{5.12}$$

$$h(3) = \frac{-5}{8} \tag{5.13}$$

$$h(4) = \frac{5}{16} \tag{5.14}$$

5.2 Find an expression for $h(n)$ using $H(z)$, given that

$$h(n) \overset{\mathcal{Z}}{\rightleftharpoons} H(z) \tag{5.15}$$

and there is a one to one relationship between $h(n)$ and $H(z)$. $h(n)$ is known as the *impulse response* of the system defined by (3.2).
Solution: From (4.12),

$$H(z) = \frac{1}{1+\frac{1}{2}z^{-1}} + \frac{z^{-2}}{1+\frac{1}{2}z^{-1}} \tag{5.16}$$

$$\implies h(n) = \left(-\frac{1}{2}\right)^n u(n) + \left(-\frac{1}{2}\right)^{n-2} u(n-2) \tag{5.17}$$

using (4.19) and (4.6).

5.3 Sketch $h(n)$. Is it bounded? Justify theoretically.
Solution: The following code plots Fig. 5.3.

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


n = np.arange(10)
fn=(−1/2)∗∗n
hn1=np.pad(fn, (0,2), 'constant',
    constant_values=(0))
hn2=np.pad(fn, (2,0), 'constant',
    constant_values=(0))
plt.stem(np.arange(12), hn1+hn2)
plt.title('Filter Impulse Response')
plt.xlabel('$n$')
plt.ylabel('$h(n)$')
plt.grid()# minor

plt.show()
```

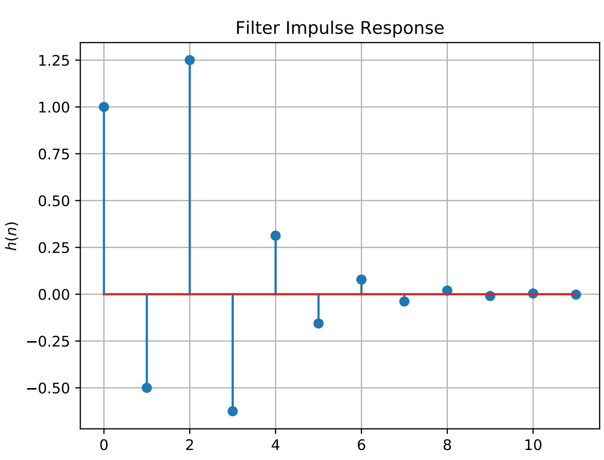


Fig. 5.3: $h(n)$ as the inverse of $H(z)$

From (5.17) we know that

$$h(n) = \left(-\frac{1}{2}\right)^n u(n) + \left(-\frac{1}{2}\right)^{n-2} u(n-2) \tag{5.18}$$

Implies we can write that

$$h\left(n\right) = \begin{cases} 0 & ,n<0 \\ \left(\frac{-1}{2}\right)^n & ,0 \le n < 2 \\ 5\left(\frac{-1}{2}\right)^n & ,n \ge 2 \end{cases} \tag{5.19}$$

A sequence is said to be bounded when

$$|x_n| \le M, \forall n \in \mathcal{N} \tag{5.20}$$

Now consider (5.19),
For n<0,

$$|h\left(n\right)| \le 0 \tag{5.21}$$

For $0 \le n < 2$,

$$|h\left(n\right)| = (\frac{1}{2})^n \tag{5.22}$$

$$\implies |h\left(n\right)| \le 1 \tag{5.23}$$

For $n \ge 2$,

$$|h\left(n\right)| = 5(\frac{1}{2})^n \tag{5.24}$$

$$\implies |h\left(n\right)| \le 5 \tag{5.25}$$

From above we can say that,

$$M = max\{0,1,5\} \tag{5.26}$$

$$= 5 \tag{5.27}$$

Therefore since M exists and is a real value, we can say that h(n) is bounded.

5.4 Convergent? Justify using the ratio test. **Solution:**
We can check if a sequence is convergent by ratio test. From the defination of ratio test we can say that a sequence is convergent if

$$\lim_{n\to\infty}\left|\frac{x_{n+1}}{x_n}\right| < 1 \tag{5.28}$$

Here, applying ratio test on (5.17) is same as applying ratio test on (5.19)

$$\lim_{n\to\infty}\left|\frac{h(n+1)}{h(n)}\right| = \lim_{n\to\infty}\left|\frac{5(\frac{-1}{2})^{n+1}}{5(\frac{-1}{2})^{n}}\right| \tag{5.29}$$

$$= \left|\frac{-1}{2}\right| \tag{5.30}$$

$$= \frac{1}{2} \tag{5.31}$$

From (5.31) we can clearly say that the sequence h(n) is convergent.

5.5 The system with $h(n)$ is defined to be stable if

$$\sum_{n=-\infty}^{\infty} h(n) < \infty \tag{5.32}$$

Is the system defined by (3.2) stable for the impulse response in (5.15)?

Solution:

From (5.17) we know that,

$$h(n) = \left(-\frac{1}{2}\right)^{n} u(n) + \left(-\frac{1}{2}\right)^{n-2} u(n-2) \tag{5.33}$$

Given that, a system is stable when

$$\sum_{n=-\infty}^{\infty} h(n) < \infty \tag{5.34}$$

$$\implies \sum_{n=-\infty}^{\infty} h(n) = \sum_{n=-\infty}^{\infty} (\left(-\frac{1}{2}\right)^{n} u(n) + \left(-\frac{1}{2}\right)^{n-2} u(n-2)) \tag{5.35}$$

$$= 2 * \sum_{n=-\infty}^{\infty} \left(-\frac{1}{2}\right)^{n} u(n) \tag{5.36}$$

$$\implies \sum_{n=-\infty}^{\infty} h(n) = 2 * (\frac{1}{1+\frac{1}{2}}) \tag{5.37}$$

$$= \frac{4}{3} < \infty \tag{5.38}$$

Hence, the system is stable.

5.6 Verify the above result using a python code.

```
def u(n):
    if n <0 :
        return 0.0
    else :
        return 1.0
def h(n):
    return u(n)*(-1.0/2)**n + u(n-2)*(-1.0/2)**(n-2)
sum = 0
for i in range(200000):
    sum +=h(i)
print(sum)
```

```
1.3333333333333333
```

5.7 Compute and sketch $h(n)$ using

$$h(n) + \frac{1}{2}h(n-1) = \delta(n) + \delta(n-2), \tag{5.39}$$

This is the definition of $h(n)$.

Solution: The following code plots Fig. 5.7. Note that this is the same as Fig. 5.3.

