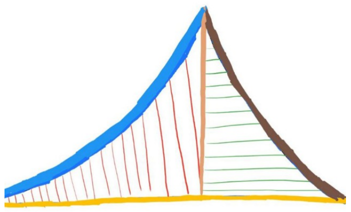
SIGNAL PROCESSING

Through Practice

G. V. V. Sharma



Contents

Introduction

This book introduces digital communication through probability.

Chapter 1

Two Dice

1.1. Problem

Two dice, one blue and one grey, are thrown at the same time. The event defined by the sum of the two numbers appearing on the top of the dice can have 11 possible outcomes 2, 3, 4, 5, 6, 6, 8, 9, 10, 11 and 12. A student argues that each of these outcomes has a probability $\frac{1}{11}$. Do you agree with this argument? Justify your answer.

1.2. Uniform Distribution: Rectangular Function

1.2.1. Let $X_i \in \{1,2,3,4,5,6\}, i = 1,2$, be the random variables representing the outcome for each die. Assuming the dice to be fair, the probability mass function (pmf) is expressed as

$$p_{X_i}(n) = \Pr(X_i = n) = \begin{cases} \frac{1}{6} & 1 \le n \le 6 \\ 0 & otherwise \end{cases} \tag{1.2.1.1}$$

1.3. Sum of Random Variables: Convolution

1.3.1. The desired outcome is

$$X = X_1 + X_2, \tag{1.3.1.1}$$

$$\implies X \in \{1, 2, \dots, 12\} \tag{1.3.1.2}$$

The objective is to show that

$$p_X(n) \neq \frac{1}{11} \tag{1.3.1.3}$$

1.3.2. <u>Convolution:</u> From (1.3.1.1),

$$p_X(n) = \Pr\left(X_1 + X_2 = n\right) = \Pr\left(X_1 = n - X_2\right) \tag{1.3.2.1}$$

$$= \sum_k \Pr\left(X_1 = n - k | X_2 = k\right) p_{X_2}(k) \tag{1.3.2.2}$$

after unconditioning. $\because$ X_1 and X_2 are independent,

$$\Pr\left(X_1 = n - k | X_2 = k\right) = \Pr\left(X_1 = n - k\right) = p_{X_1}(n - k) \tag{1.3.2.3}$$

From (1.3.2.2) and (1.3.2.3),

$$p_X(n) = \sum_k p_{X_1}(n - k) p_{X_2}(k) \triangleq p_{X_1}(n) * p_{X_2}(n) \tag{1.3.2.4}$$

where $*$ denotes the convolution operation.

1.3.3. Substituting from (1.2.1.1) in (1.3.2.4),

$$p_X(n) = \frac{1}{6}\sum_{k=1}^{6} p_{X_1}(n-k) = \frac{1}{6}\sum_{k=n-6}^{n-1} p_{X_1}(k) \tag{1.3.3.1}$$

$$\because p_{X_1}(k) = 0, \quad k \le 1, k \ge 6. \tag{1.3.3.2}$$

From (1.3.3.1),

$$p_X(n) = \begin{cases} 0 & n < 1 \\ \frac{1}{6}\sum_{k=1}^{n-1} p_{X_1}(k) & 1 \le n-1 \le 6 \\ \frac{1}{6}\sum_{k=n-6}^{6} p_{X_1}(k) & 1 < n-6 \le 6 \\ 0 & n > 12 \end{cases} \tag{1.3.3.3}$$

1.4. The Triangular function

1.4.1. Substituting from (1.2.1.1) in (1.3.3.3),

$$p_X(n) = \begin{cases} 0 & n < 1 \\ \frac{n-1}{36} & 2 \le n \le 7 \\ \frac{13-n}{36} & 7 < n \le 12 \\ 0 & n > 12 \end{cases} \tag{1.4.1.1}$$

satisfying (1.3.1.3).

1.5. The Z-transform

1.5.1. The Z-transform of $p_X(n)$ is defined as

$$P_X(z) = \sum_{n=-\infty}^{\infty} p_X(n) z^{-n}, \quad z \in \mathbb{C} \tag{1.5.1.1}$$

From (1.2.1.1) and (1.5.1.1),

$$P_{X_1}(z) = P_{X_2}(z) = \frac{1}{6}\sum_{n=1}^{6} z^{-n} \tag{1.5.1.2}$$

$$= \frac{z^{-1}\left(1 - z^{-6}\right)}{6\left(1 - z^{-1}\right)}, \quad |z| > 1 \tag{1.5.1.3}$$

upon summing up the geometric progression.

1.5.2. Show that

$$p_X(n) = p_{X_1}(n) * p_{X_2}(n) \implies P_X(z) = P_{X_1}(z) P_{X_2}(z) \tag{1.5.2.1}$$

The above property follows from Fourier analysis and is fundamental to signal processing.

1.5.3. The unit step function is defined as

$$u(n) = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases} \tag{1.5.3.1}$$

Show that the Z transform of $u(n)$ is

$$U(z) = \frac{1}{1 - z^{-1}}, \; |z| > 1 \tag{1.5.3.2}$$

1.5.4. Show that

$$nu(n) \overset{\mathcal{Z}}{\longleftrightarrow} \frac{z^{-1}}{\left(1-z^{-1}\right)^2}, \ |z| > 1 \tag{1.5.4.1}$$

1.5.5. Show that

$$p_X(n-k) \overset{\mathcal{Z}}{\longleftrightarrow} P_X(z)z^{-k} \tag{1.5.5.1}$$

1.5.6. From (1.5.1.3) and (1.5.2.1),

$$P_X(z) = \left\{\frac{z^{-1}\left(1-z^{-6}\right)}{6\left(1-z^{-1}\right)}\right\}^2 = \frac{1}{36}\frac{z^{-2}\left(1-2z^{-6}+z^{-12}\right)}{\left(1-z^{-1}\right)^2} \tag{1.5.6.1}$$

From (1.5.5.1) and (1.5.4.1), it can be shown that

$$\frac{1}{36}\left[(n-1)\,u(n-1) - 2\,(n-7)\,u(n-7) + (n-13)\,u(n-13)\right] \overset{\mathcal{Z}}{\longleftrightarrow} \frac{1}{36}\frac{z^{-2}\left(1-2z^{-6}+z^{-12}\right)}{\left(1-z^{-1}\right)^2} \tag{1.5.6.2}$$

1.5.7. From (1.5.1.1), (1.5.6.1) and (1.5.6.2)

$$p_X(n) = \frac{1}{36}\left[(n-1)\,u(n-1) - 2\,(n-7)\,u(n-7) + (n-13)\,u(n-13)\right] \tag{1.5.7.1}$$

which is the same as (1.4.1.1). Note that (1.4.1.1) can be obtained from (1.5.6.2) using contour integration as well.

1.5.8. The experiment of rolling the dice was simulated using Python for 10000 samples. These were generated using Python libraries for uniform distribution. The frequencies for each outcome were then used to compute the resulting pmf, which is plotted in

Figure 1.5.8.1. The theoretical pmf obtained in (1.4.1.1) is plotted for comparison.

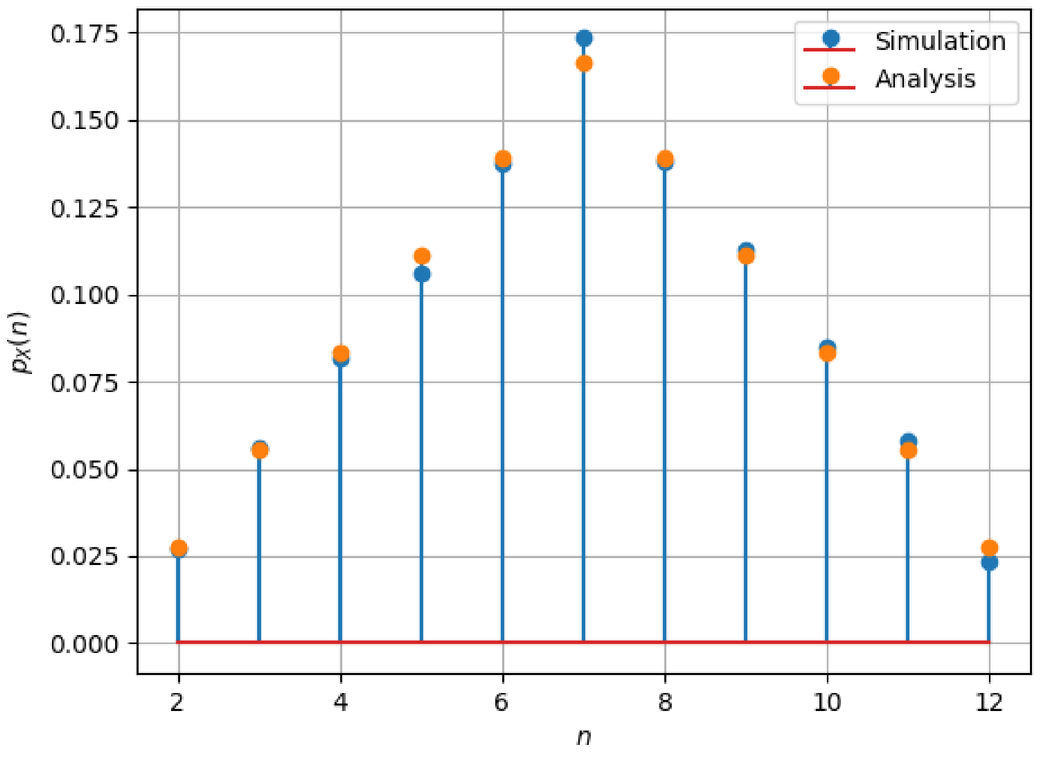


Figure 1.5.8.1: Plot of $p_X(n)$. Simulations are close to the analysis.

1.5.9. The python code is available in

```
/codes/sum/dice.py
```

Chapter 2

Pingala Series

2.1. JEE 2019

$$a_n = \frac{\alpha^n - \beta^n}{\alpha - \beta}, \quad n \ge 1 \tag{9.1}$$

$$b_n = a_{n-1} + a_{n+1}, \quad n \ge 2, \quad b_1 = 1 \tag{9.2}$$

where α and β ($\alpha > \beta$) are the roots of the

$$z^2 - z - 1 = 0 \tag{9.3}$$

Verify the following using a python code.

2.1.1

$$\sum_{k=1}^{n} a_k = a_{n+2} - 1, \quad n \ge 1 \tag{2.1.1.1}$$

2.1.2

$$\sum_{k=1}^{\infty} \frac{a_k}{10^k} = \frac{10}{89} \tag{2.1.2.1}$$

2.1.3

$$b_n = \alpha^n + \beta^n, \quad n \geq 1 \tag{2.1.3.1}$$

2.1.4

$$\sum_{k=1}^{\infty} \frac{b_k}{10^k} = \frac{8}{89} \tag{2.1.4.1}$$

Solution:

```
$ python3 pingala/codes/1.py
```

2.2. Pingala Series

2.2.1 The Pingala series is generated using the difference equation

$$x(n+2) = x(n+1) + x(n), \quad x(0) = x(1) = 1, n \geq 0 \tag{2.2.1.1}$$

Generate a stem plot for $x(n)$.

Solution: The following code generates Fig. 2.2.1.1.

```
$ python3 pingala/codes/2_1.py
```

2.2.2 The one sided Z-transform of $x(n)$ is defined as

$$X^+(z) = \sum_{n=0}^{\infty} x(n) z^{-n}, \quad z \in \mathbb{C} \tag{2.2.2.1}$$

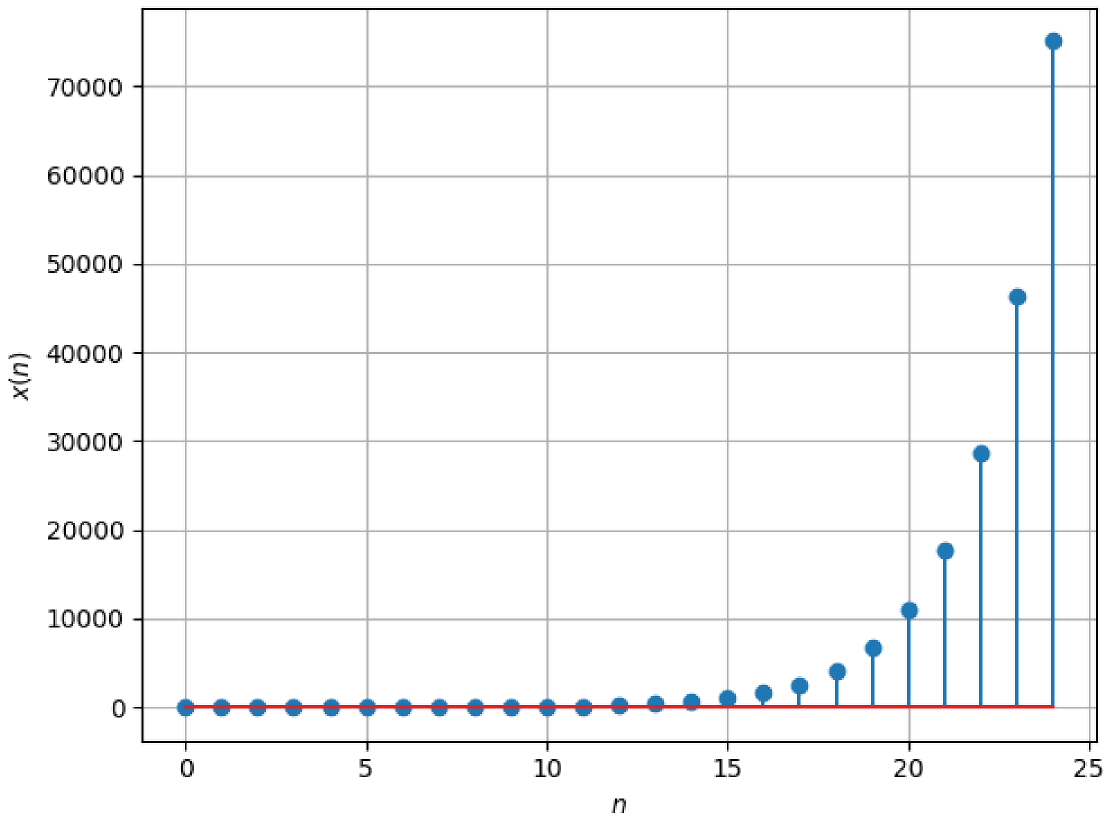


Figure 2.2.1.1: Plot of $x(n)$

Find $X^+(z)$.

Solution: Taking the one-sided Z-transform on both sides of (2.2.1.1),

$$\mathcal{Z}^+\left[x(n+2)\right] = \mathcal{Z}^+\left[x(n+1)\right] + \mathcal{Z}^+\left[x(n)\right] \tag{2.2.2.2}$$

$$\implies z^2X^+(z) - z^2x(0) - zx(1) = zX^+(z) - zx(0) + zX^+(z) \tag{2.2.2.3}$$

$$\implies \left(z^2 - z - 1\right)X^+(z) = z^2 \tag{2.2.2.4}$$

$$\implies X^+(z) = \frac{1}{1 - z^{-1} - z^{-2}} = \frac{1}{\left(1-\alpha z^{-1}\right)\left(1-\beta z^{-1}\right)}, \quad |z| > \alpha \tag{2.2.2.5}$$

2.2.3 Find $x(n)$.

Solution: Expanding $X^+(z)$ in (2.2.2.5) using partial fractions, we get

$$X^+(z) = \frac{1}{(\alpha-\beta)}\left[\frac{z}{1-\alpha z^{-1}} - \frac{z}{1-\beta z^{-1}}\right] \tag{2.2.3.1}$$

$$\implies x(n) = \frac{\alpha^{n+1} - \beta^{n+1}}{\alpha - \beta}u(n) \tag{2.2.3.2}$$

$$= a_{n+1} \tag{2.2.3.3}$$

upon comparing with (9.1).

2.3. Linear Time Invariant System

2.3.1 Sketch

$$y(n) = x\left(n-1\right) + x\left(n+1\right), \quad n \geq 0 \tag{2.3.1.1}$$

Solution: Execute

```
$ python3 pingala/codes/2_2.py
```

to obtain Fig. 2.3.1.1

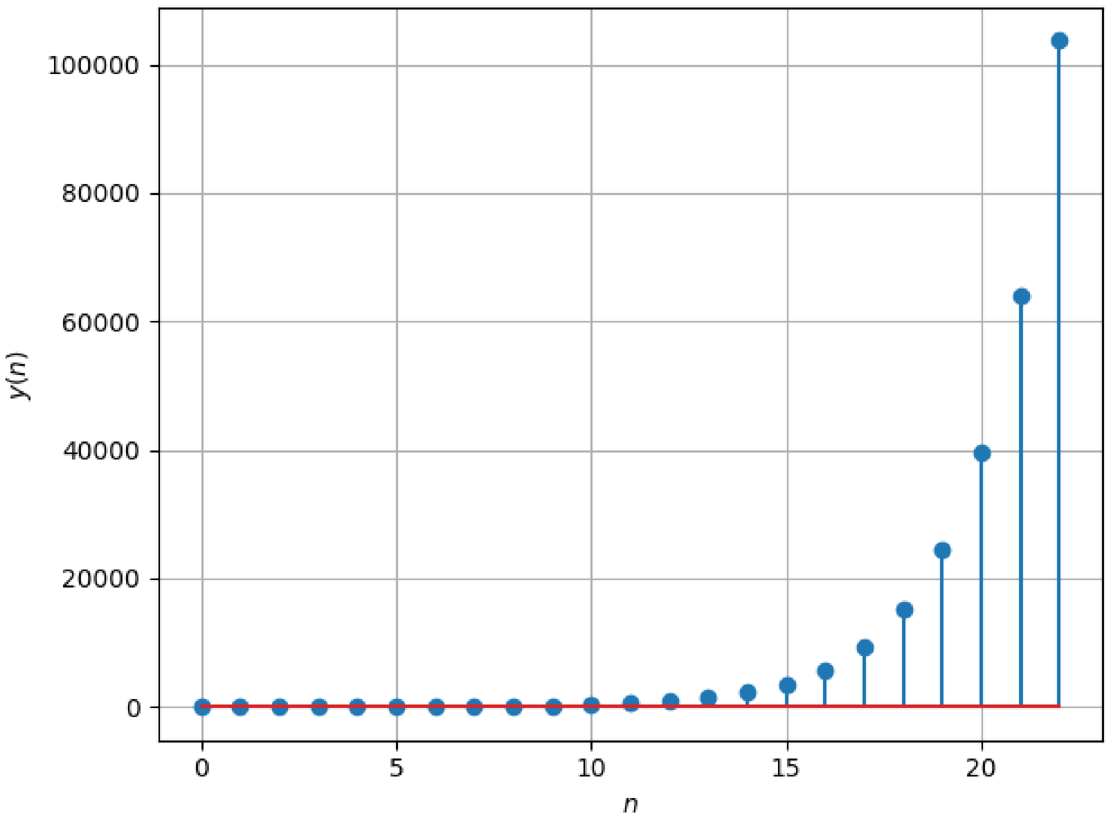


Figure 2.3.1.1: Plot of $y(n)$

2.3.2 Show that

$$x(n+1) \overset{\mathcal{Z}}{\longleftrightarrow} zX^+(z) - zx(0) \tag{2.3.2.1}$$

$$x(n-1) \overset{\mathcal{Z}}{\longleftrightarrow} z^{-1}X^+(z) + zx(-1) \tag{2.3.2.2}$$

2.3.3 Find $Y^+(z)$.

Solution: Taking the one-sided Z-transform on both sides of (2.3.1.1),

$$\mathcal{Z}^{+}\left[y(n)\right] = \mathcal{Z}^{+}\left[x(n+1)\right] + \mathcal{Z}^{+}\left[x(n-1)\right] \tag{2.3.3.1}$$

$$Y^{+}(z) = zX^{+}(z) - zx(0) + z^{-1}X^{+}(z) + zx(-1) \tag{2.3.3.2}$$

$$= \frac{z+z^{-1}}{1-z^{-1}-z^{-2}} - z = \frac{1+2z^{-1}}{1-z^{-1}-z^{-2}}, \quad |z| > \alpha \tag{2.3.3.3}$$

2.3.4 Show that

$$y(n) = b_{n+1}. \tag{2.3.4.1}$$

2.3.5 Find the impulse response of (2.3.1.1)

2.4. Power of the Z transform

2.4.1 Show that

$$\sum_{k=1}^{\infty} \frac{a_k}{10^k} = \frac{1}{10}\sum_{k=0}^{\infty} \frac{x\left(k\right)}{10^k} = \frac{1}{10}X^{+}\left(10\right) \tag{2.4.1.1}$$

Solution:

$$\sum_{k=1}^{\infty} \frac{a_k}{10^k} = \frac{1}{10}\sum_{k=0}^{\infty} \frac{a_{k+1}}{10^k} = \frac{1}{10}\sum_{k=0}^{\infty} \frac{x(k)}{10^k} \tag{2.4.1.2}$$

$$= \frac{1}{10}X^{+}(10) = \frac{1}{10} \times \frac{100}{89} = \frac{10}{89} \tag{2.4.1.3}$$

Thus, (2.1.2.1) is correct.