```
import numpy as np
import matplotlib.pyplot as plt
#If using termux
import subprocess
import shlex
#end if


k = 12
h = np.zeros(k)
h[0] = 1
h[1] = −0.5*h[0]
h[2] = −0.5*h[1] + 1

for n in range(3,k−1):
        h[n] = −0.5*h[n−1]

#subplots
plt.stem(range(0,k),h)
plt.title('Impulse Response Definition')
plt.xlabel('$n$')
plt.ylabel('$h(n)$')
plt.grid()# minor

#If using termux
plt.savefig('../figs/hndef.pdf')
subprocess.run(shlex.split("termux−open ../
    figs/hndef.pdf"))
#else
#plt.show()
```

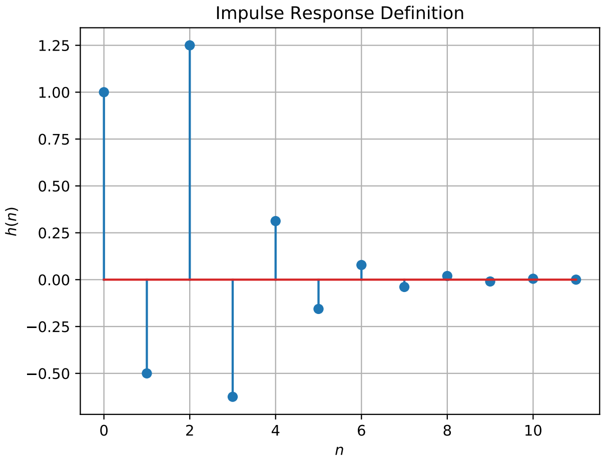


Fig. 5.7: $h(n)$ from the definition

5.8 Compute

$$y(n) = x(n) * h(n) = \sum_{n=-\infty}^{\infty} x(k)h(n-k) \quad (5.40)$$

Comment. The operation in (5.40) is known as *convolution*.
Solution: The following code plots Fig. 5.8. Note that this is the same as $y(n)$ in Fig. 3.2.

```
import numpy as np
import matplotlib.pyplot as plt
#If using termux
import subprocess
import shlex
#end if



n = np.arange(14)
fn=(−1/2)**n
hn1=np.pad(fn, (0,2), 'constant',
    constant_values=(0))
hn2=np.pad(fn, (2,0), 'constant',
    constant_values=(0))
h = hn1+hn2

nh=len(h)
x=np.array([1.0,2.0,3.0,4.0,2.0,1.0])
nx = len(x)

y = np.zeros(nx+nh−1)

for k in range(0,nx+nh−1):
        for n in range(0,nx):
                if k−n >= 0 and k−n < nh:
                        y[k]+=x[n]*h[k−n]

print(y)
#plots
plt.stem(range(0,nx+nh−1),y)
plt.title('Filter Output using Convolution')
plt.xlabel('$n$')
plt.ylabel('$y(n)$')
plt.grid()# minor

#If using termux
plt.savefig('../figs/ynconv.pdf')
plt.savefig('../figs/ynconv.eps')
subprocess.run(shlex.split("termux−open ../
    figs/ynconv.pdf"))
#else
#plt.show()
```

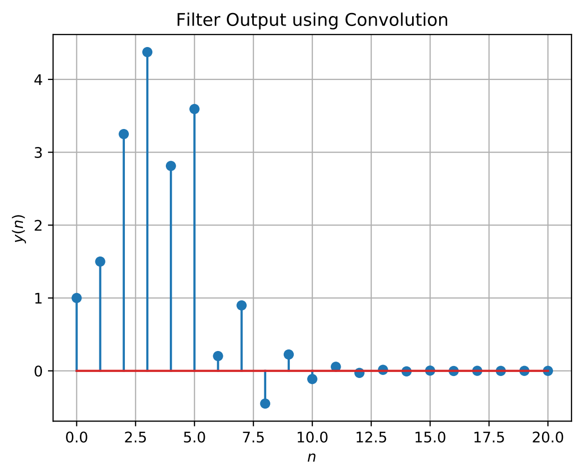


Fig. 5.8: $y(n)$ from the definition of convolution

5.9 Express the above convolution using a Teoplitz matrix.
Solution:

We know that from, (5.40),

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) \quad (5.41)$$

This can also be wrtten as a matrix-vector multiplication given by the expression,

$$y = T(h) * x \quad (5.42)$$

In the equation (5.42), $T(h)$ is a Teoplitz

matrix.
The equation (5.42) can be expanded as,

$$\mathbf{y} = \mathbf{x} \circledast \mathbf{h} \tag{5.43}$$

$$\mathbf{y} = \begin{pmatrix} h_1 & 0 & . & . & . & 0 \\ h_2 & h_1 & . & . & . & 0 \\ h_3 & h_2 & h_1 & . & . & 0 \\ . & . & . & . & . & . \\ h_{n-1} & h_{n-2} & h_{n-3} & . & . & 0 \\ h_n & h_{n-1} & h_{n-2} & . & . & h_1 \\ 0 & h_n & h_{n-1} & h_{n-2} & . & h_2 \\ . & . & . & . & . & . \\ 0 & . & . & . & 0 & h_{n-1} \\ 0 & . & . & . & 0 & h_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{pmatrix} \tag{5.44}$$

5.10 Show that

$$y(n) = \sum_{n=-\infty}^{\infty} x(n-k)h(k) \tag{5.45}$$

Solution:
From the defination of convolution given in (5.40),we know that,

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) \tag{5.46}$$

$$\implies y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) \tag{5.47}$$

Replace k with $n-k$. Now since n varies from n=$-\infty$ to n=∞, n-k will also vary from $-\infty$ to ∞. Therefore, we get,

$$y(n) = \sum_{n-k=-\infty}^{\infty} x(n-k)h(k) \tag{5.48}$$

$$= \sum_{k=-\infty}^{\infty} x(n-k)h(k) \tag{5.49}$$

6 DFT and FFT

6.1 Compute

$$X(k) \triangleq \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}, \quad k = 0, 1, \ldots, N-1 \tag{6.1}$$

and $H(k)$ using $h(n)$.
Solution:
From 3.1, we know that ,

$$x(n) = \left\{ \underset{\uparrow}{1}, 2, 3, 4, 2, 1 \right\} \tag{6.2}$$

Here, let, $\omega = e^{-j2\pi k}$. Then,

$$X(k) = 1 + 2\omega^{\frac{1}{5}} + 3\omega^{\frac{2}{5}} + 4\omega^{\frac{3}{5}} + 2\omega^{\frac{4}{5}} + \omega \tag{6.3}$$

Similarly, we know from (5.19),

$$h(n) = \begin{cases} 0 & , n < 0 \\ \left(\frac{-1}{2}\right)^n & , 0 \le n < 2 \\ 5\left(\frac{-1}{2}\right)^n & , n \ge 2 \end{cases} \tag{6.4}$$

Now,again let, $\omega = e^{-j2\pi k}$. Then,

$$H(k) = 1 + \frac{-1}{2}\omega^{\frac{1}{5}} + \frac{5}{4}\omega^{\frac{2}{5}} + \frac{-5}{8}\omega^{\frac{3}{5}} + \frac{5}{16}\omega^{\frac{4}{5}} + \frac{-5}{32}\omega \tag{6.5}$$

6.2 Compute

$$Y(k) = X(k)H(k) \tag{6.6}$$

Solution:
Now, from (6.3) and (6.5), we know $X(k) and H(k)$. Now, given that,

$$Y(k) = X(k) * H(k) \tag{6.7}$$

$$Y(k) = (1 + 2\omega^{\frac{1}{5}} + 3\omega^{\frac{2}{5}} + 4\omega^{\frac{3}{5}} + 2\omega^{\frac{4}{5}} + \omega) * \\ (1 + \frac{-1}{2}\omega^{\frac{1}{5}} + \frac{5}{4}\omega^{\frac{2}{5}} + \frac{-5}{8}\omega^{\frac{3}{5}} + \frac{5}{16}\omega^{\frac{4}{5}} + \frac{-5}{32}\omega) \tag{6.8}$$

$$Y(k) = 1 + \frac{3}{2}\omega^{\frac{1}{5}} + \frac{13}{4}\omega^{\frac{2}{5}} + \frac{35}{8}\omega^{\frac{3}{5}} + \frac{45}{16}\omega^{\frac{4}{5}} \\ \frac{115}{32}\omega^{\frac{5}{5}} + \frac{1}{8}\omega^{\frac{6}{5}} + \frac{25}{32}\omega^{\frac{7}{5}} - \frac{5}{8}\omega^{\frac{8}{5}} \\ - \frac{5}{32}\omega^{5} \tag{6.9}$$

where, $\omega = e^{-j2k\pi}$

6.3 Compute

$$y(n) = \frac{1}{N}\sum_{k=0}^{N-1} Y(k) \cdot e^{j2\pi kn/N}, \quad n = 0, 1, \ldots, N-1 \tag{6.10}$$

Solution: The following code plots Fig. 5.8. Note that this is the same as $y(n)$ in Fig. 3.2.

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


N = 14
n = np.arange(N)
fn=(−1/2)∗∗n
hn1=np.pad(fn, (0,2), 'constant',
     constant_values=(0))
hn2=np.pad(fn, (2,0), 'constant',
     constant_values=(0))
h = hn1+hn2

xtemp=np.array([1.0,2.0,3.0,4.0,2.0,1.0])
x=np.pad(xtemp, (0,8), 'constant',
     constant_values=(0))

X = np.zeros(N) + 1j∗np.zeros(N)
for k in range(0,N):
          for n in range(0,N):
                   X[k]+=x[n]∗np.exp(−1j∗2∗
                        np.pi∗n∗k/N)
H = np.zeros(N) + 1j∗np.zeros(N)
for k in range(0,N):
          for n in range(0,N):
                   H[k]+=h[n]∗np.exp(−1j∗2∗
                        np.pi∗n∗k/N)

Y = np.zeros(N) + 1j∗np.zeros(N)
for k in range(0,N):
          Y[k] = X[k]∗H[k]

y = np.zeros(N) + 1j∗np.zeros(N)
for k in range(0,N):
          for n in range(0,N):
                   y[k]+=Y[n]∗np.exp(1j∗2∗np
                        .pi∗n∗k/N)

#print(X)
y = np.real(y)/N
#plots
plt.stem(range(0,N),y)
plt.title('Filter Output using DFT')
plt.xlabel('$n$')
plt.ylabel('$y(n)$')
plt.grid()# minor
plt.show()
```

6.4 Repeat the previous exercise by computing $X(k), H(k)$ and $y(n)$ through FFT and IFFT.

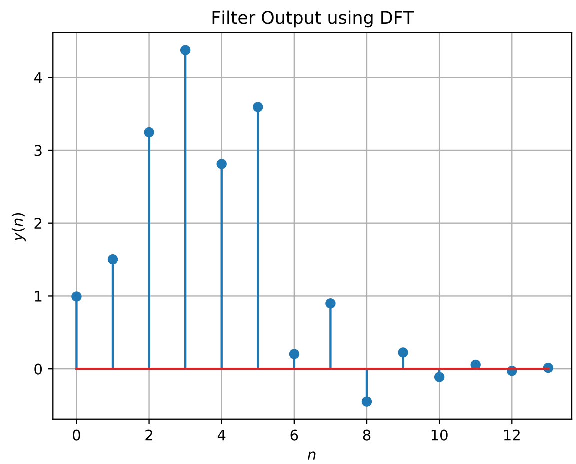


Fig. 6.3: $y(n)$ from the DFT

Solution:

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

N = 11

def x(n):
     if n < 0 or n > 5:
          return 0
     elif n < 4:
          return n + 1
     else:
          return 6 − n

def y(n):
     if n < 0:
          return 0
     else:
          return x(n) + x(n−2) − 0.5 ∗ y(n−1)

vec_y = scipy.vectorize(y, otypes=[float])

def delta(n):
     if n == 0:
          return 1
     else:
          return 0

def h(n):
     if n == 0:
```

```
        return 1
    elif n > 0:
        return delta(n) + delta(n−2) − 0.5∗h(
            n−1)
    else:
        return 2∗(delta(n+1) + delta(n−1) −
            h(n+1))

x_vec = scipy.vectorize(x, otypes=[float])
h_vec = scipy.vectorize(h, otypes=[float])

n_arr = np.linspace(0, N−1, N)
x_arr = x_vec(n_arr)
h_arr = h_vec(n_arr)

# FFT
X_arr = np.fft.fft(x_arr)
H_arr = np.fft.fft(h_arr)
Y_arr = X_arr ∗ H_arr
y_arr = np.fft.ifft(Y_arr)

# DFT
def DFT(k, inp):
    ksum = 0
    for n in range(N):
        ksum += inp(n) ∗ np.exp(−2j ∗ np.
            pi ∗ k ∗ n / N)
    return ksum

def Y(k):
    return DFT(k, x) ∗ DFT(k, h)

def IDFT(n, inp):
    nsum = 0
    for k in range(N):
        nsum += inp(k) ∗ np.exp(2j ∗ np.pi
            ∗ k ∗ n / N)
    return nsum / N