2.4.2 Show that

$$\sum_{k=1}^{\infty} \frac{b_k}{10^k} = \frac{1}{10} \sum_{k=0}^{\infty} \frac{y\left(k\right)}{10^k} = \frac{1}{10} Y^+\left(10\right) \tag{2.4.2.1}$$

Solution:

$$\sum_{k=1}^{\infty} \frac{b_k}{10^k} = \frac{1}{10} \sum_{k=0}^{\infty} \frac{b_{k+1}}{10^k} = \frac{1}{10} \sum_{k=0}^{\infty} \frac{y(k)}{10^k} \tag{2.4.2.2}$$

$$= \frac{1}{10} Y^+(z) = \frac{1}{10} \times \frac{120}{89} = \frac{12}{89} \tag{2.4.2.3}$$

Thus, (2.1.4.1) is incorrect.

2.4.3 Show that

$$\alpha^n + \beta^n, \quad n \geq 1 \tag{2.4.3.1}$$

can be expressed as

$$w(n) = \left(\alpha^{n+1} + \beta^{n+1}\right) u(n) \tag{2.4.3.2}$$

and find $W(z)$.

Solution: Putting $n = k + 1$ in (2.4.3.1) and using the definition of $u(n)$,

$$\alpha^n + \beta^n = \left(\alpha^{k+1} + \beta^{k+1}\right) u(k) \tag{2.4.3.3}$$

Hence, (2.4.3.1) can be expressed as

$$w(n) = \left(\alpha^{n+1} + \beta^{n+1}\right) u(n) \tag{2.4.3.4}$$

Therefore,

$$W(z) = Y(z) = \frac{1 + 2z^{-1}}{1 - z^{-1} - z^{-2}} \tag{2.4.3.5}$$

Thus, by invoking (2.3.4.1), we find that (2.1.3.1) is correct

2.5. Convolution

2.5.1 Show that

$$\sum_{k=1}^{n} a_k = \sum_{k=0}^{n-1} x(k) = x(n) * u(n-1) \tag{2.5.1.1}$$

Solution: From (2.2.3.3), and noting that $x(n) = 0 \ \forall \ n < 0$,

$$\sum_{k=1}^{n} a_k = \sum_{k=0}^{n-1} x(k) = \sum_{k=-\infty}^{n-1} x(k) \tag{2.5.1.2}$$

$$= \sum_{k=-\infty}^{\infty} x(k)u(n-1-k) = x(n) * u(n-1) \tag{2.5.1.3}$$

2.5.2 Show that

$$a_{n+2} - 1, \quad n \geq 1 \tag{2.5.2.1}$$

can be expressed as

$$\left[x\left(n+1\right) - 1\right] u\left(n-1\right) \tag{2.5.2.2}$$

Solution: From (2.2.3.3),

$$a_{n+2} - 1 = \left[x(n+1) - 1\right], \quad n \geq 1 \tag{2.5.2.3}$$

and so, using the definition of $u(n)$,

$$a_{n+2} - 1 = \left[x(n+1) - 1\right] u(n-1) \tag{2.5.2.4}$$

2.5.3 Show that

$$\left[x(n+1) - 1\right] u(n-1) \overset{\mathcal{Z}}{\longleftrightarrow} \frac{z^{-1}}{\left(1 - z^{-1} - z^{-2}\right)\left(1 - z^{-1}\right)} \tag{2.5.3.1}$$

Solution: The Z transform of the above signal can be expressed as

$$\begin{aligned}\sum_{n=1}^{\infty} x(n+1)z^{-n} - \frac{z^{-1}}{1 - z^{-1}} &= \sum_{n=2}^{\infty} x(n)z^{-n+1} - \frac{z^{-1}}{1 - z^{-1}} && (2.5.3.2)\\ &= z\left[X^{+}(z) - x(0) - x(1)z^{-1}\right] - \frac{z^{-1}}{1 - z^{-1}} && (2.5.3.3)\\ &= \frac{z}{1 - z^{-1} - z^{-2}} - z - 1 - \frac{z^{-1}}{1 - z^{-1}} && (2.5.3.4)\\ &= \frac{z}{1 - z^{-1} - z^{-2}} - \frac{z}{1 - z^{-1}} && (2.5.3.5)\\ &= \frac{z^{-1}}{\left(1 - z^{-1} - z^{-2}\right)\left(1 - z^{-1}\right)} && (2.5.3.6)\end{aligned}$$

From (2.2.3.3), we get

$$\sum_{k=1}^{n} a_k = a_{n+2} - 1 \tag{2.5.3.7}$$

Chapter 3

Circuits and Transforms

3.1. Definitions

1. The unit step function is defined as

$$u(t) = \begin{cases} 1 & t > 0 \\ \frac{1}{2} & t = 0 \\ 0 & t < 0 \end{cases} \tag{3.1.1.1}$$

2. The Laplace transform of $g(t)$ is defined as

$$G(s) = \int_{-\infty}^{\infty} g(t)e^{-st}\, dt \tag{3.1.2.1}$$

3.2. Laplace Transform

In the circuit in Fig. 2.1, the switch S is connected to position P for a long time so that the charge on the capacitor becomes $q_1\,\mu C$. Then S is switched to position Q. After a long time, the charge on the capacitor is $q_2\,\mu C$. Use variables R_1, R_2, C_0 for the passive elements.

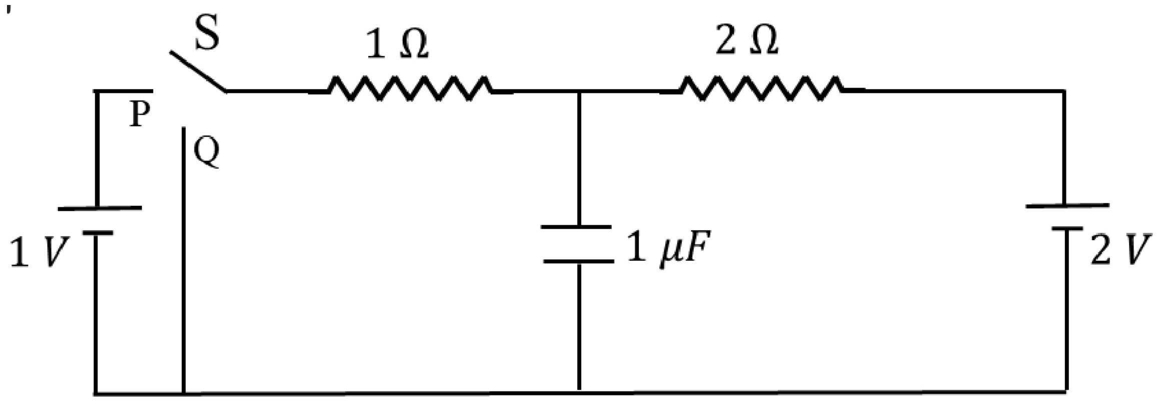


Figure 2.1:

1. Draw the circuit in position P.

   Solution: See Fig. 3.2.1.1 drawn using the code in

   ```
   cktsig/figs/ckt−q1.tex
   ```

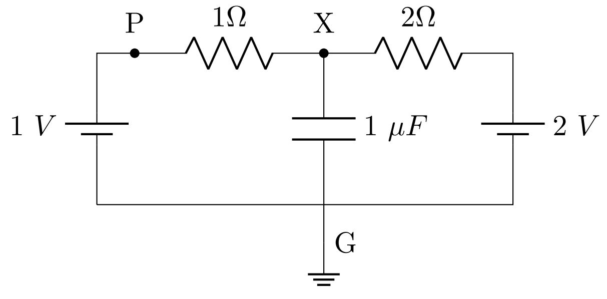


Figure 3.2.1.1:

2. Simulate the circuit in Fig. 3.2.1.1 using ngspice.

   Solution: The following ngspice script simulates the given circuit.

   ```
   ngspice codes/2_7.cir
   ```

3. Find q_1 in Fig. 3.2.3.1.

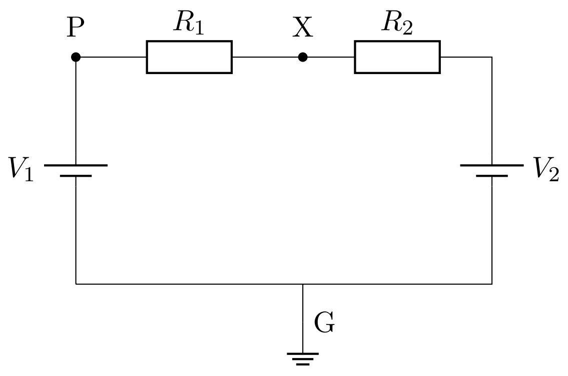


Figure 3.2.3.1:

Solution: Assuming the circuit to be grounded at G and the relative potential at point X to be V, we use KCL at X and get

$$\frac{V - V_1}{R_1} + \frac{V - V_2}{R_2} = 0 \tag{3.2.3.1}$$

$$\tag{3.2.3.2}$$

$$\implies V = \frac{V_1 R_2 + V_2 R_1}{R_1 + R_2} = \frac{4}{3}\,\text{V} \tag{3.2.3.3}$$

upon substituting numerical values. Hence,

$$q_1 = CV = C\frac{(V_1 R_2 + V_2 R_1)}{R_1 + R_2} = \frac{4}{3}\,\mu\text{C} \tag{3.2.3.4}$$

4. Show that the Laplace transform of $u(t)$ is $\frac{1}{s}$ and find the ROC.

Solution: We have,

$$u(t) \overset{\mathcal{L}}{\longleftrightarrow} \int_0^\infty u(t)e^{-st}dt \tag{3.2.4.1}$$

$$= \int_0^0 \frac{1}{2}e^{-st}dt + \int_0^\infty e^{-st}dt \tag{3.2.4.2}$$

$$= \frac{1}{s}, \quad \Re(s) > 0 \tag{3.2.4.3}$$

5. Now consider the following resistive circuit in Fig. 3.2.5.1 transformed from Fig. 2.1 where

$$v_1(t) = u(t) \overset{\mathcal{L}}{\longleftrightarrow} V_1(s) = \frac{1}{s} \tag{3.2.5.1}$$

$$v_2(t) = 2u(t) \overset{\mathcal{L}}{\longleftrightarrow} V_2(s) = \frac{2}{s} \tag{3.2.5.2}$$

Find the voltage across the capacitor $V_{C_0}(s)$.

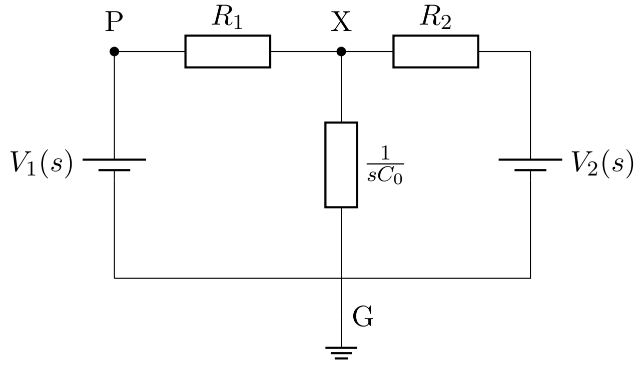


Figure 3.2.5.1:

Solution: Applying KCL at X,

$$\frac{V(s)-\frac{1}{s}}{R_1}+\frac{V(s)-\frac{2}{s}}{R_2}+sC_0V(s)=0 \tag{3.2.5.3}$$

$$\implies V(s)\left(\frac{1}{R_1}+\frac{1}{R_2}+sC_0\right)=\frac{1}{s}\left(\frac{1}{R_1}+\frac{2}{R_2}\right) \tag{3.2.5.4}$$

$$\implies V(s)=\frac{\frac{1}{R_1}+\frac{2}{R_2}}{s\left(\frac{1}{R_1}+\frac{1}{R_2}+sC_0\right)} \tag{3.2.5.5}$$

$$=\frac{\frac{1}{R_1}+\frac{2}{R_2}}{\frac{1}{R_1}+\frac{1}{R_2}}\left(\frac{1}{s}-\frac{1}{\frac{1}{C_0}\left(\frac{1}{R_1}+\frac{1}{R_2}\right)+s}\right) \tag{3.2.5.6}$$

6. Show that

$$e^{-at}u(t)\xleftrightarrow{\mathcal{L}}\frac{1}{s+a},\quad a>0 \tag{3.2.6.1}$$

and find the ROC.

Solution: Note that by substituting $s:=s+a$ in (3.2.4.3), and considering $a\in\mathbb{R}$,

$$e^{-at}u(t)\xleftrightarrow{\mathcal{L}}\int_0^\infty u(t)e^{-(s+a)t}dt \tag{3.2.6.2}$$

$$=\frac{1}{s+a},\quad \Re(s)>-a \tag{3.2.6.3}$$

7. Find $v_{C_0}(t)$.

Solution: Taking the inverse Laplace transform in (3.2.5.6),

$$V(s)\xleftrightarrow{\mathcal{L}^{-1}}\frac{2R_1+R_2}{R_1+R_2}u(t)\left(1-e^{-\left(\frac{1}{R_1}+\frac{1}{R_2}\right)\frac{t}{C_0}}\right) \tag{3.2.7.1}$$

from (3.2.6.1).

8. Verify your result numerically.

Solution: The following python code plots the graph in Fig. 3.2.8.1

```
python3 codes/2_6−sim.py
```

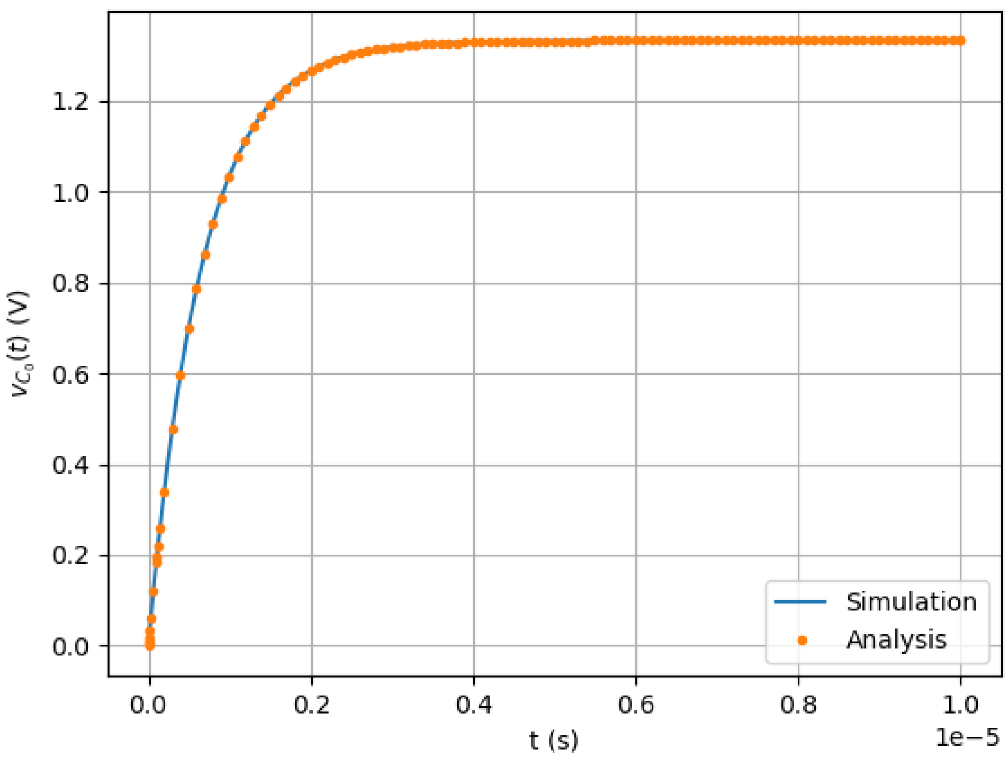


Figure 3.2.8.1: $v_{C_0}(t)$ simulation

3.3. Initial Conditions

1. Draw the equivalent circuit when the switch is at Q.

   Solution: See Fig. 3.3.1.1. below.

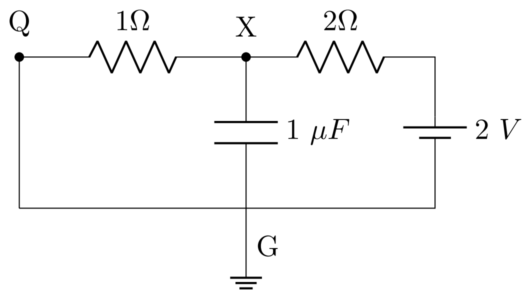


Figure 3.3.1.1:

2. Find q_2 in Fig. 3.3.2.1.

   Solution: The equivalent circuit at steady state when the switch is at Q is shown in Fig. 3.3.2.1. Since the capacitor behaves as an open circuit, we use voltage division at

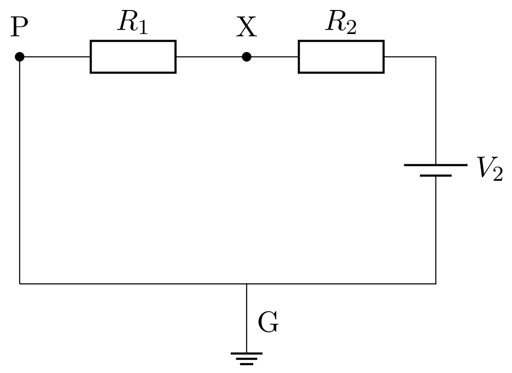


Figure 3.3.2.1:

   X to obtain

$$V = V_1 \frac{R_1}{R_1 + R_2} = \frac{2}{3}\,\text{V} \tag{3.3.2.1}$$

   upon substituting numerical values. Hence,

$$q_2 = CV = \frac{2}{3}\,\mu\text{C}. \tag{3.3.2.2}$$

3. Draw the equivalent s-domain resistive circuit when S is switched to position Q.

   See Fig. 3.3.3.1.