plt.stem(n_arr, vec_y(n_arr), markerfmt='
    bo', label='$y(n)$')
plt.stem(n_arr, np.real(IDFT(n_arr, Y)),
    markerfmt='go', label='$y_D(n)$')
plt.stem(n_arr, np.real(y_arr), markerfmt='ro
    ', label='$y_F(n)$')
plt.title('Filter Output using FFT')
plt.ylabel('$y(n)$')
plt.xlabel('$n$')
plt.grid()
plt.legend()
plt.show()
```

This code gives the below figure:

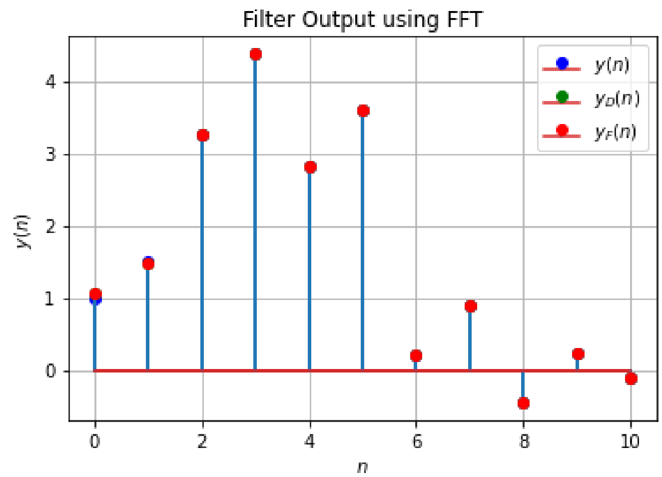


Fig. 6.4: $y(n)$ from different techniques

6.5 Wherever possible, express all the above equations as matrix equations.
Solution:
We use the DFT Matrix, where $\omega = e^{-\frac{j2k\pi}{N}}$, which is given by

$$\mathbf{W} = \begin{pmatrix} \omega^0 & \omega^0 & \dots & \omega^0 \\ \omega^0 & \omega^1 & \dots & \omega^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^0 & \omega^{N-1} & \dots & \omega^{(N-1)(N-1)} \end{pmatrix} \tag{6.11}$$

i.e. $W_{jk} = \omega^{jk}$, $0 \le j, k < N$. Hence, we can write any DFT equation as

$$\mathbf{X} = \mathbf{W}\mathbf{x} = \mathbf{x}\mathbf{W} \tag{6.12}$$

where

$$\mathbf{x} = \begin{pmatrix} x(0) \\ x(1) \\ \vdots \\ x(n-1) \end{pmatrix} \tag{6.13}$$

The inverse Fourier Transform is given by

$$\mathbf{x} = \mathcal{F}^{-1}(\mathbf{X}) = \mathbf{W}^{-1}\mathbf{X} = \frac{1}{N}\mathbf{W}^{\mathbf{H}}\mathbf{X} = \frac{1}{N}\mathbf{X}\mathbf{W}^{\mathbf{H}} \tag{6.14}$$

$$\implies \mathbf{W}^{-1} = \frac{1}{N}\mathbf{W}^{\mathbf{H}} \tag{6.15}$$

where H denotes hermitian operator. We can rewrite (**??**) using the element-wise multiplication operator as

$$\mathbf{Y} = \mathbf{H} \cdot \mathbf{X} = (\mathbf{Wh}) \cdot (\mathbf{Wx}) \tag{6.16}$$

6.6 Verify the above equations by generating the DFT matrix in python.
Solution:

```
import numpy as np
from numpy.fft import fft, ifft
import matplotlib.pyplot as plt


N = 14
n = np.arange(N)
fn=(−1/2)∗∗n
hn1=np.pad(fn, (0,2), 'constant',
    constant_values=(0,0))
hn2=np.pad(fn, (2,0), 'constant',
    constant_values=(0,0))
h = hn1+hn2
xtemp=np.array([1.0,2.0,3.0,4.0,2.0,1.0])
x=np.pad(xtemp, (0,10), 'constant',
    constant_values=(0))
dftmtx = fft(np.eye(len(x)))
X = x@dftmtx
H = h@dftmtx
Y = H∗X
invmtx = np.linalg.inv(dftmtx)
y = (Y@invmtx).real
#plots
plt.stem(range(0,16),y)
plt.title('$y(n)$ from DFT Matrix')
plt.xlabel('$n$')
plt.ylabel('$y(n)$')
plt.grid()
plt.show()
```

The code gives the following figure: We can see that the Fig. 6.6 is same as in Fig 3.2

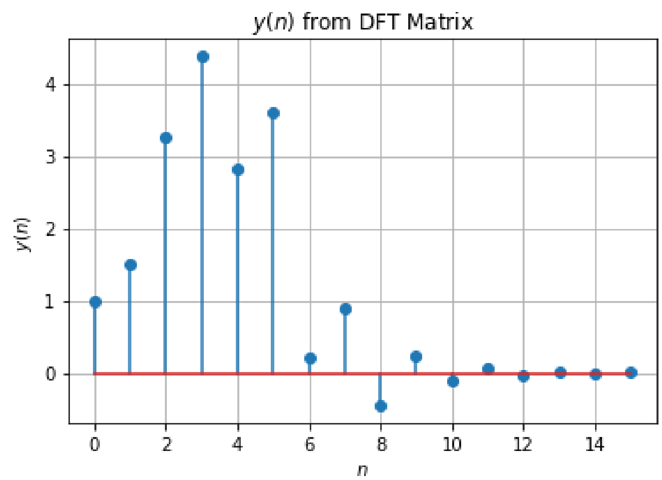


Fig. 6.6: $y(n)$ from DFT Matrix

7 FFT

1. The DFT of $x(n)$ is given by

$$X(k) \triangleq \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}, \quad k = 0, 1, \dots, N-1 \tag{7.1}$$

2. Let

$$W_N = e^{-j2\pi/N} \tag{7.2}$$

Then the N-point *DFT matrix* is defined as

$$\mathbf{F}_N = [W_N^{mn}], \quad 0 \le m, n \le N-1 \tag{7.3}$$

where W_N^{mn} are the elements of $\mathbf{F}_N$.

3. Let

$$\mathbf{I}_4 = \begin{pmatrix} \mathbf{e}_4^1 & \mathbf{e}_4^2 & \mathbf{e}_4^3 & \mathbf{e}_4^4 \end{pmatrix} \tag{7.4}$$

be the 4×4 identity matrix. Then the 4 point *DFT permutation matrix* is defined as

$$\mathbf{P}_4 = \begin{pmatrix} \mathbf{e}_4^1 & \mathbf{e}_4^3 & \mathbf{e}_4^2 & \mathbf{e}_4^4 \end{pmatrix} \tag{7.5}$$

4. The 4 point *DFT diagonal matrix* is defined as

$$\mathbf{D}_4 = diag\begin{pmatrix} W_4^0 & W_N^1 & W_N^2 & W_N^3 \end{pmatrix} \tag{7.6}$$

5. Show that

$$W_N^2 = W_{N/2} \tag{7.7}$$

Solution: We write

$$W_N = e^{-\frac{j2\pi}{N}} \tag{7.8}$$

$$W_N^2 = \left(e^{-\frac{j2\pi}{N}}\right)^2 = e^{-\frac{j2\pi}{N/2}} = W_{N/2} \tag{7.9}$$

6. Show that

$$\mathbf{F}_4 = \begin{bmatrix} \mathbf{I}_2 & \mathbf{D}_2 \\ \mathbf{I}_2 & -\mathbf{D}_2 \end{bmatrix} \begin{bmatrix} \mathbf{F}_2 & 0 \\ 0 & \mathbf{F}_2 \end{bmatrix} \mathbf{P}_4 \tag{7.10}$$

Solution: Observe that for $n \in \mathbb{N}$, $W_4^{4n} = 1$ and

$W_4^{4n+2} = -1$.

$$\mathbf{D}_2\mathbf{F}_2 = \begin{bmatrix} W_4^0 & 0 \\ 0 & W_4^1 \end{bmatrix}\begin{bmatrix} W_2^0 & W_2^0 \\ W_2^0 & W_2^1 \end{bmatrix} \tag{7.11}$$
$$= \begin{bmatrix} W_4^0 & 0 \\ 0 & W_4^1 \end{bmatrix}\begin{bmatrix} W_4^0 & W_4^0 \\ W_4^0 & W_4^2 \end{bmatrix} \tag{7.12}$$
$$= \begin{bmatrix} W_4^0 & W_4^0 \\ W_4^1 & W_4^3 \end{bmatrix} \tag{7.13}$$
$$\implies -\mathbf{D}_2\mathbf{F}_2 = \begin{bmatrix} W_4^2 & W_4^6 \\ W_4^3 & W_4^9 \end{bmatrix} \tag{7.14}$$

and

$$\mathbf{F}_2 = \begin{pmatrix} W_2^0 & W_2^0 \\ W_2^0 & W_2^1 \end{pmatrix} \tag{7.15}$$
$$= \begin{pmatrix} W_4^0 & W_4^0 \\ W_4^0 & W_4^2 \end{pmatrix} \tag{7.16}$$

Hence,

$$\mathbf{W}_4 = \begin{pmatrix} W_4^0 & W_4^0 & W_4^0 & W_4^0 \\ W_4^0 & W_4^2 & W_4^1 & W_4^3 \\ W_4^0 & W_4^4 & W_4^2 & W_4^6 \\ W_4^0 & W_4^6 & W_4^3 & W_4^9 \end{pmatrix} \tag{7.17}$$
$$= \begin{bmatrix} \mathbf{I}_2\mathbf{F}_2 & \mathbf{D}_2F_2 \\ \mathbf{I}_2\mathbf{F}_2 & -\mathbf{D}_2F_2 \end{bmatrix} \tag{7.18}$$
$$= \begin{bmatrix} \mathbf{I}_2 & \mathbf{D}_2 \\ \mathbf{I}_2 & \mathbf{D}_2 \end{bmatrix}\begin{bmatrix} \mathbf{F}_2 & 0 \\ 0 & \mathbf{F}_2 \end{bmatrix} \tag{7.19}$$

Multiplying (7.19) by $\mathbf{P}_4$ on both sides, and noting that $\mathbf{W}_4\mathbf{P}_4 = \mathbf{F}_4$ gives us.

7. Show that

$$\mathbf{F}_N = \begin{bmatrix} \mathbf{I}_{N/2} & \mathbf{D}_{N/2} \\ \mathbf{I}_{N/2} & -\mathbf{D}_{N/2} \end{bmatrix}\begin{bmatrix} \mathbf{F}_{N/2} & 0 \\ 0 & \mathbf{F}_{N/2} \end{bmatrix}\mathbf{P}_N \tag{7.20}$$

Solution: Observe that for even N and letting $\mathbf{f}_N^i$ denote the i^{th} column of $\mathbf{F}_N$, from (7.13) and (7.14),

$$\begin{pmatrix} \mathbf{D}_{N/2}\mathbf{F}_{N/2} \\ -\mathbf{D}_{N/2}\mathbf{F}_{N/2} \end{pmatrix} = \begin{pmatrix} \mathbf{f}_N^2 & \mathbf{f}_N^4 & \dots & \mathbf{f}_N^N \end{pmatrix} \tag{7.21}$$

and

$$\begin{pmatrix} \mathbf{I}_{N/2}\mathbf{F}_{N/2} \\ \mathbf{I}_{N/2}\mathbf{F}_{N/2} \end{pmatrix} = \begin{pmatrix} \mathbf{f}_N^1 & \mathbf{f}_N^3 & \dots & \mathbf{f}_N^{N-1} \end{pmatrix} \tag{7.22}$$

Thus,

$$\begin{bmatrix} \mathbf{I}_2\mathbf{F}_2 & \mathbf{D}_2\mathbf{F}_2 \\ \mathbf{I}_2\mathbf{F}_2 & -\mathbf{D}_2\mathbf{F}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{N/2} & \mathbf{D}_{N/2} \\ \mathbf{I}_{N/2} & -\mathbf{D}_{N/2} \end{bmatrix}\begin{bmatrix} \mathbf{F}_{N/2} & 0 \\ 0 & \mathbf{F}_{N/2} \end{bmatrix}$$
$$= \begin{pmatrix} \mathbf{f}_N^1 & \dots & \mathbf{f}_N^{N-1} & \mathbf{f}_N^2 & \dots & \mathbf{f}_N^N \end{pmatrix} \tag{7.23}$$

and so,

$$\begin{bmatrix} \mathbf{I}_{N/2} & \mathbf{D}_{N/2} \\ \mathbf{I}_{N/2} & -\mathbf{D}_{N/2} \end{bmatrix}\begin{bmatrix} \mathbf{F}_{N/2} & 0 \\ 0 & \mathbf{F}_{N/2} \end{bmatrix}\mathbf{P}_N$$
$$= \begin{pmatrix} \mathbf{f}_N^1 & \mathbf{f}_N^2 & \dots & \mathbf{f}_N^N \end{pmatrix} = \mathbf{F}_N \tag{7.24}$$

8. Find

$$\mathbf{P}_4\mathbf{x} \tag{7.25}$$

Solution: We have,

$$\mathbf{P}_4\mathbf{x} = \begin{pmatrix} \mathbf{e}_4^1 & \mathbf{e}_4^3 & \mathbf{e}_4^2 & \mathbf{e}_4^4 \end{pmatrix}\begin{pmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{pmatrix} = \begin{pmatrix} x(0) \\ x(2) \\ x(1) \\ x(3) \end{pmatrix} \tag{7.26}$$

9. Show that

$$\mathbf{X} = \mathbf{F}_N\mathbf{x} \tag{7.27}$$

where $\mathbf{x}, \mathbf{X}$ are the vector representations of $x(n), X(k)$ respectively.

Solution: Writing the terms of X,