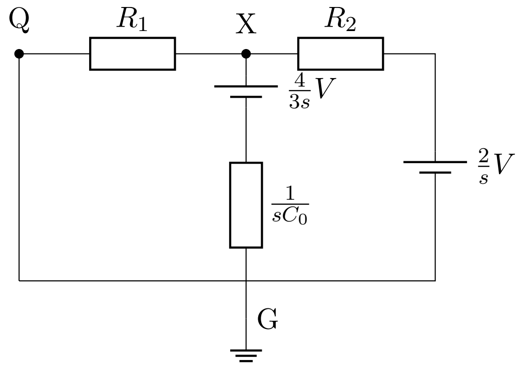


Figure 3.3.3.1:

4. $V_{C_0}(s) = ?$

   Solution: Using KCL at node X in Fig. 3.3.3.1,

$$\frac{V(s)-0}{R_1}+\frac{V(s)-\frac{2}{s}}{R_2}+sC_0\left(V(s)-\frac{4}{3s}\right)=0 \tag{3.3.4.1}$$

$$\implies V_{C_0}(s)=\frac{\frac{2}{sR_2}+\frac{4C_0}{3}}{\frac{1}{R_1}+\frac{2}{R_2}+sC_0} \tag{3.3.4.2}$$

5. $v_{C_0}(t) = ?$

   Solution: From (3.3.4.2), using partial fractions,

$$V_{C_0}(s)=\frac{4}{3}\left(\frac{1}{\frac{1}{C_0}\left(\frac{1}{R_1}+\frac{1}{R_2}\right)+s}\right)+\frac{2}{R_2\left(\frac{1}{R_1}+\frac{1}{R_2}\right)}\left(\frac{1}{s}-\frac{1}{\frac{1}{C_0}\left(\frac{1}{R_1}+\frac{1}{R_2}\right)+s}\right) \tag{3.3.5.1}$$

Taking the inverse Laplace Transform,

$$v_{C_0}(t) = \frac{4}{3}e^{-\left(\frac{1}{R_1}+\frac{1}{R_2}\right)\frac{t}{C_0}}u(t) + \frac{2}{R_2\left(\frac{1}{R_1}+\frac{1}{R_2}\right)}\left(1 - e^{-\left(\frac{1}{R_1}+\frac{1}{R_2}\right)\frac{t}{C_0}}\right)u(t) \tag{3.3.5.2}$$

from (3.2.6.1).

6. Verify your result numerically.

   Solution: Substituting numerical values,

$$v_{C_0}(t) = \frac{2}{3}\left(1 + e^{-\left(1.5\times10^6\right)t}\right)u(t) \tag{3.3.6.1}$$

   This is verified by the following code that plots Fig. 3.3.6.1 using ngspice and python.

```
python3 codes/3_4−sim.py
```

7. Find $v_{C_0}(0-), v_{C_0}(0+)$ and $v_{C_0}(\infty)$.

   Solution: From the initial conditions,

$$v_{C_0}(0-) = \frac{q_1}{C_0} = \frac{4}{3}\,\mathrm{V} \tag{3.3.7.1}$$

   Using (3.3.6.1),

$$v_{C_0}(0+) = \lim_{t\to0+} v_{C_0}(t) = \frac{4}{3}\,\mathrm{V} \tag{3.3.7.2}$$

$$v_{C_0}(\infty) = \lim_{t\to\infty} v_{C_0}(t) = \frac{2}{3}\,\mathrm{V} \tag{3.3.7.3}$$

3.4. Bilinear Transform

1. Formulate the differential equation for Fig. 3.4.1.1.

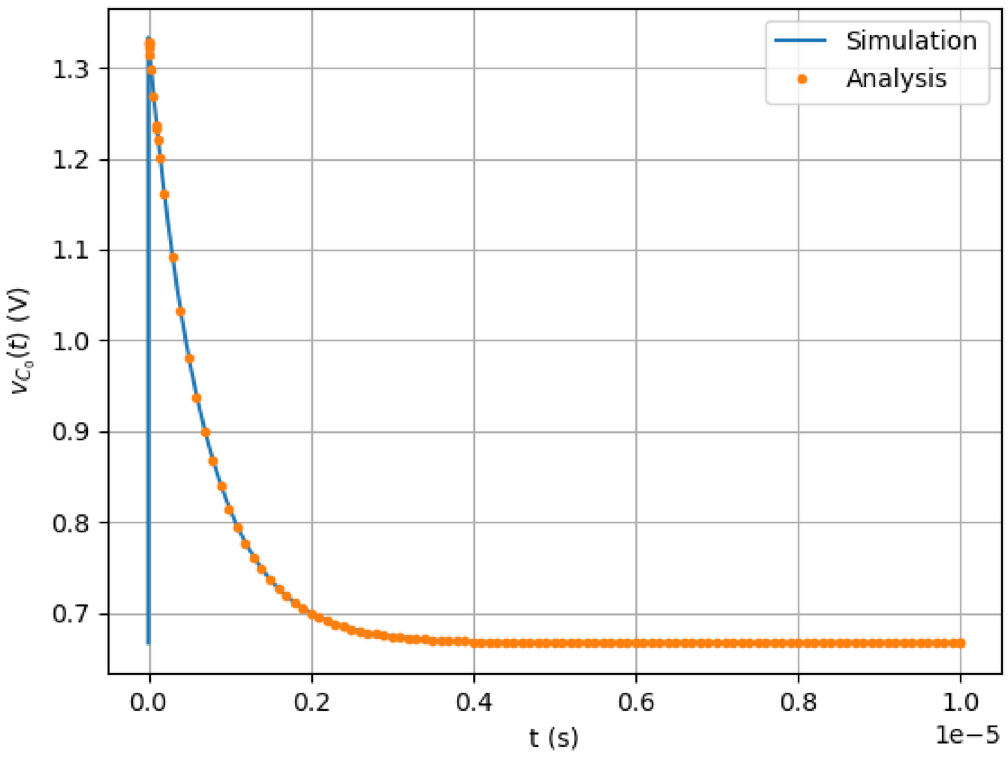


Figure 3.3.6.1: $v_{C_0}(t)$ after the switch is flipped

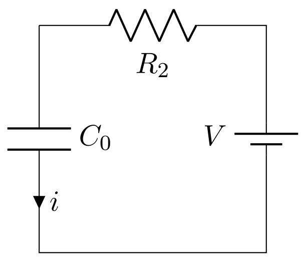


Figure 3.4.1.1:

Solution: Applying KVL on the loop,

$$V - iR_2 - \frac{1}{C_0}\int_0^t i dt = 0 \tag{3.4.1.1}$$

where $i(0) = 0$, $V_C(0) = 0$. Denote by V_C the voltage at the capacitor. Then,

$$i = C\frac{dV_C}{dt} \tag{3.4.1.2}$$

and therefore using (3.4.1.2) in (3.4.1.1), we get the differential equation

$$V - \tau\frac{dV_C}{dt} - V_C = 0 \tag{3.4.1.3}$$

where $\tau \triangleq R_2C_0$ is the time constant of the circuit.

2. Fig. 3.4.1.1 is transformed to Fig. 3.4.2.1 in the s domain. Find the system transfer function

$$H(s) = \frac{V(s)}{V_c(s)} \tag{3.4.2.1}$$

Solution: Using the voltage division formula, the voltage across the capacitor is

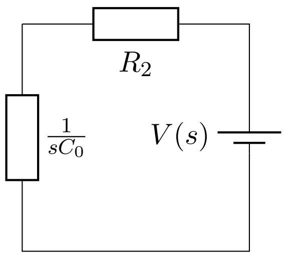


Figure 3.4.2.1:

given by

$$V_{C_0}(s) = V(s)\frac{\frac{1}{sC_0}}{\frac{1}{sC_0} + R_2} = V(s)\frac{1}{1 + sC_0R_2} \tag{3.4.2.2}$$

$$\implies H(s) = \frac{V_{C_0}(s)}{V(s)} = \frac{1}{1 + sC_0R_2} \tag{3.4.2.3}$$

3. Plot $|H(\jmath\omega|$. What kind of filter is it?

   Solution:

4. Using trapezoidal rule for integration, formulate (3.4.1.3) as a difference equation by considering

$$V_C(n) = V_C(t)|_{t=n} \tag{3.4.4.1}$$

   Solution: Integrating (3.4.1.3) between limits $n-1$ to n and applying the trapezoidal formula,

$$\frac{V_C(n) + V_C(n-1)}{2} + \tau\left(V_C(n) - V_C(n-1)\right) = \frac{V(n) + V(n-1)}{2} \tag{3.4.4.2}$$

$$\implies V_C(n)\left(\frac{1}{2} + \tau\right) + V_C(n-1)\left(\frac{1}{2} - \tau\right) = \frac{V(n) + V(n-1)}{2} \tag{3.4.4.3}$$

5. Find

$$H(z) = \frac{V_C(z)}{V(z)}. \tag{3.4.5.1}$$

   Solution: Applying the Z-transform on both sides of (3.4.4.3),

$$V_C(z)\left[(2\tau+1) - z^{-1}(2\tau-1)\right] = V(z)\left(1 + z^{-1}\right) \tag{3.4.5.2}$$

   Hence,

$$H(z) = \frac{1 + z^{-1}}{(2\tau+1) - (2\tau-1)z^{-1}} \tag{3.4.5.3}$$

6. Is the system defined by (3.4.4.3) stable?

Solution: From (3.4.5.3), $H(z)$ has a pole at

$$z = \frac{2\tau - 1}{2\tau + 1} \tag{3.4.6.1}$$

Since

$$\left|\frac{2\tau - 1}{2\tau + 1}\right| < 1, \tag{3.4.6.2}$$

the ROC of $H(z)$ is $|z| > 1$, which implies that (3.4.4.3) represents a stable system.

7. How can you obtain $H(z)$ from $H(s)$?

   Solution: Substituting

$$s = 2\left(\frac{1 - z^{-1}}{1 + z^{-1}}\right) \tag{3.4.7.1}$$

   in (3.4.2.3), we obtain (3.4.5.3). This is a special case of the the bilinear transformation for $T = 1$ where

$$s = \frac{2}{T}\frac{1 - z^{-1}}{1 + z^{-1}} \tag{3.4.7.2}$$

8. Find $V_C(n)$. Verify using ngspice.

   Solution: Since $V(n) = Vu(n)$,

$$V(z) = \frac{V}{1 - z^{-1}} \tag{3.4.8.1}$$

Therefore,

$$V_C(z) = H(z)V(z) \quad = \frac{V\left(1+z^{-1}\right)}{\left(1-z^{-1}\right)\left((2\tau+1)-(2\tau-1)z^{-1}\right)} \tag{3.4.8.2}$$

$$\tag{3.4.8.3}$$

Taking the inverse,

$$V_C(n) = \begin{cases} \frac{V}{2}\left[u(n)\left(1-p^n\right)+u(n-1)\left(1-p^{n-1}\right)\right] & n > 0 \\ 0 & n \le 0 \end{cases} \tag{3.4.8.4}$$

where

$$p = \frac{2\tau-1}{2\tau+1} \tag{3.4.8.5}$$

The following python code

```
cktsig/codes/1_7.py
```

plots V_C for the sampling interval $T = 10^{-7}$ in Fig. 3.4.8.1.

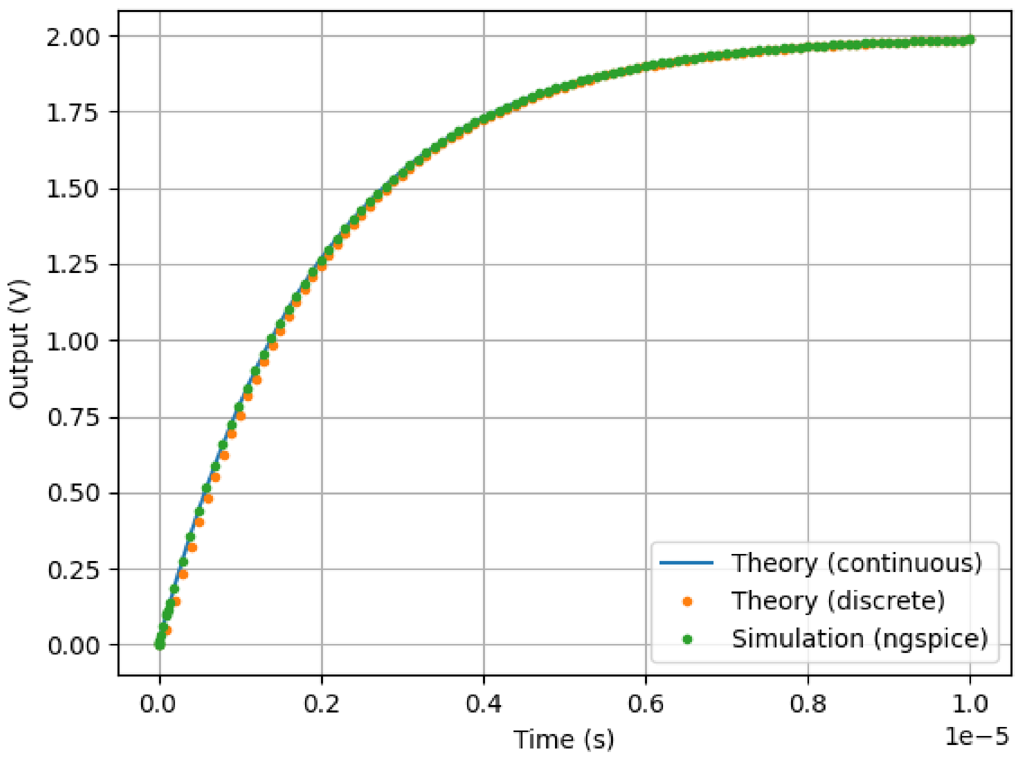


Figure 3.4.8.1: Representation of output across C.

Chapter 4

Frequency Modulation

4.1. Message

Play the message signal using

```
sudo apt install ffmpeg
ffplay fm/input−audio/Sound.wav
```

1. Find the sampling rate of the message.

   Solution: Executing

   ```
   python3 fm/msg/codes/sample_rate.py
   ```

   gives the sampling rate of the input signal as 44100Hz.

2. Plot the spectrum of the message signal using the builtin FFT algorithm.

   Solution: The folowing code plots the spectrum in Fig. 4.1.2.1 using the builtin FFT algorithm of the python library 'Numpy.'

   Executing

   ```
   python3 fm/msg/codes/msg_spec.py
   ```

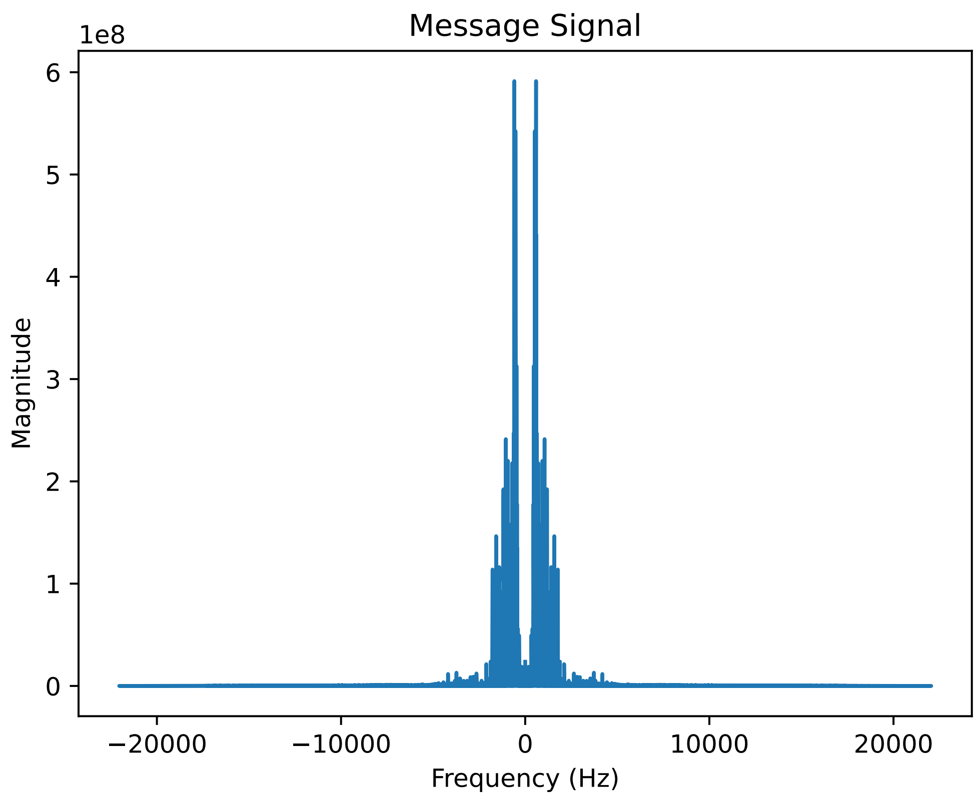


Figure 4.1.2.1: Plot of spectrum of message signal using builtin FFT algorithm.

3. Find the number of samples used to compute the FFT.

   Solution: The following code finds the number of samples used to compute the FFT

   ```
   python3 fm/msg/codes/no_of_samples.py
   ```

   and gives number of samples as 1226536

4. What does the following command do?

   ```
   f_i = np.fft.fftfreq(len(audio_data), d=1/sample_rate)
   ```

Parameter	**Description**	**Value**
f	Frequency	Varies with k
k	index of the DFT output component	$0 \leq k \leq n-1$
n	Length of the input signal	1226536
d	Sample space	1/44100

Table 4.2: Parameters of Message signal

Solution: The np.fft.fftfreq function calculates the frequencies corresponding to the discrete Fourier transform (DFT) output by using the formula:

$$f = \frac{k}{nd} \tag{4.1.4.1}$$

for k=100,

$$f = \frac{100}{1226536(1/44100)}$$

$$f = 3.595Hz$$

The np.fft.fftfreq function generates an array of length n, where each element represents the frequency corresponding to the DFT output component at the respective index.The index k ranges from 0 to n-1, and each index corresponds to a specific frequency component.

5. Plot the spectrum of the message signal by writing your own FFT algorithm.

   Solution:

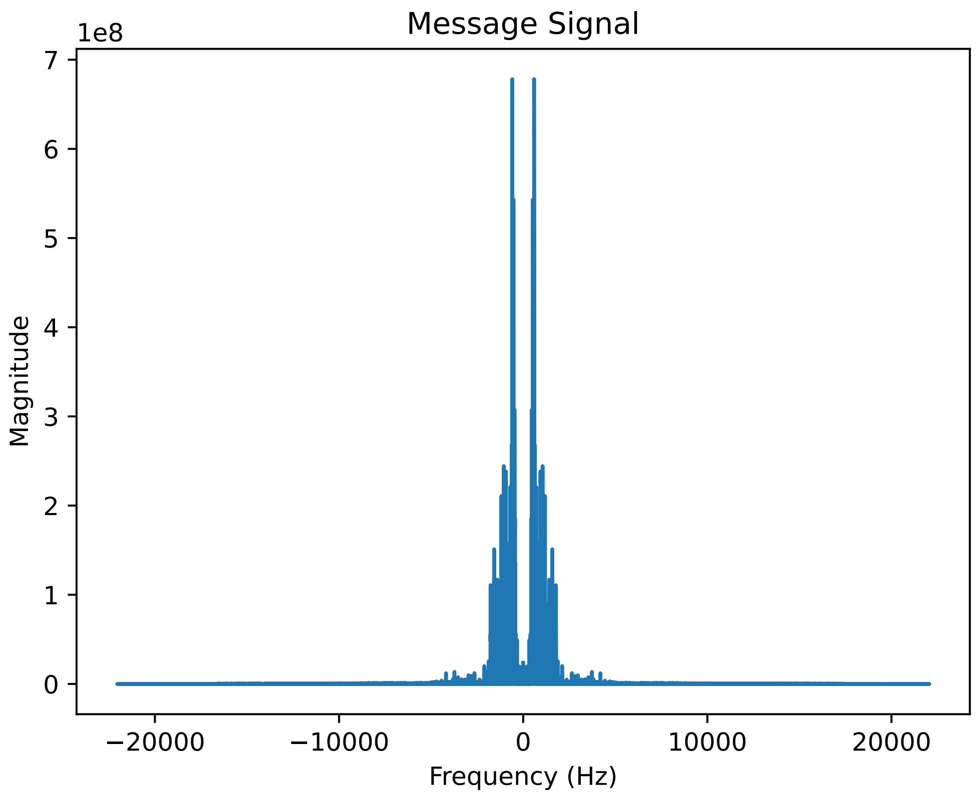


Figure 4.1.5.1: Plot of spectrum of message signal using own FFT algorithm.

The folowing code plots the spectrum in Fig. 4.1.5.1 using the DFT defined in

```
python3 fm/msg/codes/FFTalgorithm.py
```

6. Compute and plot the PSD of the message signal using

   Solution: Executing

```
python3 fm/msg/codes/msg_psd.py
```

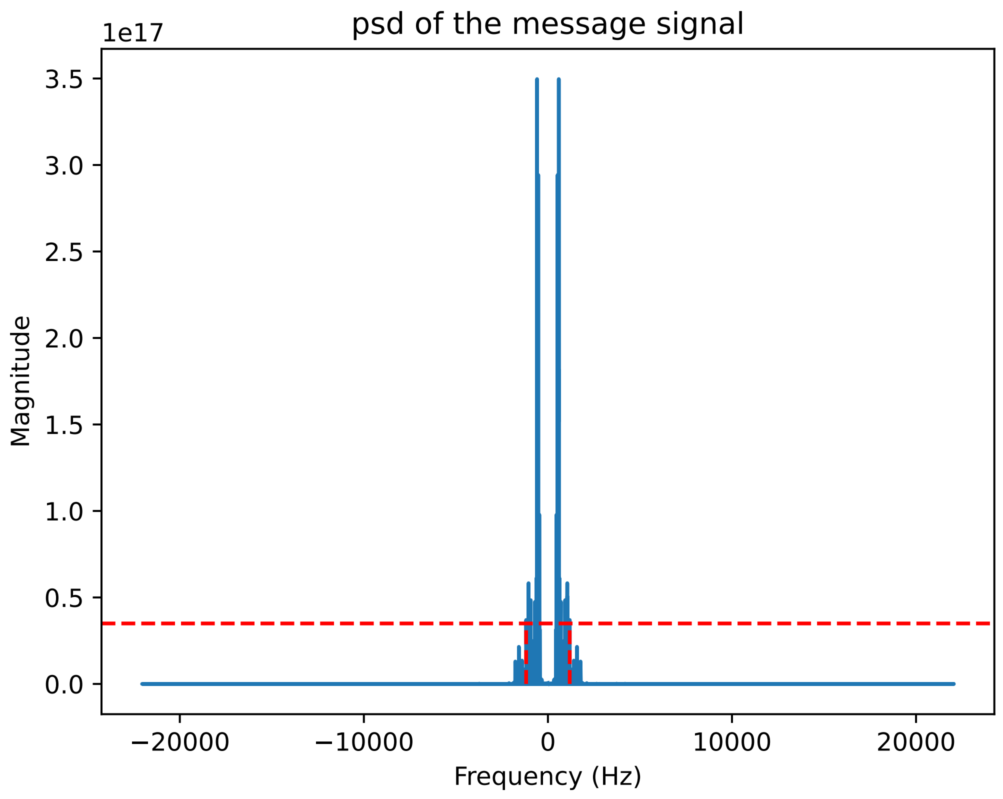


Figure 4.1.6.1: Plot of PSD of the message signal.

7. Find the bandwidth of the message signal. Through a plot, explain the principle used for computing the bandwidth.

   Solution: Executing

   ```
   python3 fm/msg/codes/msg_bw.py
   ```

   gives the bandwidth of the message signal as 2366.62454 Hz

   The threshold of the message signal from the plot is 3.4965 $\times 10^{16}$

   The maximum and minimum frequencies are 1183.31227 Hz and -1183.31227 respectively.

Therefore bandwidth is

$$f_{max} - f_{min} = 1183.31227 - (-1183.31227) = 2366.62454Hz \tag{4.1.7.1}$$

4.2. Transmitter

1. Construct a Transmitter circuit.

   Solution: :

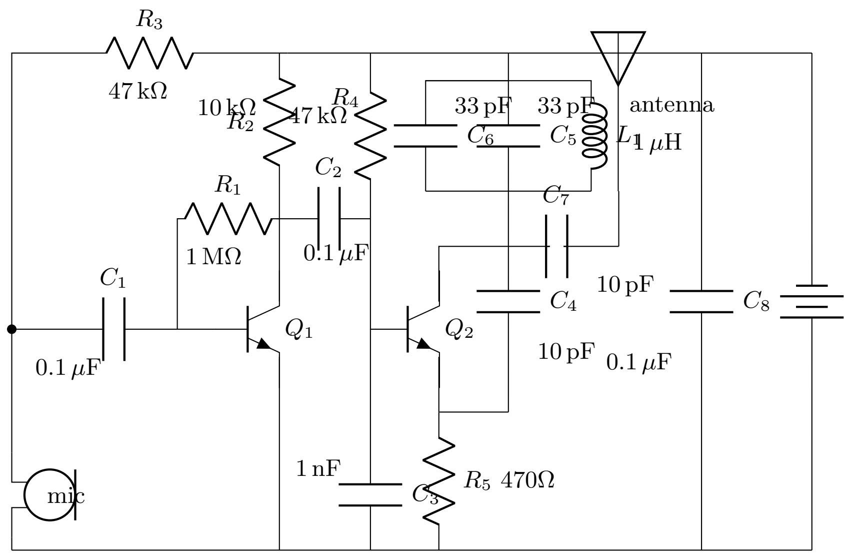


Figure 4.2.1.1: FM Transmitter Circuit diagram

2. List the components in the transmitter circuit diagram.

   Solution:

Component	**Quantity**	**Value**
Transistor	2	2N3904
Inductor	1	1μH
Resistor	1	$1\text{M}\Omega$
	2	470Ω
	2	$47\text{k}\Omega$
	1	$10\text{k}\Omega$
Capacitor		1nF
	3	0.1μF
	2	10pF
	2	3.3pF
Antenna	1	
Condenser mic	1	
Battery/power supply	1	12V

Table 4.4: Transmitter components

3. The modulated signal is given by

$$s(t) = \cos\left(2\pi f_c t + \phi(t)\right) \tag{4.2.3.1}$$

where

$$\phi(t) = 2\pi k_f \int_0^t m(\tau)\, d\tau \tag{4.2.3.2}$$

List the various parameters in a table.

Solution: Parameters are in the table

Parameter	**Description**	**Value**
F_s	Sampling rate	44100 Hz
K_f	Frequency sensitivity	20 Hz/volt
A_c	Amplitude of the carrier signal	1
t	Sampling time	22 μ

$\tau_n = t_0 + (n + 1/2)\Delta t$, where $n = 0, 1, 2, ..., N - 1$. This gives:

$$m(\tau_n) \approx m(n\Delta t)$$

we can approximate the integral in the equation 4.2.3.2

$$\phi(t) \approx 2\pi k f_c \Delta t \sum_{n=0}^{N-1} m(n\Delta t)$$

ϕ at the nth time step $t_n = t_0 + n\Delta t$. Then, we can write:

$$\phi_n = 2\pi k f_c \Delta t \sum_{k=0}^{n-1} m(k\Delta t) \tag{4.2.4.1}$$

$$\phi_{n-1} = 2\pi k f_c \Delta t \sum_{k=0}^{n-2} m(k\Delta t) \tag{4.2.4.2}$$

Subtracting the 4.2.4.2 from 4.2.4.1, we get:

$$\phi_n - \phi_{n-1} = 2\pi k_f f_c \Delta t, m((n-1)\Delta t) \tag{4.2.4.3}$$

5. Plot the spectrum of the transmitted signal.

   Solution: The folowing code plots the spectrum of transmitted signal in Fig. 4.2.5.1

   Executing

```
python3 fm/tx/codes/tx_spec.py
```

6. Compute and plot the PSD of the message signal

   Solution: Executing

H

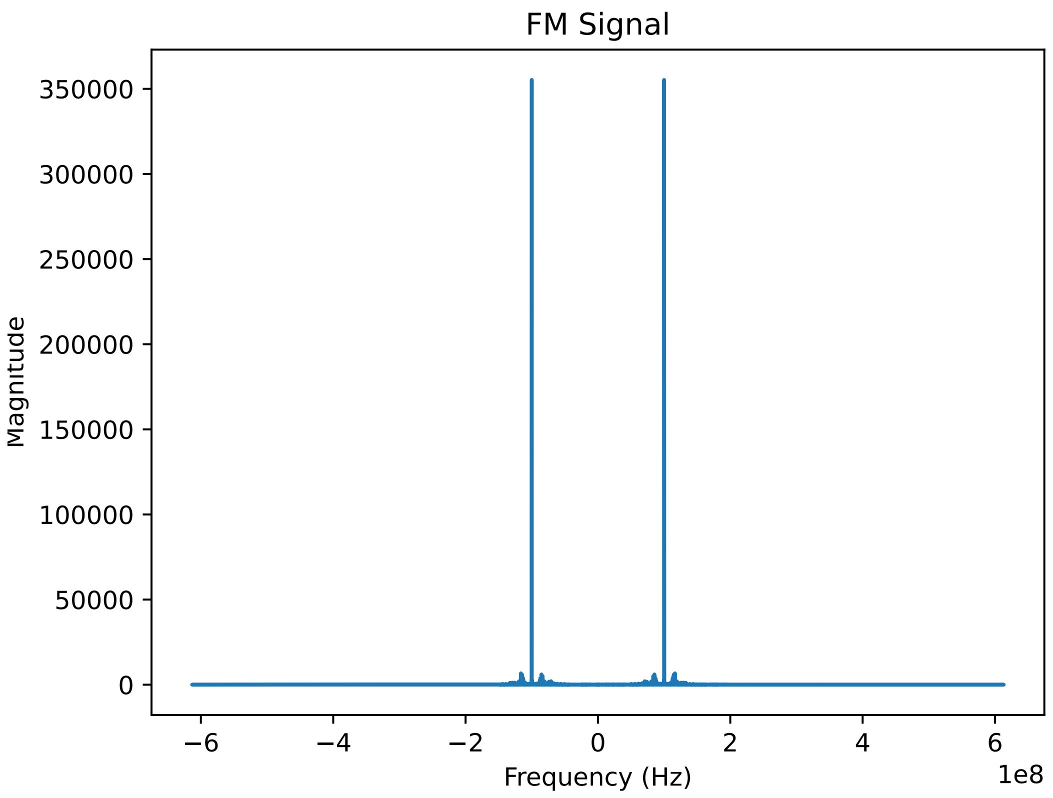


Figure 4.2.5.1: Plot of spectrum of transmitted signal.

```
python3 fm/tx/codes/tx_psd.py
```

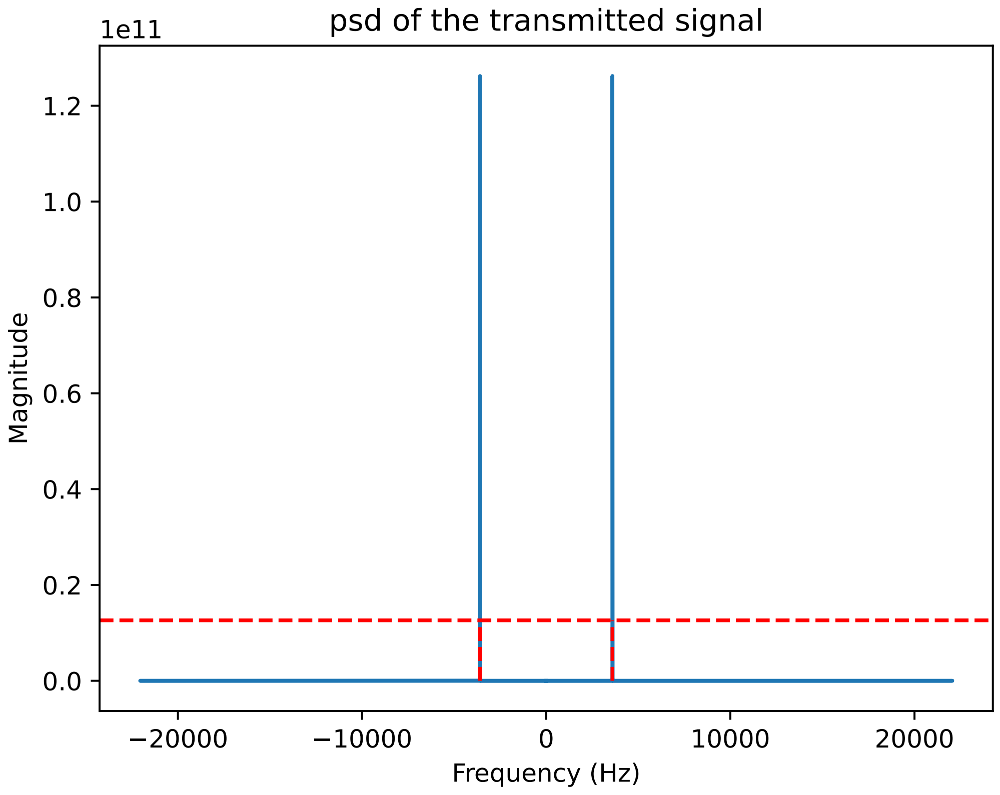


Figure 4.2.6.1: Plot of PSD of the message signal.

7. Compute the bandwidth of the transmitted signal.

   Solution: Executing

   ```
   python3 fm/tx/codes/tx_bw.py
   ```

   gives the bandwidth of the transmitted signal as 7190.69574 Hz

   The threshold of the transmitted signal from the plot is 1.2621 $\times 10^{10}$

The maximum and minimum frequencies are 3595.34787 Hz and -3595.34787 respectively.

Therefore bandwidth is

$$f_{max} - f_{min} = 3595.34787 - (-3595.34787) = 7190.69574Hz \tag{4.2.7.1}$$

4.3. Transmitter using USRP

1. List the various components used to implement transmitter.

   Solution:

   Components are listed in the table Table 4.8

Component	**Type**
USRP	Hardware
GNU radio	Software
Antenna	Hardware

Table 4.8: Components Required

2. Installations required to detect and work with USRP.

   Setting up the dependencies on ubuntu:-

   Run the following command in the terminal.

```
sudo apt—get install autoconf automake build—essential ccache cmake cpufrequtils
    doxygen ethtool \
g++ git inetutils—tools libboost—all—dev libncurses5 libncurses5—dev libusb
    —1.0—0 libusb—1.0—0—dev \
libusb—dev python3—dev python3—mako python3—numpy python3—requests
    python3—scipy python3—setuptools \
python3—ruamel.yaml
```

Getting the source code

The UHD source is stored in a git repository. To download it, run the following command.

```
git clone https://github.com/EttusResearch/uhd.git
```

Generate Makefiles with CMake

```
cd <uhd—repo—path>/host
mkdir build
cd build
cmake ../
```

Build and install

```
make
make test # This step is optional
sudo make install
```

Setup the library path

```
sudo ldconfig
```

Note: Refer the following website for any queries.

```
https://files.ettus.com/manual/page_build_guide.html
```

The picture of USRP and Antenna is given in Fig. 4.3.2.1.This set is used to transmit the FM signals.

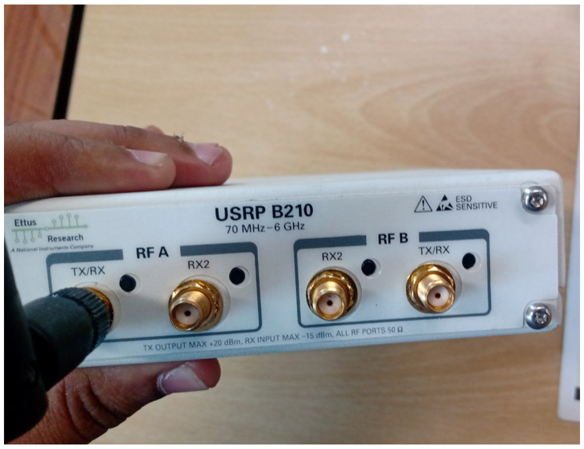


Figure 4.3.2.1: USRP

3. Install and open GNU Radio using the following commands

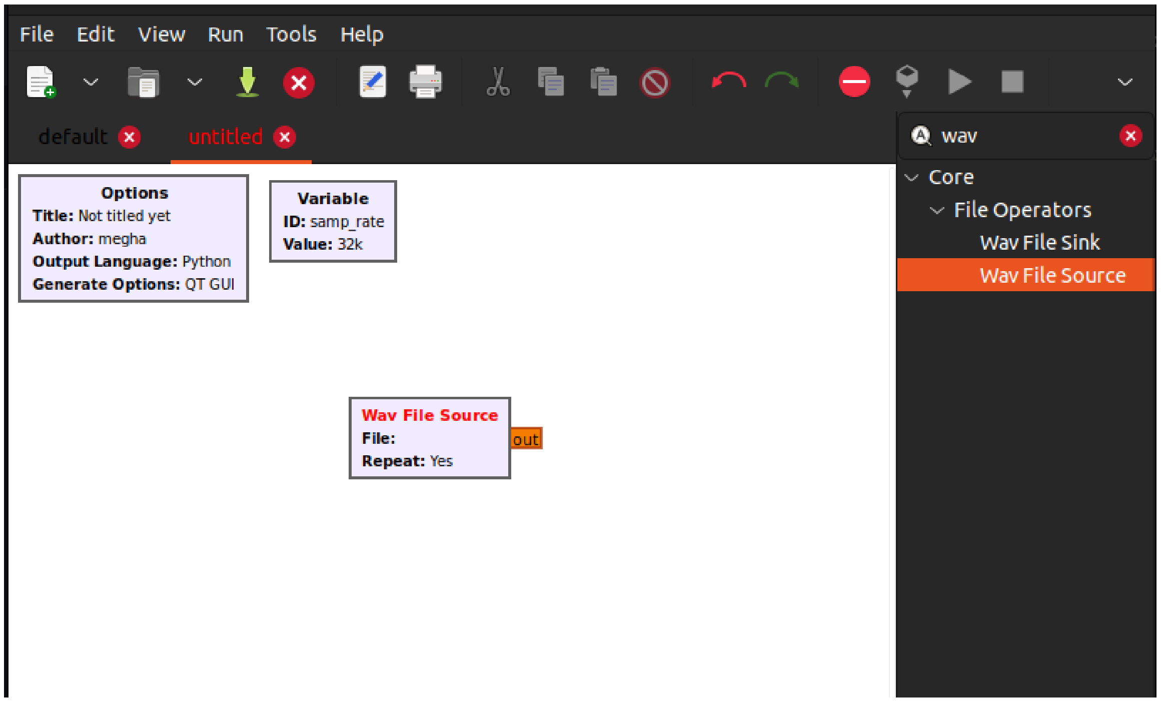
```
sudo apt update
sudo apt install gnuradio
gnuradio–companion
```

4. How to construct the block diagram in GNU radio?

   Solution:

Step-1:

Search for Wav file source block and drag it to the work space.