$$X(0) = x(0) + x(1) + \dots + x(N-1) \tag{7.28}$$
$$X(1) = x(0) + x(1)e^{-\frac{\jmath 2\pi}{N}} + \dots + \\ + x(N-1)e^{-\frac{\jmath 2(N-1)\pi}{N}} \tag{7.29}$$
$$\vdots$$
$$X(N-1) = x(0) + x(1)e^{-\frac{\jmath 2(N-1)\pi}{N}} + \dots + \\ + x(N-1)e^{-\frac{\jmath 2(N-1)(N-1)\pi}{N}} \tag{7.30}$$

Clearly, the term in the m^{th} row and n^{th} column is given by ($0 \le m \le N-1$ and $0 \le n \le N-1$)

$$T_{mn} = x(n)e^{-\frac{\jmath 2mn\pi}{N}} \tag{7.31}$$

and so, we can represent each of these terms as a matrix product

$$\mathbf{X} = \mathbf{F}_N\mathbf{x} \tag{7.32}$$

where $\mathbf{F}_N = \left[e^{-\frac{-\jmath 2mn\pi}{N}}\right]_{mn}$ for $0 \le m \le N-1$ and $0 \le n \le N-1$.

10. Derive the following Step-by-step visualisation

of 8-point FFTs into 4-point FFTs and so on

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} X_1(0) \\ X_1(1) \\ X_1(2) \\ X_1(3) \end{bmatrix} + \begin{bmatrix} W_8^0 & 0 & 0 & 0 \\ 0 & W_8^1 & 0 & 0 \\ 0 & 0 & W_8^2 & 0 \\ 0 & 0 & 0 & W_8^3 \end{bmatrix} \begin{bmatrix} X_2(0) \\ X_2(1) \\ X_2(2) \\ X_2(3) \end{bmatrix} \tag{7.33}$$

$$\begin{bmatrix} X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} = \begin{bmatrix} X_1(0) \\ X_1(1) \\ X_1(2) \\ X_1(3) \end{bmatrix} - \begin{bmatrix} W_8^0 & 0 & 0 & 0 \\ 0 & W_8^1 & 0 & 0 \\ 0 & 0 & W_8^2 & 0 \\ 0 & 0 & 0 & W_8^3 \end{bmatrix} \begin{bmatrix} X_2(0) \\ X_2(1) \\ X_2(2) \\ X_2(3) \end{bmatrix} \tag{7.34}$$

4-point FFTs into 2-point FFTs

$$\begin{bmatrix} X_1(0) \\ X_1(1) \end{bmatrix} = \begin{bmatrix} X_3(0) \\ X_3(1) \end{bmatrix} + \begin{bmatrix} W_4^0 & 0 \\ 0 & W_4^1 \end{bmatrix} \begin{bmatrix} X_4(0) \\ X_4(1) \end{bmatrix} \tag{7.35}$$

$$\begin{bmatrix} X_1(2) \\ X_1(3) \end{bmatrix} = \begin{bmatrix} X_3(0) \\ X_3(1) \end{bmatrix} - \begin{bmatrix} W_4^0 & 0 \\ 0 & W_4^1 \end{bmatrix} \begin{bmatrix} X_4(0) \\ X_4(1) \end{bmatrix} \tag{7.36}$$

$$\begin{bmatrix} X_2(0) \\ X_2(1) \end{bmatrix} = \begin{bmatrix} X_5(0) \\ X_5(1) \end{bmatrix} + \begin{bmatrix} W_4^0 & 0 \\ 0 & W_4^1 \end{bmatrix} \begin{bmatrix} X_6(0) \\ X_6(1) \end{bmatrix} \tag{7.37}$$

$$\begin{bmatrix} X_2(2) \\ X_2(3) \end{bmatrix} = \begin{bmatrix} X_5(0) \\ X_5(1) \end{bmatrix} - \begin{bmatrix} W_4^0 & 0 \\ 0 & W_4^1 \end{bmatrix} \begin{bmatrix} X_6(0) \\ X_6(1) \end{bmatrix} \tag{7.38}$$

$$P_8 \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \\ x(7) \end{bmatrix} = \begin{bmatrix} x(0) \\ x(2) \\ x(4) \\ x(6) \\ x(1) \\ x(3) \\ x(5) \\ x(7) \end{bmatrix} \tag{7.39}$$

$$P_4 \begin{bmatrix} x(0) \\ x(2) \\ x(4) \\ x(6) \end{bmatrix} = \begin{bmatrix} x(0) \\ x(4) \\ x(2) \\ x(6) \end{bmatrix} \tag{7.40}$$

$$P_4 \begin{bmatrix} x(1) \\ x(3) \\ x(5) \\ x(7) \end{bmatrix} = \begin{bmatrix} x(1) \\ x(5) \\ x(3) \\ x(7) \end{bmatrix} \tag{7.41}$$

Therefore,

$$\begin{bmatrix} X_3(0) \\ X_3(1) \end{bmatrix} = F_2 \begin{bmatrix} x(0) \\ x(4) \end{bmatrix} \tag{7.42}$$

$$\begin{bmatrix} X_4(0) \\ X_4(1) \end{bmatrix} = F_2 \begin{bmatrix} x(2) \\ x(6) \end{bmatrix} \tag{7.43}$$

$$\begin{bmatrix} X_5(0) \\ X_5(1) \end{bmatrix} = F_2 \begin{bmatrix} x(1) \\ x(5) \end{bmatrix} \tag{7.44}$$

$$\begin{bmatrix} X_6(0) \\ X_6(1) \end{bmatrix} = F_2 \begin{bmatrix} x(3) \\ x(7) \end{bmatrix} \tag{7.45}$$

Solution: We write out the values of performing an 8-point FFT on $\mathbf{x}$ as follows.

$$X(k) = \sum_{n=0}^{7} x(n) e^{-\frac{j2kn\pi}{8}} \tag{7.46}$$

$$= \sum_{n=0}^{3} \left(x(2n) e^{-\frac{j2kn\pi}{4}} + e^{-\frac{j2k\pi}{8}} x(2n+1) e^{-\frac{j2kn\pi}{4}} \right) \tag{7.47}$$

$$= X_1(k) + e^{-\frac{j2k\pi}{4}} X_2(k) \tag{7.48}$$

where $\mathbf{X}_1$ is the 4-point FFT of the even-numbered terms and $\mathbf{X}_2$ is the 4-point FFT of the odd numbered terms. Noticing that for $k \geq 4$,

$$X_1(k) = X_1(k-4) \tag{7.49}$$

$$e^{-\frac{j2k\pi}{8}} = -e^{-\frac{j2(k-4)\pi}{8}} \tag{7.50}$$

we can now write out $X(k)$ in matrix form as in (**??**) and (**??**). We also need to solve the two 4-point FFT terms so formed.

$$X_1(k) = \sum_{n=0}^{3} x_1(n) e^{-\frac{j2kn\pi}{8}} \tag{7.51}$$

$$= \sum_{n=0}^{1} \left(x_1(2n) e^{-\frac{j2kn\pi}{4}} + e^{-\frac{j2k\pi}{8}} x_2(2n+1) e^{-\frac{j2kn\pi}{4}} \right) \tag{7.52}$$

$$= X_3(k) + e^{-\frac{j2k\pi}{4}} X_4(k) \tag{7.53}$$

using $x_1(n) = x(2n)$ and $x_2(n) = x(2n+1)$. Thus we can write the 2-point FFTs

$$\begin{bmatrix} X_3(0) \\ X_3(1) \end{bmatrix} = F_2 \begin{bmatrix} x(0) \\ x(4) \end{bmatrix} \tag{7.54}$$

$$\begin{bmatrix} X_4(0) \\ X_4(1) \end{bmatrix} = F_2 \begin{bmatrix} x(2) \\ x(6) \end{bmatrix} \tag{7.55}$$

Using a similar idea for the terms X_2,

$$\begin{bmatrix} X_5(0) \\ X_5(1) \end{bmatrix} = F_2 \begin{bmatrix} x(1) \\ x(5) \end{bmatrix} \tag{7.56}$$

$$\begin{bmatrix} X_6(0) \\ X_6(1) \end{bmatrix} = F_2 \begin{bmatrix} x(3) \\ x(7) \end{bmatrix} \tag{7.57}$$

But observe that from (7.26),

$$\mathbf{P}_8\mathbf{x} = \begin{pmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{pmatrix} \tag{7.58}$$

$$\mathbf{P}_4\mathbf{x}_1 = \begin{pmatrix}\mathbf{x}_3\\ \mathbf{x}_4\end{pmatrix} \tag{7.59}$$

$$\mathbf{P}_4\mathbf{x}_2 = \begin{pmatrix}\mathbf{x}_5\\ \mathbf{x}_6\end{pmatrix} \tag{7.60}$$

where we define $x_3(k) = x(4k)$, $x_4(k) = x(4k+2)$, $x_5(k) = x(4k+1)$, and $x_6(k) = x(4k+3)$ for $k = 0, 1$.

11. For

$$\mathbf{x} = \begin{pmatrix}1\\2\\3\\4\\2\\1\end{pmatrix} \tag{7.61}$$

compte the DFT using (7.27)

Solution:

The code below gives the answer