Options
Title: Not titled yet
Author: megha
Output Language: Python
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 32k

Wav File Source
File:
Repeat: Yes

Figure 4.3.4.1: Adding blocks

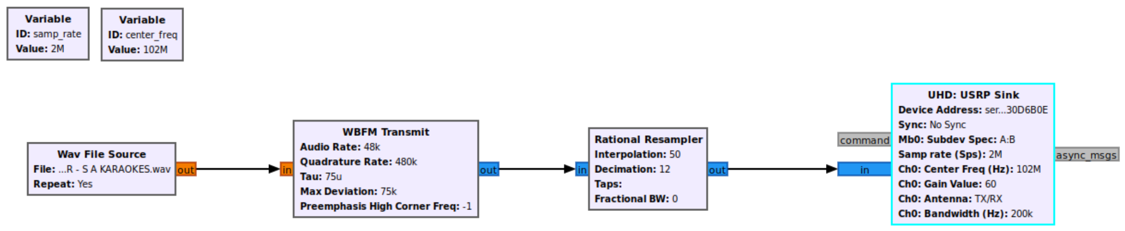
Change the address of the audio file in wav file source block by double clicking on it.

```
fm\tx—gnu\Naatu—Naatu.wav
```

Step-2: Similarly do for WBFM transmit,Rational resampler and UHD:USRP sink block.

Step-3: Change the parameters in each block according to your values by double cliking on it.

Step-4: Connect them according to the flowgraph shown in Fig. 4.3.4.2.



Variable
ID: samp_rate
Value: 2M

Variable
ID: center_freq
Value: 102M

Wav File Source
File: ...R - S A KARAOKES.wav
Repeat: Yes

WBFM Transmit
Audio Rate: 48k
Quadrature Rate: 480k
Tau: 75u
Max Deviation: 75k
Preemphasis High Corner Freq: -1

Rational Resampler
Interpolation: 50
Decimation: 12
Taps:
Fractional BW: 0

UHD: USRP Sink
Device Address: ser...30D6B0E
Sync: No Sync
Mb0: Subdev Spec: A:B
Samp rate (Sps): 2M
Ch0: Center Freq (Hz): 102M
Ch0: Gain Value: 60
Ch0: Antenna: TX/RX
Ch0: Bandwidth (Hz): 200k

Figure 4.3.4.2: Block diagram of Transmitter in GNU Radio

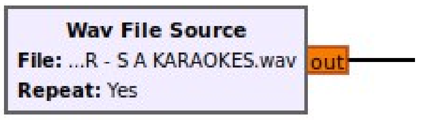
5. Explain each block in block diagram of Transmitter.

   Solution:

   1. Wav file source block:

   The "WAV File Source" block in GNU Radio is used to read audio data from a WAV file and provide it as a continuous stream of samples with specified sample rate in GNU Radio.

   It allows you to use pre-recorded audio files as a source of input for your GNU Radio flowgraph.



Wav File Source
File: ...R - S A KARAOKES.wav
Repeat: Yes

Figure 4.3.5.1: Wav file Source Block

   2.WBFM Transmitter:

   The "WBFM Transmit" block in GNU Radio is used to transmit a wideband FM (WBFM) signal over a specified frequency range.

It takes an audio stream as input and modulates it onto an RF carrier frequency using WBFM modulation techniques.

This block allows you to create and transmit FM signals in GNU Radio

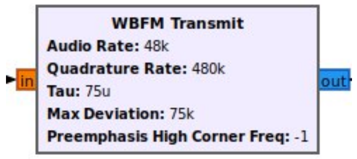

WBFM Transmit
Audio Rate: 48k
Quadrature Rate: 480k
Tau: 75u
Max Deviation: 75k
Preemphasis High Corner Freq: -1

Figure 4.3.5.2: WBFM Transmit Block

3.Rational Resampler:

The "Rational Resampler" block in GNU Radio is used for resampling a signal by a rational factor.

Resampling is the process of changing the sample rate of a signal while preserving its content and characteristics.

The main function of the "Rational Resampler" block is to change the sample rate of the input signal according to the specified rational resampling factor.

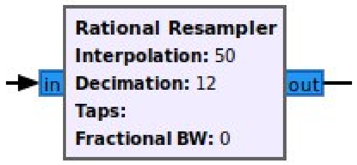
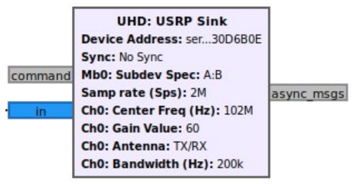

Rational Resampler
Interpolation: 50
Decimation: 12
Taps:
Fractional BW: 0

Figure 4.3.5.3: Rational resampler Block

4.USRP Sink:

The "UHD: USRP Sink" block in GNU Radio is used to send data from a GNU Radio flowgraph to a Universal Software Radio Peripheral (USRP) device for transmission. It acts as an interface between the GNU Radio software and the USRP hardware



UHD: USRP Sink
Device Address: ser...30D6B0E
Sync: No Sync
Mb0: Subdev Spec: A:B
Samp rate (Sps): 2M
Ch0: Center Freq (Hz): 102M
Ch0: Gain Value: 60
Ch0: Antenna: TX/RX
Ch0: Bandwidth (Hz): 200k

Figure 4.3.5.4: USRP Sink Block

Connect USRP to the system and run the below command to know it's serial number or address.

```
sudo uhd_find_devices
```

Double click on USRP sink block in GNU radio and enter the address at device address to work with that respective USRP.

Execute the flowgraph in Fig. 4.3.4.2.

Now we can transmit the FM signal through USRP.

A default python code is generated and stored as

```
fm/tx–gnu/codes/fm_transmitter.py
```

4.4. Receiver

1. List the various components used to implement receiver.

   Solution:

Components are listed in the table Table 4.10

Component	**Type**
RTL-SDR chip	Hardware
GNU radio	Software
Antenna	Hardware

Table 4.10: Components Required

Installation required.

```
sudo apt install rtl-sdr
```

Command to detect rtlsdr.

```
rtl_test -t
```

The picture of RTL-SDR and Antenna is given in Fig. 4.4.1.1.This set is used to receive the Fm signals.

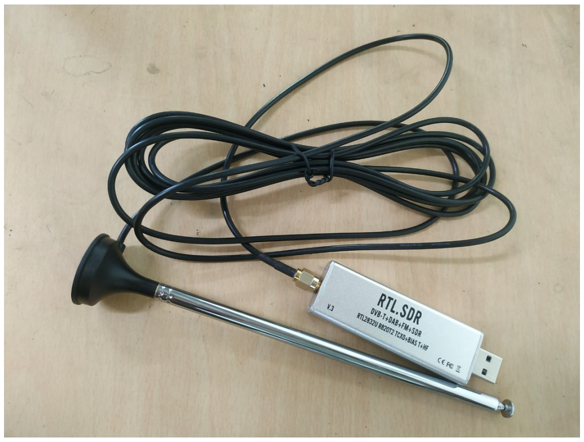


Figure 4.4.1.1: RTL-SDR

2. Install and open GNU Radio using the following commands

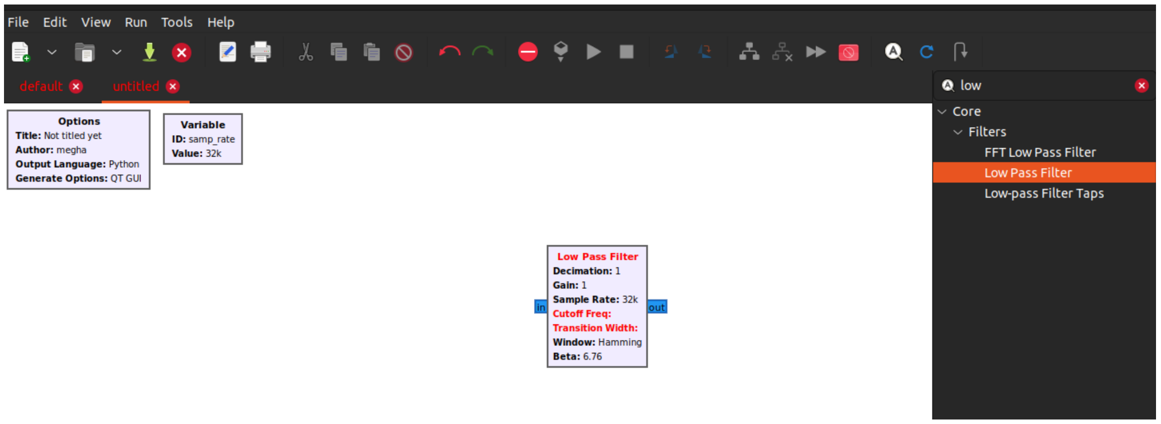
```
sudo apt update

sudo apt install gnuradio

gnuradio—companion
```

3. How to construct the block diagram in GNU radio?

   Solution:

   Step-1:

   Search for Low pass filter block and add it to the work space.



Options
Title: Not titled yet
Author: megha
Output Language: Python
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 32k

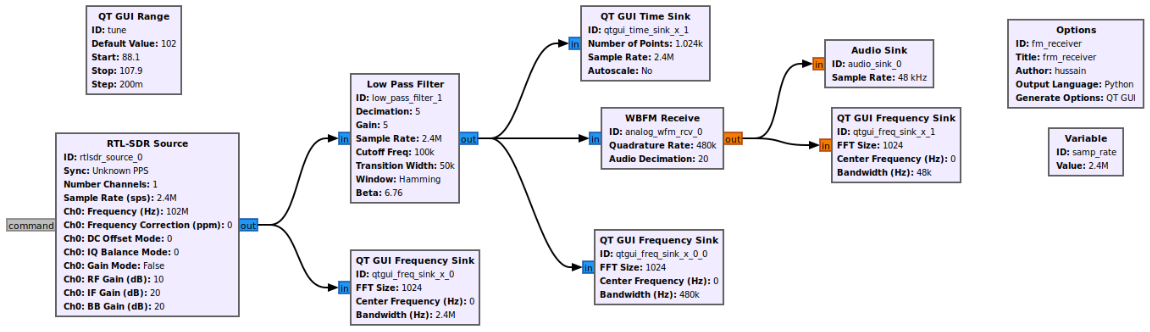
Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 32k
Cutoff Freq:
Transition Width:
Window: Hamming
Beta: 6.76

Figure 4.4.3.1: Adding blocks

   Step-2: Similarly do for RTL-Source block,WBFM and Audio sink block.

   Step-3: Change the parameters in each block according to your values by double clicking on it.

   Step-4: Connect them according to the flowgraph shown in Fig. 4.4.3.2.



QT GUI Range
ID: tune
Default Value: 102
Start: 88.1
Stop: 107.9
Step: 200m

RTL-SDR Source
ID: rtlsdr_source_0
Sync: Unknown PPS
Number Channels: 1
Sample Rate (sps): 2.4M
Ch0: Frequency (Hz): 102M
Ch0: Frequency Correction (ppm): 0
Ch0: DC Offset Mode: 0
Ch0: IQ Balance Mode: 0
Ch0: Gain Mode: False
Ch0: RF Gain (dB): 10
Ch0: IF Gain (dB): 20
Ch0: BB Gain (dB): 20

Low Pass Filter
ID: low_pass_filter_1
Decimation: 5
Gain: 5
Sample Rate: 2.4M
Cutoff Freq: 100k
Transition Width: 50k
Window: Hamming
Beta: 6.76

QT GUI Frequency Sink
ID: qtgui_freq_sink_x_0
FFT Size: 1024
Center Frequency (Hz): 0
Bandwidth (Hz): 2.4M

QT GUI Time Sink
ID: qtgui_time_sink_x_1
Number of Points: 1.024k
Sample Rate: 2.4M
Autoscale: No

WBFM Receive
ID: analog_wfm_rcv_0
Quadrature Rate: 480k
Audio Decimation: 20

QT GUI Frequency Sink
ID: qtgui_freq_sink_x_0_0
FFT Size: 1024
Center Frequency (Hz): 0
Bandwidth (Hz): 480k

Audio Sink
ID: audio_sink_0
Sample Rate: 48 kHz

QT GUI Frequency Sink
ID: qtgui_freq_sink_x_1
FFT Size: 1024
Center Frequency (Hz): 0
Bandwidth (Hz): 48k

Options
ID: fm_receiver
Title: frm_receiver
Author: hussain
Output Language: Python
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 2.4M

Figure 4.4.3.2: Block diagram of Receiver in GNU Radio

Note: Refer the following website for any queries.

https://wiki.gnuradio.org/index.php?title=Creating_Your_First_Block

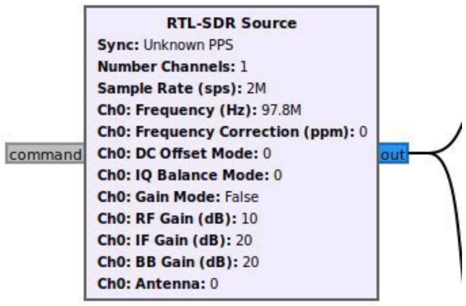
Can check graphical output of each block in both time domain and frequency domain using GUI time sink and GUI frequency sink blocks respectively.

4. Explain each block in block diagram of Receiver.

Solution:

1. RTL-SDR Source:

The RTL-SDR Source block is used to stream samples from RTL-SDR device.

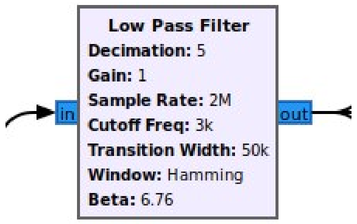


RTL-SDR Source
Sync: Unknown PPS
Number Channels: 1
Sample Rate (sps): 2M
Ch0: Frequency (Hz): 97.8M
Ch0: Frequency Correction (ppm): 0
Ch0: DC Offset Mode: 0
Ch0: IQ Balance Mode: 0
Ch0: Gain Mode: False
Ch0: RF Gain (dB): 10
Ch0: IF Gain (dB): 20
Ch0: BB Gain (dB): 20
Ch0: Antenna: 0

Figure 4.4.4.1: RTL-SDR Source Block

2. Low Pass Filter:

Low Pass Filter removes frequencies that are higher than the cutoff.



Low Pass Filter
Decimation: 5
Gain: 1
Sample Rate: 2M
Cutoff Freq: 3k
Transition Width: 50k
Window: Hamming
Beta: 6.76

Figure 4.4.4.2: Low Pass Filter Block

3. WBFM:

WBFM Receive is a block for demodulating a broadcast FM signal. The output is the demodulated audio.

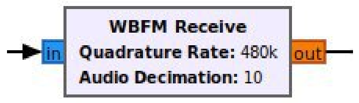


Figure 4.4.4.3: WBFM Block

4. Audio sink:

Audio Sink block provides audio output signal at the computer speakers.

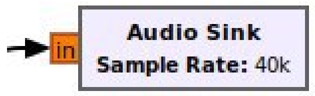


Figure 4.4.4.4: Audio Sink Block

Connect the RTL-SDR to the system and execute the flowgraph in Fig. 4.4.3.2.

We can listen to the sound which we have transmitted.

A python code for entire grc file is generated and saved with the name that you give as ID in options block which can be seen in block diagram.

```
fm/rx/codes/fm_receiver.py
```

5. How to replace default block with own block in gnuradio?

   Solution:

   Let us consider we need to replace default lowpass filter block with own lowpass filter block.

   Step-1: Disable Low pass filter block and search for the python block and add it to the workspace.

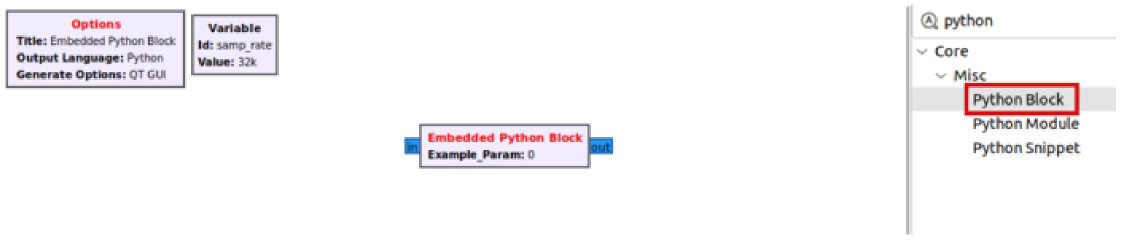


Figure 4.4.5.1: python block

Step-2: Double-click the block to edit the properties as in Fig. 4.4.5.2

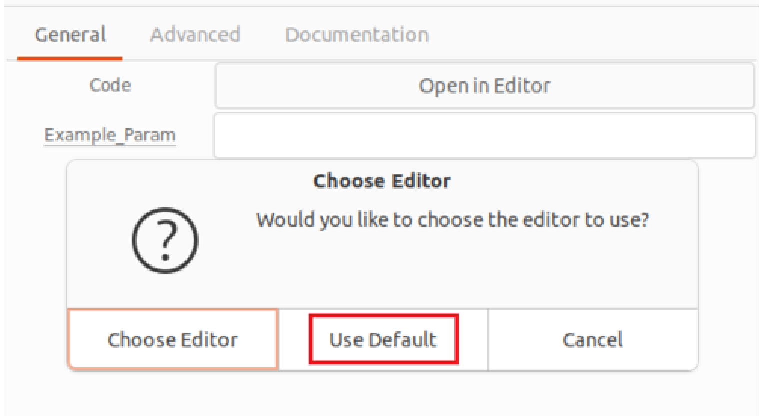
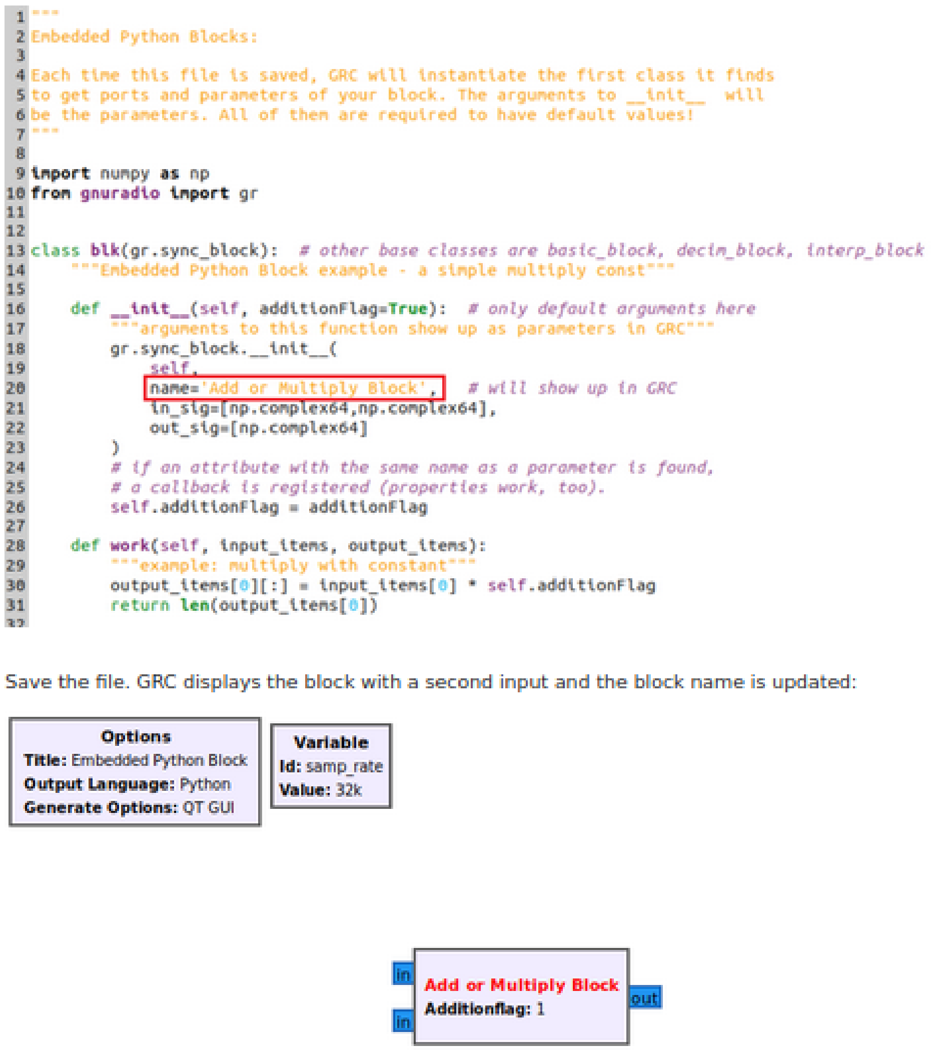


Figure 4.4.5.2: Edit properties

Click on open in editor and use the default editor to which own code can be added where you can change the name of the block by following below step.

Step-3: Change the block name as required by editing in the code as shown in Fig. 4.4.5.3



```python
"""
Embedded Python Blocks:

Each time this file is saved, GRC will instantiate the first class it finds
to get ports and parameters of your block. The arguments to __init__  will
be the parameters. All of them are required to have default values!
"""

import numpy as np
from gnuradio import gr


class blk(gr.sync_block):  # other base classes are basic_block, decim_block, interp_block
    """Embedded Python Block example - a simple multiply const"""

    def __init__(self, additionFlag=True):  # only default arguments here
        """arguments to this function show up as parameters in GRC"""
        gr.sync_block.__init__(
            self,
            name='Add or Multiply Block',   # will show up in GRC
            in_sig=[np.complex64,np.complex64],
            out_sig=[np.complex64]
        )
        # if an attribute with the same name as a parameter is found,
        # a callback is registered (properties work, too).
        self.additionFlag = additionFlag

    def work(self, input_items, output_items):
        """example: multiply with constant"""
        output_items[0][:] = input_items[0] * self.additionFlag
        return len(output_items[0])
```

Save the file. GRC displays the block with a second input and the block name is updated:

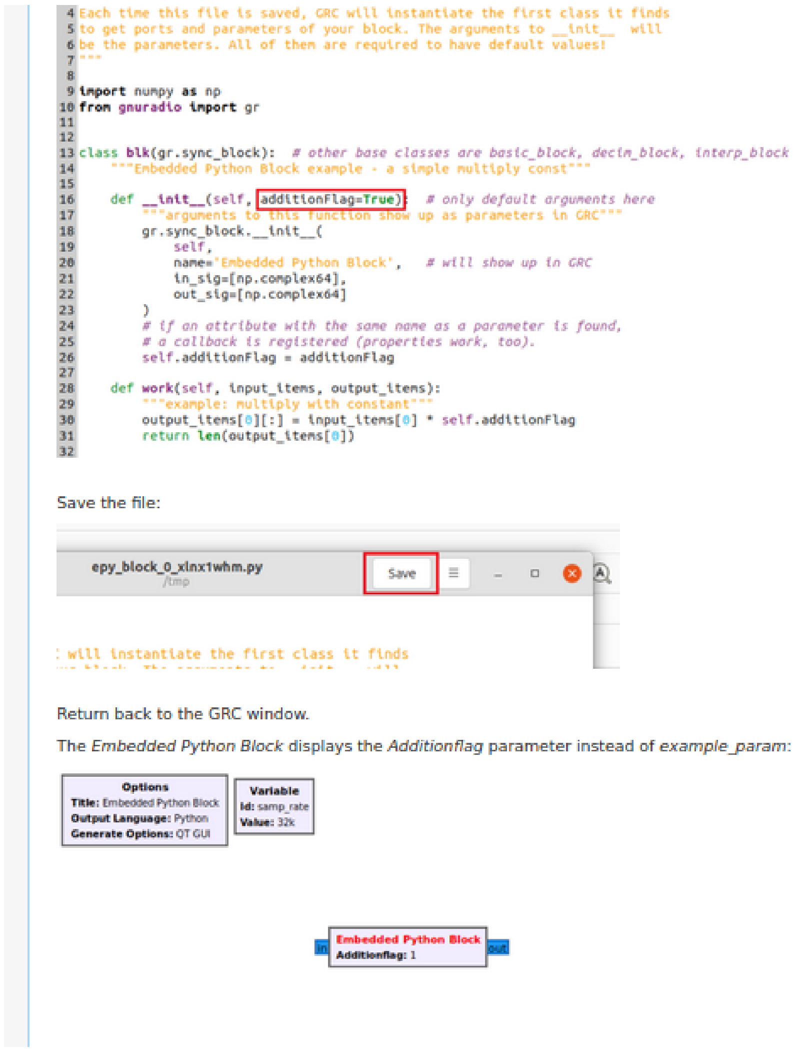
Options
Title: Embedded Python Block
Output Language: Python
Generate Options: QT GUI

Variable
Id: samp_rate
Value: 32k

Add or Multiply Block
Additionflag: 1

Figure 4.4.5.3: Changing block name

Step-4: Change the parameter name as required by editing in the code as shown in Fig. 4.4.5.4



```
Each time this file is saved, GRC will instantiate the first class it finds
to get ports and parameters of your block. The arguments to __init__  will
be the parameters. All of them are required to have default values!
"""

import numpy as np
from gnuradio import gr


class blk(gr.sync_block):  # other base classes are basic_block, decim_block, interp_block
    """Embedded Python Block example - a simple multiply const"""

    def __init__(self, additionFlag=True):  # only default arguments here
        """arguments to this function show up as parameters in GRC"""
        gr.sync_block.__init__(
            self,
            name='Embedded Python Block',   # will show up in GRC
            in_sig=[np.complex64],
            out_sig=[np.complex64]
        )
        # if an attribute with the same name as a parameter is found,
        # a callback is registered (properties work, too).
        self.additionFlag = additionFlag

    def work(self, input_items, output_items):
        """example: multiply with constant"""
        output_items[0][:] = input_items[0] * self.additionFlag
        return len(output_items[0])
```

Save the file:

Return back to the GRC window.

The *Embedded Python Block* displays the *Additionflag* parameter instead of *example_param*:

Figure 4.4.5.4: Changing parameter name

Below path provides custom code for the lowpass filter.

```
fm/rx/codes/lpf_block.py
```

6. Read the data from RTL SDR using python code and without using GNU radio.

   Solution:

Run the below command to install rtlsdr into the python environment.

```
pip install pyrtlsdr
```

The following code reads the data from RTL-SDR without GNU radio.

```
fm/rx/codes/source_own.py
```

Above code generates samples which are read into samples.iq file.

7. Design Low pass filter block without using GNU Radio

   Solution:

   The following code works as low pass filter. The .iq file from above code which has samples from rtlsdr is given as input to the below code on which low pass filter is applied.

```
fm/rx/codes/lpf_own.py
```

The output of above code is read into filtered_output.iq file which can be further processed by reading as input file into other codes.

Chapter 5

Digital Signal Processing

5.1. Signal Generation

Consider the continuous-time signal $\sin(\Omega t)$, where $\Omega = 2\pi f$. The range of f: $-\infty < f < \infty$. Ω is known as the analog frequency.To convert it into a discrete-time signal, put $t = nT_s$, where $T_s = \frac{1}{f_s}$ is the sampling interval and f_s is the sampling frequency.

$$\sin(2\pi f n T_s) = \sin(2\pi \frac{f}{f_s}) \tag{7.1}$$

$$= \sin(\omega n) \tag{7.2}$$

where ω is the digital frequency.

Analog frequencies are represented by Ω, while digital frequencies are represented by ω. $\omega = \Omega T_s$, where Ω lies in $(-\infty, \infty)$, and ω ranges from $[-\pi, \pi]$.

5.2. Frequency Analysis

The input signal is analyzed in the frequency domain using the Discrete Fourier Transform (DFT). The DFT equation is given by:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j2\pi \frac{kn}{N}} \tag{7.3}$$

5.3. Filtering

Filtering the input signal using a predefined filter impulse response h. This is achieved through convolution. Convolution can be mathematically expressed as follows:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k] \tag{7.4}$$

Where:

$x[n]$ is the input signal, $h[n]$ is the filter impulse response and $y[n]$ is the filtered output signal.

The process of filtering can be mathematically represented as follows:

$$y[n] = x[n] * h[n] \tag{7.5}$$

5.3.1. Types of Digital Filters

There are two main types of digital filters:

1. FIR (Finite Impulse Response) Filters
2. IIR (Infinite Impulse Response) Filters

FIR filters are generally easier to design in discrete time compared to IIR filters. FIR filters can exhibit either linear or non-linear phase responses. The window method is one of the simplest methods for FIR filter design. In this method, the goal is often to design a linear phase FIR filter to avoid phase distortions.

5.3.2. Window Method for FIR Filter Design

In the window method for FIR filter design, the process involves the following steps:

1. Define the desired ideal frequency response $H_d(e^{j\omega})$.
2. Compute the inverse discrete Fourier transform (IFFT) of $H_d(e^{j\omega})$ to obtain the impulse response $h_d[n]$.
3. Since $h_d[n]$ is of infinite length, it needs to be truncated using a finite-length window function $w[n]$ to obtain the practical impulse response $h[n]$.

$$h[n] = h_d[n] \times w[n] \tag{3.1}$$

5.3.3. Commonly Used Window Functions

Several commonly used window functions include:

1. Rectangular window

The rectangular window function, also known as the boxcar window, is defined as:

$$w[n] = \begin{cases} 1, & \text{for } 0 \leq n < N \\ 0, & \text{otherwise} \end{cases} \tag{1.1}$$

2. Bartlett (Triangular) Window

The Bartlett window, also known as the triangular window, is defined as:

$$w[n] = \begin{cases} 1 - \frac{|n-(N-1)/2|}{(N-1)/2}, & \text{for } 0 \leq n < N \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

3. Hamming Window

The Hamming window is defined as:

$$w[n] = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right), \quad \text{for } 0 \leq n < N \tag{3.1}$$

4. Hanning (Hann) Window

The Hanning window, also known as the Hann window, is defined as:

$$w[n] = 0.5 - 0.5 \cos\left(\frac{2\pi n}{N-1}\right), \quad \text{for } 0 \leq n < N \tag{4.1}$$

5. Blackman Window

The Blackman window is defined as:

$$w[n] = 0.42 - 0.5 \cos\left(\frac{2\pi n}{N-1}\right) + 0.08 \cos\left(\frac{4\pi n}{N-1}\right), \quad \text{for } 0 \leq n < N \tag{5.1}$$

5.3.4. Low-Pass Filter

The impulse response of a Low-Pass Filter (LPF) is characterized by the expression below, where $h_d[n]$ represents the filter's response at discrete time indices n.

$$h_d[n]\Big|_{n=0} = \lim_{n\to 0} h_d[n] \tag{5.2}$$

$$\tag{5.3}$$

For the LPF, we can express $h_d[n]$ as:

$$\begin{gathered} h_d[n] = \frac{\sin(\omega_c n)}{\pi\omega_c n} \quad \text{for} \quad -\frac{N-1}{2} \leq n \leq \frac{N-1}{2} \\ \text{where } \omega_c = \frac{\omega_c}{\pi} \quad \text{and} \quad \omega_c = \frac{2\pi f_c}{F_s}. \end{gathered} \tag{5.4}$$

Therefore, for $n = 0$, we have:

$$\begin{aligned} h_d[n]\Big|_{n=0} &= \lim_{n\to 0} \omega_c \frac{\cos(\omega_c n)}{\pi n} \\ &= \lim_{n\to 0} \frac{\omega_c n}{\pi n} = \frac{\omega_c}{\pi} \end{aligned} \tag{5.5}$$

For the LPF, the impulse response $h_d[n]$ is given by:

$$h_d[n] = \begin{cases} \frac{\sin(\omega_c n)}{\pi n}, & -\frac{N-1}{2} \leq n \leq \frac{N-1}{2}, \\ \\ \frac{\omega_c}{\pi}, & n = 0, \end{cases}$$

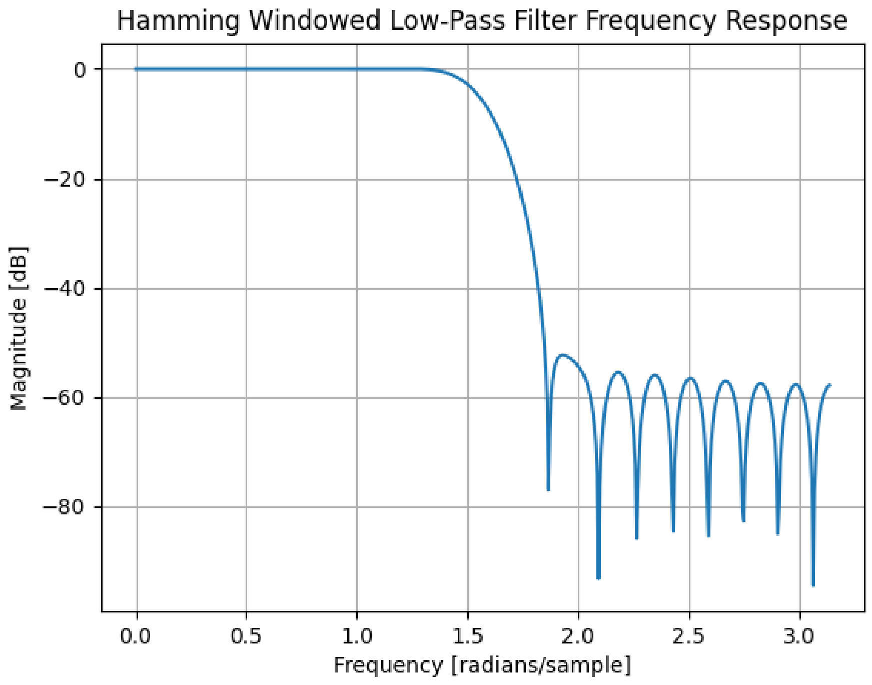


Figure 5.1: Frequency Response

/codes/lpf.py

5.3.5. Band-Pass Filter (BPF)

To design a Band-Pass Filter (BPF), the impulse response $h_d[n]$ is constructed as follows using the Inverse Fast Fourier Transform (IFFT):

$$h_d[n] = \frac{\sin(\omega_{c2} n)}{\pi n} - \frac{\sin(\omega_{c1} n)}{\pi n} \tag{5.6}$$

Where:

ω_{c1} and ω_{c2} are the normalized cutoff frequencies of the filter, f_{c1} and f_{c2} are the corresponding actual cutoff frequencies and n is the discrete time index.

The idea behind designing a Band-Pass Filter (BPF) involves subtracting the magnitude response of a Low-Pass Filter (LPF) with cutoff frequency $\omega_{c1}$from another LPF magnitude response with cutoff frequency ω_{c2}.

For the BPF, the impulse response $h_d[n]$ is given by:

$$h_d[n] = \begin{cases} \frac{\sin(\omega_{c2}n)}{\pi n} - \frac{\sin(\omega_{c1}n)}{\pi n}., & \text{for } -\frac{N-1}{2} \leq n \leq \frac{N-1}{2}, \\ \frac{\omega_{c2}-\omega_{c1}}{\pi}, n = 0, \end{cases} \tag{5.7}$$

For practical implementation, the impulse response of the BPF is often windowed by a window function $w[n]$, resulting in the practical impulse response

$$h[n] = h_d[n] \times w[n] \tag{5.8}$$

```
/codes/bandpass.py
```

This Python script generates and plots the magnitude and phase responses of a Hamming windowed FIR filter, as shown in 5.2

5.3.6. Half Band Filter (HBF)

The Half Band Filter (HBF) is a special case of a Low-Pass Filter (LPF) with a cutoff frequency $f_c = \frac{f_s}{4}$, where f_s is the sampling frequency.