```
import numpy as np
from numpy.fft import fft, ifft
import matplotlib.pyplot as plt
#If using termux
#import subprocess
#import shlex
#end if

x=np.array([1.0,2.0,3.0,4.0,2.0,1.0])
dftmtx = fft(np.eye(len(x)))
X = x@dftmtx
print(X)
```

12. Repeat the above exercise using the FFT after zero padding **x**.
13. Write a C program to compute the 8-point FFT.

Solution:

```
#include <stdio.h>
#include <stdbool.h>
#include <math.h>
#include <stdlib.h>
#include <complex.h>
#include <time.h>
#define EPS 1e-6

complex *myfft(int n, complex *a)
{
        if (n == 1) return a;
        complex *g = (complex *)malloc(n
            /2*sizeof(complex));
        complex *h = (complex *)malloc(n
            /2*sizeof(complex));
        for (int i = 0; i < n; i++)
    {
                if (i%2) h[i/2] = a[i];
                else g[i/2] = a[i];
        }
        g = myfft(n/2, g);
        h = myfft(n/2, h);
        for (int i = 0; i < n; i++) a[i] = g[i
            %(n/2)] + cexp(-I*2*M_PI*i/n)
            *h[i%(n/2)];
        free(g); free(h);
        return a;
}

int main()
{
        int n = 8;
        complex *a = (complex *)malloc(
            sizeof(complex)*n);
        a[0] = 1.0, a[1] = 2.0, a[2] = 3.0, a
            [3] = 4.0, a[4] = 2.0, a[5] = 1.0,
             a[6] = 0.0, a[7] = 0.0;
        a = myfft(n, a);
        for (int i = 0; i < n; i++) printf("X
            (%d) = %lf + %lfj\n", i, creal(a
            [i]), cimag(a[i]));
        free(a);
        return 0;

}
```

8 Exercises

Answer the following questions by looking at the python code in Problem 2.3.

8.1 The command

```
        output_signal = signal.lfilter(b, a,
            input_signal)
```

in Problem 2.3 is executed through the following difference equation

$$\sum_{m=0}^{M} a(m)\, y(n-m) = \sum_{k=0}^{N} b(k)\, x(n-k) \quad (8.1)$$

where the input signal is $x(n)$ and the output signal is $y(n)$ with initial values all 0. Replace **signal.filtfilt** with your own routine and verify.
Solution:

```
import soundfile as sf
from scipy import signal, fft
import numpy as np
from numpy.polynomial import Polynomial
    as P
from matplotlib import pyplot as plt

def myfiltfilt(b, a, input_signal):
    X = fft.fft(input_signal)
    w = np.linspace(0, 1, len(X) + 1)
    W = np.exp(2j*np.pi*w[:-1])
    B = (np.absolute(np.polyval(b,W)))**2
    A = (np.absolute(np.polyval(a,W)))**2
    Y = B*(1/A)*X
    return fft.ifft(Y).real

#read .wav file
input_signal,fs = sf.read('Sound_Noise.wav
    ')

#sampling frequency of Input signal
sampl_freq=fs

#order of the filter
order=4

#cutoff frquency 4kHz
cutoff_freq=4000.0

#digital frequency
Wn=2*cutoff_freq/sampl_freq

# b and a are numerator and denominator
    polynomials respectively
b, a = signal.butter(order, Wn, 'low')

#filter the input signal with butterworth filter
output_signal = signal.filtfilt(b, a,
    input_signal)
#output_signal1 = signal.lfilter(b, a,
    input_signal)
os1 = myfiltfilt(b, a, input_signal)
x_plt = np.arange(len(input_signal))
#Verify outputs by plotting
plt.plot(x_plt[:100], output_signal[:100], 'b
    .')
plt.plot(x_plt[:100], os1[:100], 'r.')
plt.grid()
plt.show()
```

8.2 Repeat all the exercises in the previous sections for the above a and b.
Solution: For the given values, the difference equation is

$$\begin{aligned}
&y(n) - (4.44)\, y(n-1) + (8.78)\, y(n-2)\\
&- (9.93)\, y(n-3) + (6.90)\, y(n-4)\\
&- (2.93)\, y(n-5) + (0.70)\, y(n-6)\\
&- (0.07)\, y(n-7) = \left(5.02 \times 10^{-5}\right) x(n)\\
&+ \left(3.52 \times 10^{-4}\right) x(n-1) + \left(1.05 \times 10^{-3}\right) x(n-2)\\
&+ \left(1.76 \times 10^{-3}\right) x(n-3) + \left(1.76 \times 10^{-3}\right) x(n-4)\\
&+ \left(1.05 \times 10^{-3}\right) x(n-5) + \left(3.52 \times 10^{-4}\right) x(n-6)\\
&+ \left(5.02 \times 10^{-5}\right) x(n-7) \qquad (8.2)
\end{aligned}$$

From (8.1), we see that the transfer function can be written as follows

$$H(z) = \frac{\sum_{k=0}^{N} b(k) z^{-k}}{\sum_{k=0}^{M} a(k) z^{-k}} \quad (8.3)$$

$$= \sum_{i} \frac{r(i)}{1 - p(i) z^{-1}} + \sum_{j} k(j) z^{-j} \quad (8.4)$$

where $r(i)$, $p(i)$, are called residues and poles respectively of the partial fraction expansion of $H(z)$. $k(i)$ are the coefficients of the direct polynomial terms that might be left over. We can now take the inverse z-transform of (8.4) and get using (4.19),

$$h(n) = \sum_{i} r(i)[p(i)]^{n} u(n) + \sum_{j} k(j)\delta(n-j) \quad (8.5)$$

Substituting the values,

$$
\begin{aligned}
h(n) = &[(2.76)\,(0.55)^n \\
&+ (-1.05 - 1.84\jmath)\,(0.57 + 0.16\jmath)^n \\
&+ (-1.05 + 1.84\jmath)\,(0.57 - 0.16\jmath)^n \\
&+ (-0.53 + 0.08\jmath)\,(0.63 + 0.32\jmath)^n \\
&+ (-0.53 - 0.08\jmath)\,(0.63 - 0.32\jmath)^n \\
&+ (0.20 + 0.004\jmath)\,(0.75 + 0.47\jmath)^n \\
&+ (0.20 - 0.004\jmath)\,(0.75 - 0.47\jmath)^n]u(n) \\
&+ \left(-6.81 \times 10^{-4}\right)\delta(n)
\end{aligned} \tag{8.6}
$$

The values $r(i)$, $p(i)$, $k(i)$ and thus the impulse response function are computed and plotted at