The impulse response of the half band filter is given by:

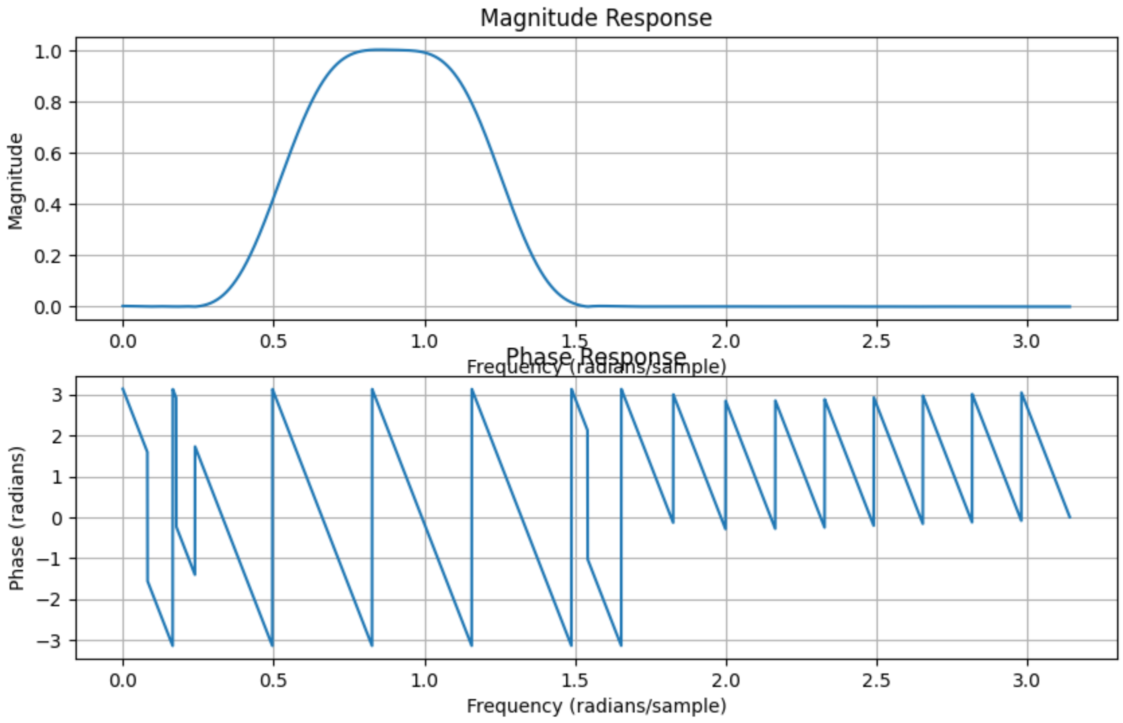


Figure 5.2: Magnitude and Phase Responses of the Hamming Windowed FIR Filter

$$h(2n) = \begin{cases} \frac{1}{2}, & n = 0 \\ 0, & n \neq 0 \end{cases}$$

Where n is the discrete time index.

The HBF has a unique property where it passes half of the frequency spectrum while attenuating the other half. Its impulse response consists of a central value of $\frac{1}{2}$ at $n = 0$ and zeros for all other discrete time indices.

5.3.7. M Band Filter

The M Band Filter is a generalization of the Half Band Filter (HBF), designed to allow for the selective passage of specific frequency bands within the spectrum.

The impulse response of the M Band Filter is given by:

$$h(Mn) = \begin{cases} c, & n = 0 \\ 0, & n \neq 0 \end{cases}$$

Where: n is the discrete time index and c is a constant scaling factor.

For the Half Band Filter, $c = \frac{1}{2}$, and cut-off frequency :$\omega_c = \frac{\pi}{2}$ For the M Band Filter, $c = \frac{1}{M}$, and cut-off frequency :$\omega_c = \frac{\pi}{M}$

5.4. Downsampling

Downsampling is the process of reducing the number of samples in a signal. Given an input signal $x[n]$, downsampling by a factor of M produces a new signal

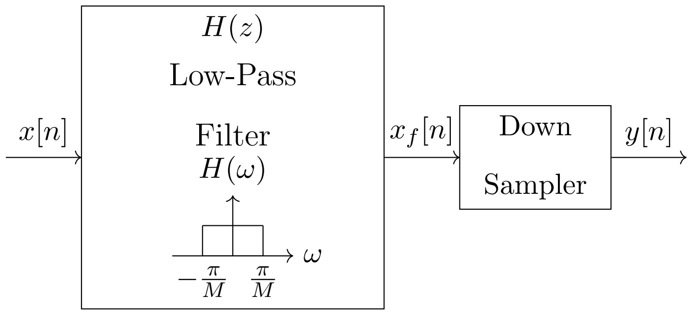
$$y[n] = x[Mn]. \tag{5.9}$$

Time Varying (LTV) System The system described in the code can be characterized as a Linear Time Varying (LTV) system. In the frequency domain, there exists a relation between the input and output of the system.

For a given value of M, the frequency domain relation between the input $X(e^{j\omega})$ and the output $Y(e^{j\omega})$ can be expressed as:

$$Y(e^{j\omega}) = \frac{1}{M} \sum_{k=0}^{M-1} X\left(e^{j(\omega - 2\pi k)/M}\right) \tag{5.10}$$

5.5. Decimation



1. LPF followed by downsampler is known as a decimator.
2. The primary role of the LPF is to prevent aliasing, earning it the name anti-aliasing filter.
3. The cutoff frequency for the LPF is set to π/M.
4. When $M = 2$, the LPF becomes a Half Band Filter (HBF) with a cutoff frequency of $\pi/2$.

The Python code below is accompanied by corresponding plots shown in

```
/codes/decimation.py
```

In Figure 4.1, the input signal composed of multiple sinusoidal components is shown. The output after applying a low-pass filter is depicted in Figure 4.2. The effect of decimation is visible in Figure 4.3, and the outcome of downsampling without a low-pass filter is illustrated in Figure 4.4.

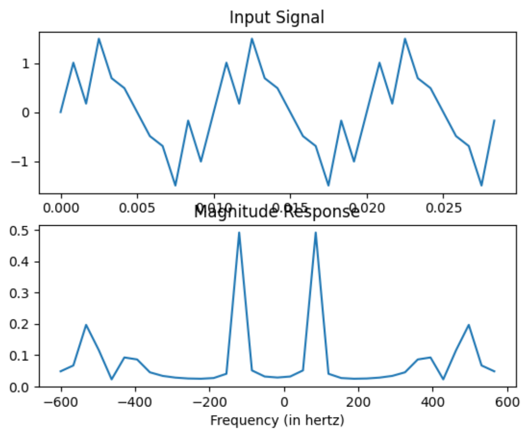


Figure 4.1: Input Signal

5.5.1. Upsampler (Expander)

1. **Time domain relation between input and output:**

$$y[n] = \begin{cases} x[n/L], & \text{if } n \text{ is a multiple of } L \\ 0, & \text{otherwise} \end{cases}$$

2. **Frequency domain relation between input and output:**

$$Y(e^{j\omega}) = X(e^{j(\omega/L)}), \quad \text{for } 0 \leq \omega < 2\pi$$

5.6. Interpolation

1. Upsampler followed by LPF is known as an interpolator.

2. The job of LPF is to remove the unwanted image of $Xe^{j\omega}$. Hence, it is known as an anti-imaging filter.

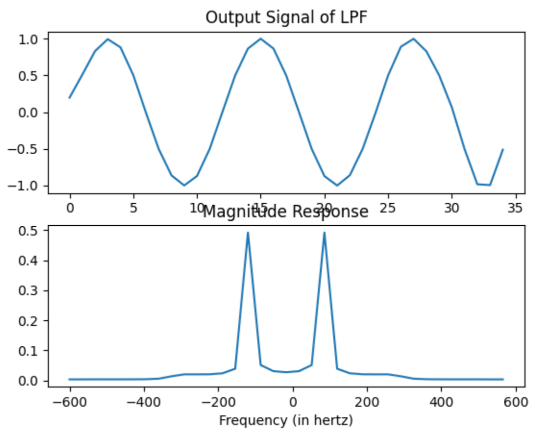


Figure 4.2: Output Signal of Low-Pass Filter

3. Cutoff frequency is $\frac{\pi}{L}$.

4. When $L = 2$, then the LPF is also known as a Half Band Filter (HBF) with a cutoff frequency of $\frac{\pi}{2}$.

The Python code demonstrates upsampling and interpolation of a discrete input signal using a half-band filter. It generates plots illustrating the signal processing steps and visualizes the magnitude response in the frequency domain.

```
/codes/interpolation.py
```

In the results presented in Figure 4.2, we can observe the original input signal. The interpolated output signal and its corresponding frequency response are illustrated in Figure 4.3. Similarly, Figure 4.4 provides insight into the upsampled input signal and its frequency response.

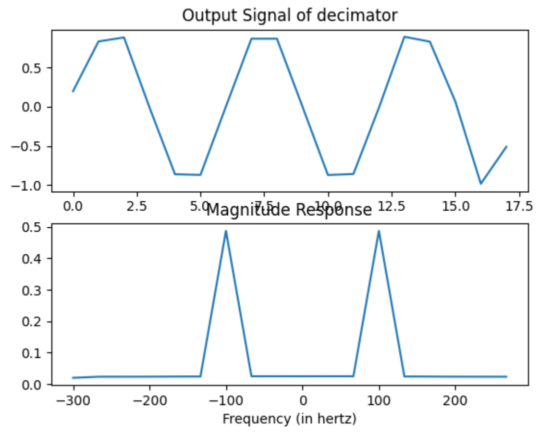


Figure 4.3: Output Signal of Decimator

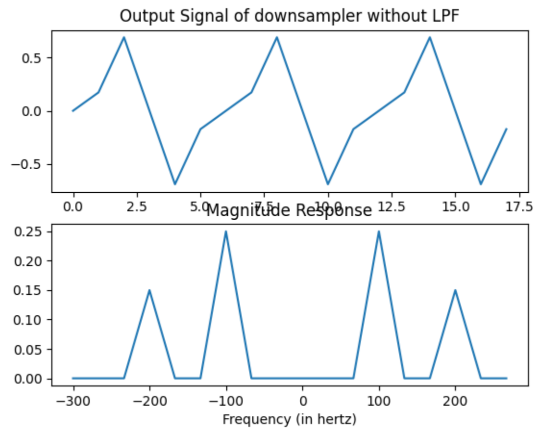


Figure 4.4: Output Signal of Downsampler Without LPF

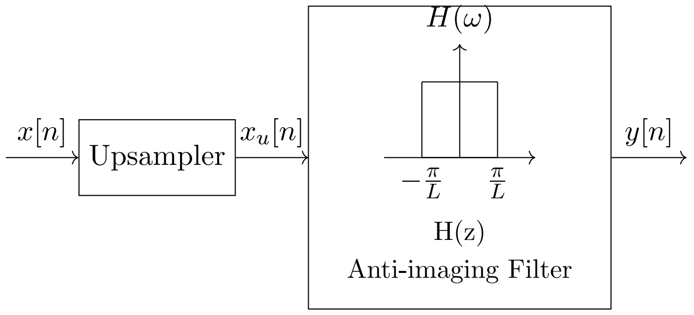


Figure 4.1: Interpolation and Filtering Diagram

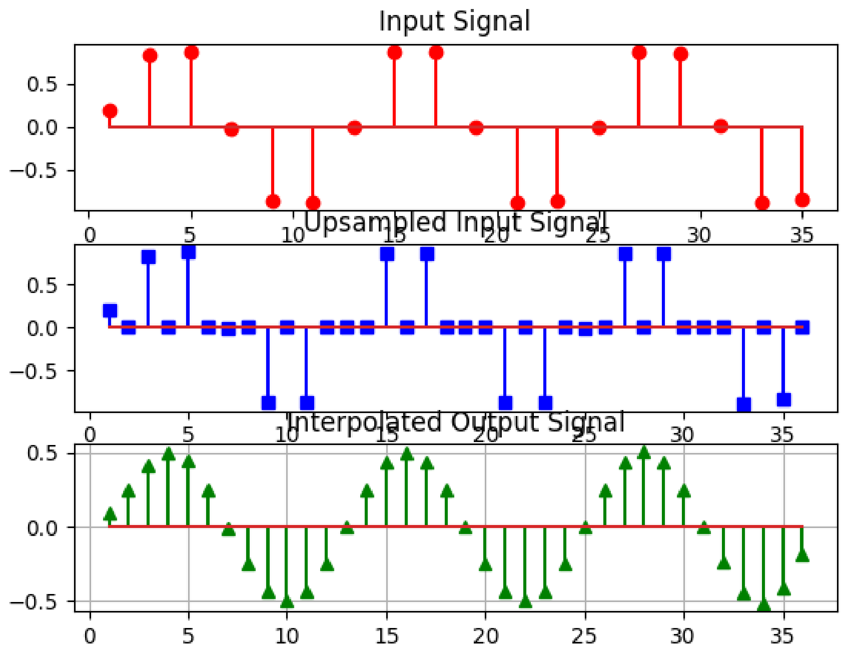


Figure 4.2: Input Signal

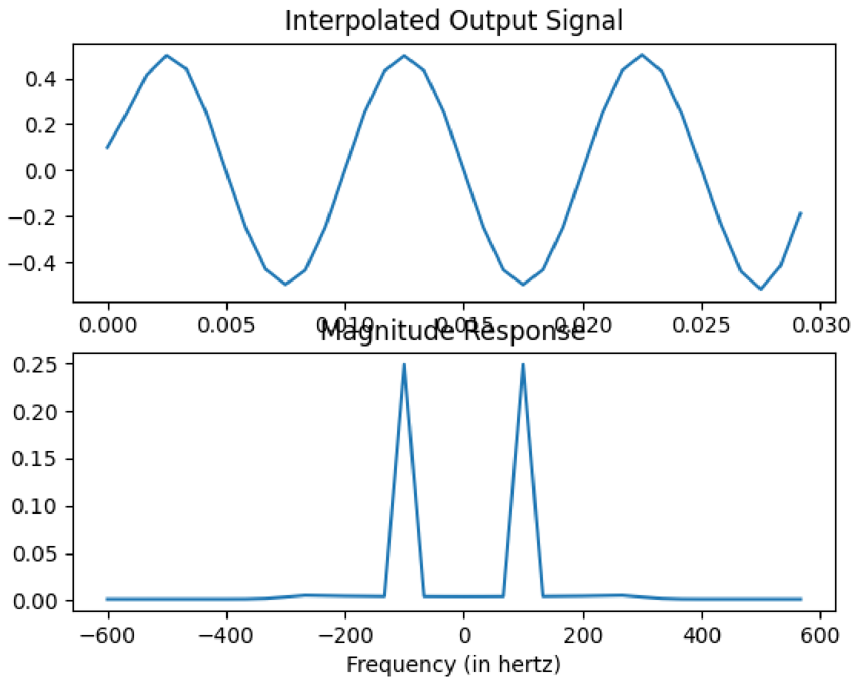


Figure 4.3: Interpolated Output Signal and Frequency Response

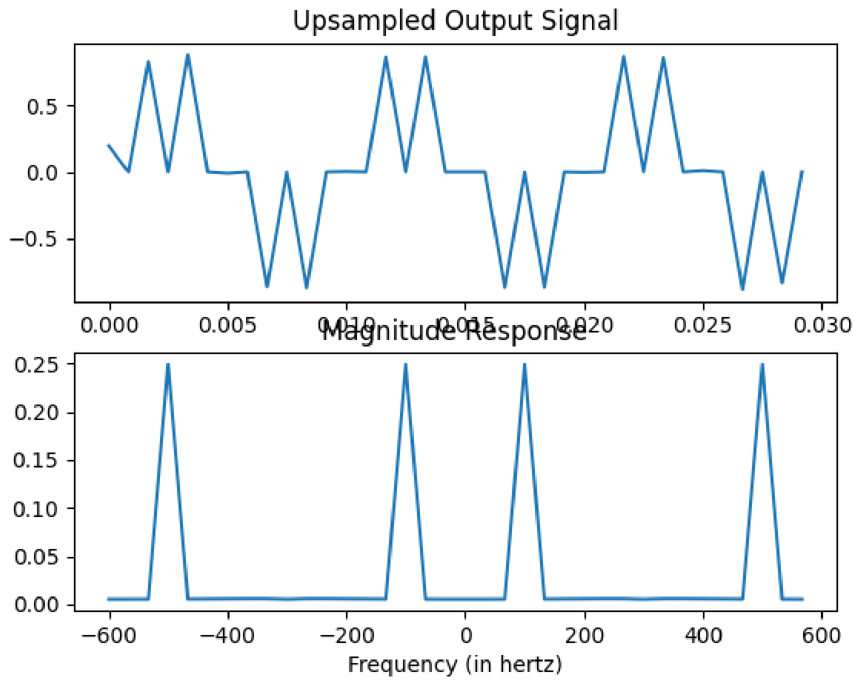


Figure 4.4: Upsampled Input Signal and Frequency Response

Appendix A

Discrete Fourier Transform

A.1. FFT

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

$$X(k) \triangleq \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}, \quad k = 0, 1, \dots, N-1 \tag{A.1.1.1}$$

2. Let

$$W_N = e^{-j2\pi/N} \tag{A.1.2.1}$$

Then the N-point DFT matrix is defined as

$$\mathbf{F}_N = [W_N^{mn}], \quad 0 \le m, n \le N-1 \tag{A.1.2.2}$$

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

A.2. FFT

1. Let

$$\mathbf{I}_4 = \begin{pmatrix} \mathbf{e}_4^1 & \mathbf{e}_4^2 & \mathbf{e}_4^3 & \mathbf{e}_4^4 \end{pmatrix} \tag{A.2.1.1}$$

be the 4×4 identity matrix. Then the 4 point <u>DFT permutation matrix</u> is defined as

$$\mathbf{P}_4 = \begin{pmatrix} \mathbf{e}_4^1 & \mathbf{e}_4^3 & \mathbf{e}_4^2 & \mathbf{e}_4^4 \end{pmatrix} \tag{A.2.1.2}$$

2. The 4 point <u>DFT diagonal matrix</u> is defined as

$$\mathbf{D}_4 = diag \begin{pmatrix} W_8^0 & W_8^1 & W_8^2 & W_8^3 \end{pmatrix} \tag{A.2.2.1}$$

3. Show that

$$W_N^2 = W_{N/2} \tag{A.2.3.1}$$

4. Show that

$$\mathbf{F}_4 = \begin{bmatrix} \mathbf{I}_2 & \mathbf{D}_2 \\ \mathbf{I}_2 & -\mathbf{D}_2 \end{bmatrix} \begin{bmatrix} \mathbf{F}_2 & 0 \\ 0 & \mathbf{F}_2 \end{bmatrix} \mathbf{P}_4 \tag{A.2.4.1}$$

5. Show that

$$\mathbf{F}_N = \begin{bmatrix} \mathbf{I}_{N/2} & \mathbf{D}_{N/2} \\ \mathbf{I}_{N/2} & -\mathbf{D}_{N/2} \end{bmatrix} \begin{bmatrix} \mathbf{F}_{N/2} & 0 \\ 0 & \mathbf{F}_{N/2} \end{bmatrix} \mathbf{P}_N \tag{A.2.5.1}$$

6. Find

$$\mathbf{P}_4 \mathbf{x} \tag{A.2.6.1}$$

7. Show that

$$\mathbf{X} = \mathbf{F}_N \mathbf{x} \tag{A.2.7.1}$$

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

8. Derive the following Step-by-step visualisation of 8-point FFTs into 4-point FFTs and so on

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} X_1(0) \\ X_1(1) \\ X_1(2) \\ X_1(3) \end{bmatrix} + \begin{bmatrix} W_8^0 & 0 & 0 & 0 \\ 0 & W_8^1 & 0 & 0 \\ 0 & 0 & W_8^2 & 0 \\ 0 & 0 & 0 & W_8^3 \end{bmatrix} \begin{bmatrix} X_2(0) \\ X_2(1) \\ X_2(2) \\ X_2(3) \end{bmatrix} \tag{A.2.8.1}$$

$$\begin{bmatrix} X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} = \begin{bmatrix} X_1(0) \\ X_1(1) \\ X_1(2) \\ X_1(3) \end{bmatrix} - \begin{bmatrix} W_8^0 & 0 & 0 & 0 \\ 0 & W_8^1 & 0 & 0 \\ 0 & 0 & W_8^2 & 0 \\ 0 & 0 & 0 & W_8^3 \end{bmatrix} \begin{bmatrix} X_2(0) \\ X_2(1) \\ X_2(2) \\ X_2(3) \end{bmatrix} \tag{A.2.8.2}$$