```
import soundfile as sf
import matplotlib.pyplot as plt
from scipy import signal
from scipy import vectorize as vec
import numpy as np

#read .wav file
input_signal,fs = sf.read('/home/saqib/iith/
    courseWork/sem5/EE3900/filter/codes/
    Sound_Noise.wav')

#sampling frequency of Input signal
sampl_freq=fs

#order of the filter
order=7

#cutoff frquency 4kHz
cutoff_freq=4000.0

#digital frequency
Wn=2*cutoff_freq/sampl_freq

# b and a are numerator and denominator
    polynomials respectively
b, a = signal.butter(order, Wn, 'low')

# get partial fraction expansion
r, p, k = signal.residuez(b, a)

#number of terms of the impulse response
sz = 50
sz_lin = np.arange(sz)

def rp(x):
    return r@(p**x).T

rp_vec = vec(rp, otypes=['double'])

h1 = rp_vec(sz_lin)
k_add = np.pad(k, (0, sz − len(k)), 'constant
    ', constant_values=(0,0))
h = h1 + k_add
plt.stem(sz_lin, h)
plt.xlabel('n')
plt.ylabel('h(n)')
plt.grid()
plt.plot()
plt.show()
```

The filter frequency response is plotted at

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
import soundfile as sf

input_signal,fs = sf.read('/home/saqib/iith/
    courseWork/sem5/EE3900/filter/codes/
    Sound_Noise.wav')
sampl_freq=fs
order=4
cutoff_freq=4000.0

Wn=2*cutoff_freq/sampl_freq

b, a = signal.butter(order, Wn, 'low')


def H(z):
        num = np.polyval(b,z**(−1))
        den = np.polyval(a,z**(−1))
        H = num/den
        return H



omega = np.linspace(0,np.pi,100)


plt.plot(omega, abs(H(np.exp(1j*omega))))
plt.xlabel('$\omega$')
plt.ylabel('$|H(e^{\jmath\omega})| $')
plt.grid()


plt.show()
```

Observe that for a series $t_n = r^n$, $\frac{t_{n+1}}{t_n} = r$. By the ratio test, t_n converges if $|r| < 1$. We note that observe that $|p(i)| < 1$ and so, as $h(n)$ is the sum of convergent series, we see that $h(n)$ converges. From Fig. (8.2), it is clear that $h(n)$ is bounded. From (4.1),

$$\sum_{n=0}^{\infty} h(n) = H(1) = 1 < \infty \tag{8.7}$$

Therefore, the system is stable. From $h(n)$ is negligible after $n \geq 64$, and we can apply a 64-bit FFT to get y(n). The following code uses the DFT matrix to generate $y(n)$.

```
import soundfile as sf
import matplotlib.pyplot as plt
from scipy import signal
from scipy import vectorize as vec
import numpy as np


input_signal,fs = sf.read('codes/
     Sound_Noise.wav')

sampl_freq=fs

order=7

cutoff_freq=4000.0

Wn=2*cutoff_freq/sampl_freq

# b and a are numerator and denominator
     polynomials respectively
b, a = signal.butter(order, Wn, 'low')
output_signal = signal.filtfilt(b, a,
     input_signal)

# get partial fraction expansion
r, p, k = signal.residuez(b, a)
#number of terms of the impulse response
sz = 64
sz_lin = np.arange(sz)

dftmtx = np.fft.fft(np.eye(sz))
invmtx = np.linalg.inv(dftmtx)
def rp(x):
     return r@(p**x).T
rp_vec = vec(rp, otypes=['double'])

h1 = rp_vec(sz_lin)
k_add = np.pad(k, (0, sz - len(k)), 'constant
     ', constant_values=(0,0))
h = h1 + k_add
H = h@dftmtx
X = input_signal[:sz]@dftmtx
Y = H*X
y = (Y@invmtx).real
plt.stem(np.arange(sz), y[:sz])
plt.xlabel('n')
plt.ylabel('y(n)')
plt.grid()
plt.plot()
plt.show()
```

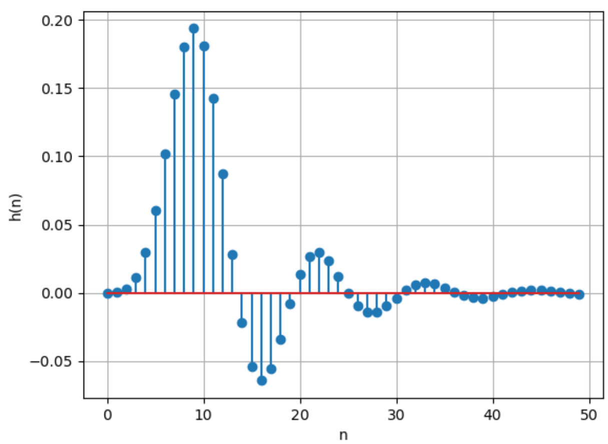


Fig. 8.2: Plot of $h(n)$

8.3 What is the sampling frequency of the input signal?
Solution:
Sampling frequency(fs)=44.1kHZ.

8.4 What is type, order and cutoff-frequency of the above butterworth filter
Solution:
The given butterworth filter is low pass with order=2 and cutoff-frequency=4kHz.

8.5 Modifying the code with different input parameters and to get the best possible output.
Solution:
A better filtering was found on setting the order of the filter to be 7.

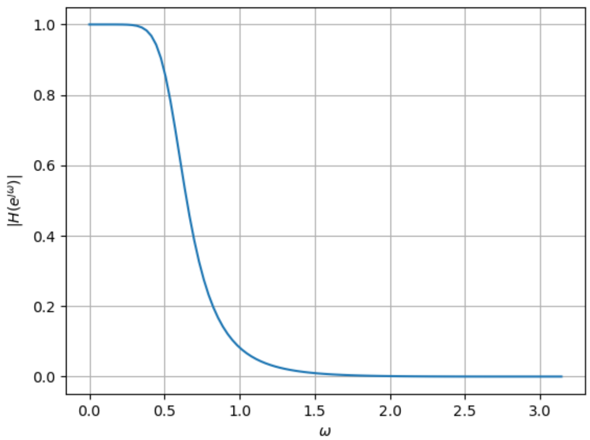


Fig. 8.2: Filter frequency response

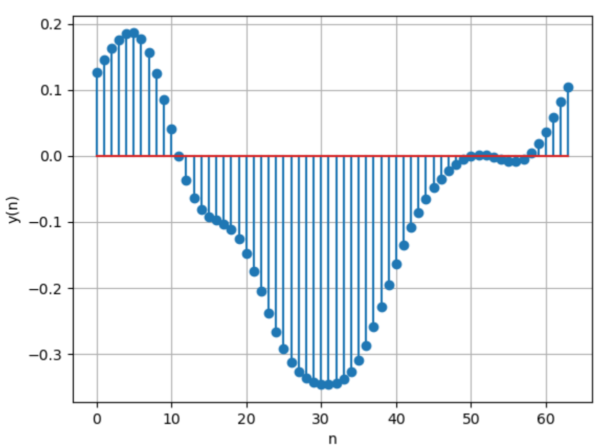


Fig. 8.2: Plot of $y(n)$