4-point FFTs into 2-point FFTs

$$\begin{bmatrix} X_1(0) \\ X_1(1) \end{bmatrix} = \begin{bmatrix} X_3(0) \\ X_3(1) \end{bmatrix} + \begin{bmatrix} W_4^0 & 0 \\ 0 & W_4^1 \end{bmatrix} \begin{bmatrix} X_4(0) \\ X_4(1) \end{bmatrix} \tag{A.2.8.3}$$

$$\begin{bmatrix} X_1(2) \\ X_1(3) \end{bmatrix} = \begin{bmatrix} X_3(0) \\ X_3(1) \end{bmatrix} - \begin{bmatrix} W_4^0 & 0 \\ 0 & W_4^1 \end{bmatrix} \begin{bmatrix} X_4(0) \\ X_4(1) \end{bmatrix} \tag{A.2.8.4}$$

$$\begin{bmatrix} X_2(0) \\ X_2(1) \end{bmatrix} = \begin{bmatrix} X_5(0) \\ X_5(1) \end{bmatrix} + \begin{bmatrix} W_4^0 & 0 \\ 0 & W_4^1 \end{bmatrix} \begin{bmatrix} X_6(0) \\ X_6(1) \end{bmatrix} \tag{A.2.8.5}$$

$$\begin{bmatrix} X_2(2) \\ X_2(3) \end{bmatrix} = \begin{bmatrix} X_5(0) \\ X_5(1) \end{bmatrix} - \begin{bmatrix} W_4^0 & 0 \\ 0 & W_4^1 \end{bmatrix} \begin{bmatrix} X_6(0) \\ X_6(1) \end{bmatrix} \tag{A.2.8.6}$$

$$P_8 \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \\ x(7) \end{bmatrix} = \begin{bmatrix} x(0) \\ x(2) \\ x(4) \\ x(6) \\ x(1) \\ x(3) \\ x(5) \\ x(7) \end{bmatrix} \tag{A.2.8.7}$$

$$P_4 \begin{bmatrix} x(0) \\ x(2) \\ x(4) \\ x(6) \end{bmatrix} = \begin{bmatrix} x(0) \\ x(4) \\ x(2) \\ x(6) \end{bmatrix} \tag{A.2.8.8}$$

$$P_4 \begin{bmatrix} x(1) \\ x(3) \\ x(5) \\ x(7) \end{bmatrix} = \begin{bmatrix} x(1) \\ x(5) \\ x(3) \\ x(7) \end{bmatrix} \tag{A.2.8.9}$$

Therefore,

$$\begin{bmatrix} X_3(0) \\ X_3(1) \end{bmatrix} = F_2 \begin{bmatrix} x(0) \\ x(4) \end{bmatrix} \tag{A.2.8.10}$$

$$\begin{bmatrix} X_4(0) \\ X_4(1) \end{bmatrix} = F_2 \begin{bmatrix} x(2) \\ x(6) \end{bmatrix} \tag{A.2.8.11}$$

$$\begin{bmatrix} X_5(0) \\ X_5(1) \end{bmatrix} = F_2 \begin{bmatrix} x(1) \\ x(5) \end{bmatrix} \tag{A.2.8.12}$$

$$\begin{bmatrix} X_6(0) \\ X_6(1) \end{bmatrix} = F_2 \begin{bmatrix} x(3) \\ x(7) \end{bmatrix} \tag{A.2.8.13}$$

9. For

$$\mathbf{x} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 2 \\ 1 \end{pmatrix} \tag{A.2.9.1}$$

compte the DFT using (A.2.7.1)

10. Repeat the above exercise using the FFT after zero padding $\mathbf{x}$.

11. Write a C program to compute the 8-point FFT.

A.3. Power Spectral Density(PSD)

1. Power spectral density (PSD) is a measure of the power distribution over frequency components of a signal.The PSD of (A.1.1.1)is given by

$$X(k) = |X(k)|^2 \tag{A.3.1.1}$$

Appendix B

Filter Design

B.1. Introduction

We are supposed to design the equivalent FIR and IIR filter realizations for filter number 114. This is a bandpass filter whose specifications are available below.

B.2. Filter Specifications

The sampling rate for the filter has been specified as $F_s = 48$ kHz. If the un-normalized discrete-time (natural) frequency is F, the corresponding normalized digital filter (angular) frequency is given by $\omega = 2\pi \left(\frac{F}{F_s}\right)$.

B.2.1. The Digital Filter

1. Tolerances: The passband (δ_1) and stopband (δ_2) tolerances are given to be equal, so we let $\delta_1 = \delta_2 = \delta = 0.15$.

2. Passband: The passband of filter number j, j going from 109 to 135 is from {3 + 0.6(j-109)}kHz to {3 + 0.6(j-107)}kHz. Since our filter number is 114, substituting

$j = 114$ gives the passband range for our bandpass filter as 6 kHz - 7.2 kHz. Hence, the un-normalized discrete time filter passband frequencies are $F_{p1} = 7.2$ kHz and $F_{p2} = 6.0$ kHz. The corresponding normalized digital filter passband frequencies are $\omega_{p1} = 2\pi\frac{F_{p1}}{F_s} = 0.3\pi$ and $\omega_{p2} = 2\pi\frac{F_{p2}}{F_s} = 0.25\pi$ kHz. The centre frequency is then given by $\omega_c = \frac{\omega_{p1}+\omega_{p2}}{2} = 0.275\pi$.

3. Stopband: The transition band for bandpass filters is $\Delta F = 0.3$ kHz on either side of the passband. Hence, the un-normalized stopband frequencies are $F_{s1} = 7.2+0.3 = 7.5$ kHz and $F_{s2} = 6.0 - 0.3 = 5.7$ kHz. The corresponding normalized frequencies are $\omega_{s1} = 0.3125\pi$ and $\omega_{s2} = 0.2375\pi$.

B.2.2. The Analog filter

In the bilinear transform, the analog filter frequency (Ω) is related to the corresponding digital filter frequency (ω) as $\Omega = \tan\frac{\omega}{2}$. Using this relation, we obtain the analog passband and stopband frequencies as $\Omega_{p1} = 0.5095$, $\Omega_{p2} = 0.4142$ and $\Omega_{s1} = 0.5345$, $\Omega_{s2} = 0.3914$ respectively.

B.3. The IIR Filter Design

Filter Type: We are supposed to design filters whose stopband is monotonic and passband equiripple. Hence, we use the Chebyschev approximation to design our bandpass IIR filter.

B.3.1. The Analog Filter

1. Low Pass Filter Specifications: If $H_{a,BP}(j\Omega)$ be the desired analog band pass filter, with the specifications provided in Section 2.2, and $H_{a,LP}(j\Omega_L)$ be the equivalent low pass filter, then

$$\Omega_L = \frac{\Omega^2 - \Omega_0^2}{B\Omega} \tag{1.1}$$

where $\Omega_0 = \sqrt{\Omega_{p1}\Omega_{p2}} = 0.4594$ and $B = \Omega_{p1} - \Omega_{p2} = 0.0953$. The low pass filter has the passband edge at $\Omega_{Lp} = 1$ and stopband edges at $\Omega_{Ls_1} = 1.4653$ and $\Omega_{Ls_2} = -1.5511$. We choose the stopband edge of the analog low pass filter as $\Omega_{Ls} = \min(|\Omega_{Ls_1}|, |\Omega_{Ls_2}|) = 1.4653$.

2. The Low Pass Chebyschev Filter Paramters: The magnitude squared of the Chebyschev low pass filter is given by

$$|H_{a,LP}(j\Omega_L)|^2 = \frac{1}{1 + \epsilon^2 c_N^2(\Omega_L/\Omega_{Lp})} \tag{2.1}$$

where $c_N(x) = \cosh(N \cosh^{-1} x)$ and the integer N, which is the order of the filter, and ϵ are design paramters. Since $\Omega_{Lp} = 1$, (2.1) may be rewritten as

$$|H_{a,LP}(j\Omega_L)|^2 = \frac{1}{1 + \epsilon^2 c_N^2(\Omega_L)} \tag{2.2}$$

Also, the design paramters have the following constraints

$$\begin{aligned} \frac{\sqrt{D_2}}{c_N(\Omega_{Ls})} &\le \epsilon \le \sqrt{D_1}, \\ N &\ge \left\lceil \frac{\cosh^{-1}\sqrt{D_2/D_1}}{\cosh^{-1}\Omega_{Ls}} \right\rceil, \end{aligned} \tag{2.3}$$

where $D_1 = \frac{1}{(1-\delta)^2} - 1$ and $D_2 = \frac{1}{\delta^2} - 1$. After appropriate substitutions, we obtain

$N \geq 4$ and $0.3184 \leq \epsilon \leq 0.6197$. In Figure 1, we plot $|H(j\Omega)|$ for a range of values of ϵ, for $N = 4$. We find that for larger values of ϵ, $|H(j\Omega)|$ decreases in the transition band. We choose $\epsilon = 0.4$ for our IIR filter design.

3. <u>The Low Pass Chebyschev Filter:</u> Thus, we obtain

$$|H_{a,LP}(j\Omega_L)|^2 = \frac{1}{1 + 0.16c_4^2(\Omega_L)} \tag{3.1}$$

where

$$c_4(x) = 8x^4 + 8x^2 + 1. \tag{3.2}$$

The poles of the frequency response in (2.1) lying in the left half plane are in general obtained as $r_1 \cos\phi_k + jr_2 \sin\phi_k$, where

$$\phi_k = \frac{\pi}{2} + \frac{(2k+1)\pi}{2N}, k = 0, 1, \ldots, N-1$$
$$r_1 = \frac{\beta^2 - 1}{2\beta}, r_2 = \frac{\beta^2 + 1}{2\beta}, \beta = \left[\frac{\sqrt{1+\epsilon^2}+1}{\epsilon}\right]^{\frac{1}{N}} \tag{3.3}$$

Thus, for N even, the low-pass stable Chebyschev filter, with a gain G has the form

$$H_{a,LP}(s_L) = \frac{G_{LP}}{\prod_{k=1}^{\frac{N}{2}-1}(s_L^2 - 2r_1\cos\phi_k s_L + r_1^2\cos^2\phi_k + r_2^2\sin^2\phi_k)} \tag{3.4}$$

Substituting $N = 4$, $\epsilon = 0.5$ and $H_{a,LP}(j) = \frac{1}{\sqrt{1+\epsilon^2}}$, from (3.3) and (3.4), we obtain

$$H_{a,LP}(s_L) = \frac{0.3125}{s_L^4 + 1.1068s_L^3 + 1.6125s_L^2 + 0.9140s_L + 0.3366} \tag{3.5}$$

In Fig. 4.2 we plot $|H(j\Omega)|$ using (3.1) and (3.5), thereby verifying that our low-pass Chebyschev filter design meets the specifications.

4. <u>The Band Pass Chebyschev Filter:</u> The analog bandpass filter is obtained from (3.5) by substituting $s_L = \frac{s^2+\Omega_0^2}{Bs}$. Hence

$$H_{a,BP}(s) = G_{BP} H_{a,LP}(s_L)\Big|_{s_L = \frac{s^2+\Omega_0^2}{Bs}}, \tag{4.1}$$

where G_{BP} is the gain of the bandpass filter. After appropriate substitutions, and evaluating the gain such that $H_{a,BP}(j\Omega_{p1}) = 1$, we obtain

$$H_{a,BP}(s) = \frac{2.7776 \times 10^{-5} s^4}{s^8 + 0.1055s^7 + 0.8589s^6 + 0.0676s^5 + 0.2735s^4 + 0.0143s^3 + 0.0383s^2 + 0.001s + 0.002} \tag{4.2}$$

In Fig 4.3, we plot $|H_{a,BP}(j\Omega)|$ as a function of Ω for both positve as

well as negative frequencies. We find that the passband and stopband frequencies in the figure match well with those obtained analytically through the bilinear transformation.

B.3.2. The Digital Filter

From the bilinear transformation, we obtain the digital bandpass filter from the corresponding analog filter as

$$H_{d,BP}(z) = G H_{a,BP}(s)\Big|_{s = \frac{1-z^{-1}}{1+z^{-1}}} \tag{4.3}$$

where G is the gain of the digital filter. From (4.2) and (4.3), we obtain

$$H_{d,BP}(z) = G\frac{N(z)}{D(z)} \tag{4.4}$$

where $G = 2.7776 \times 10^{-5}$,

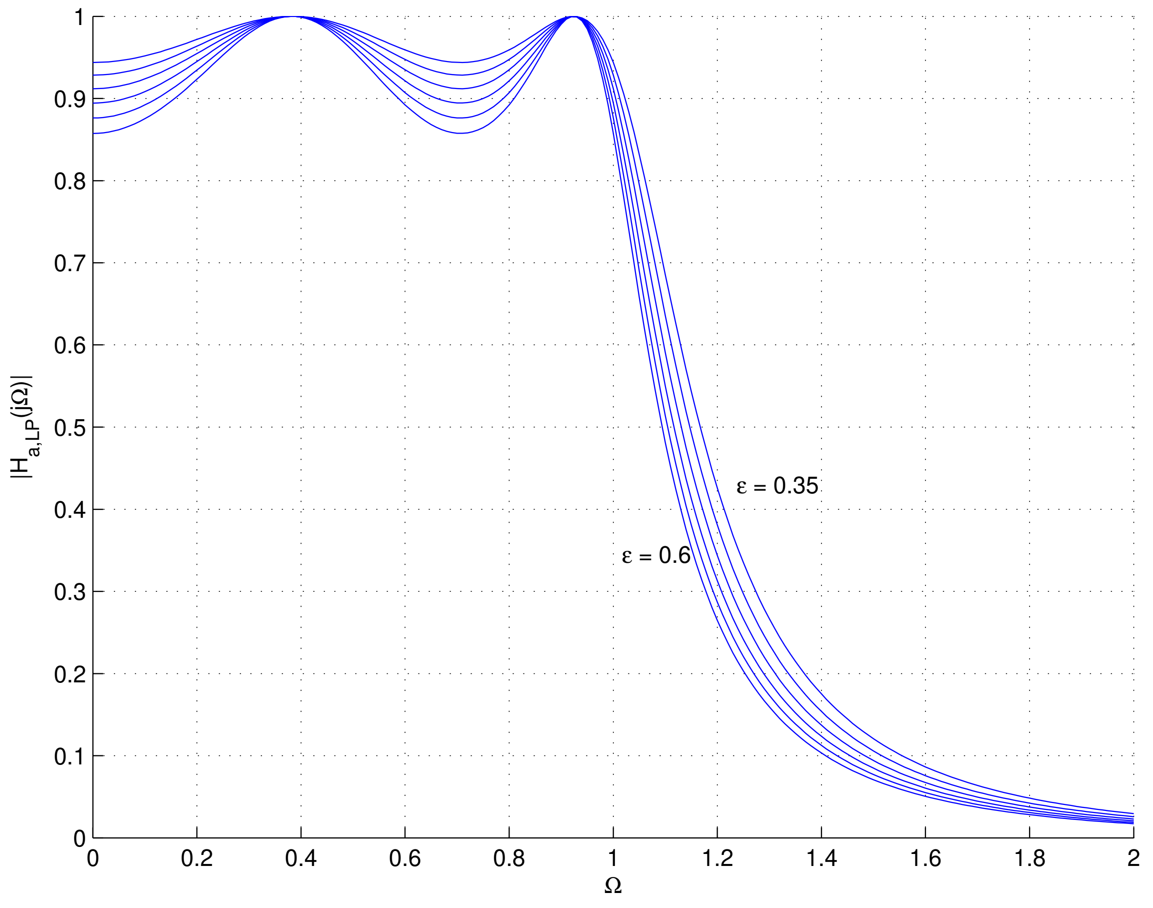


Figure 4.1: The Analog Low-Pass Frequency Response for $0.35 \le \epsilon \le 0.6$

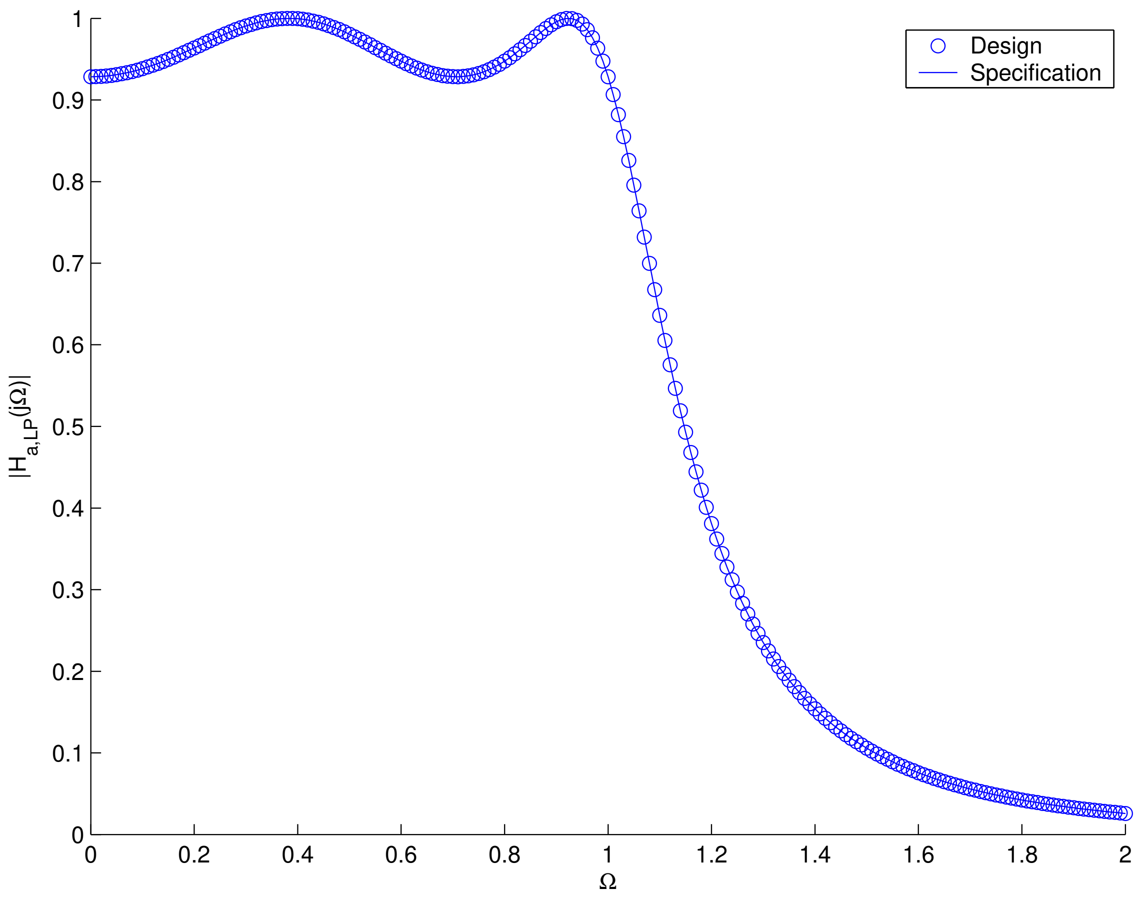


Figure 4.2: The magnitude response plots from the specifications in Equation 3.1 and the design in Equation 3.5

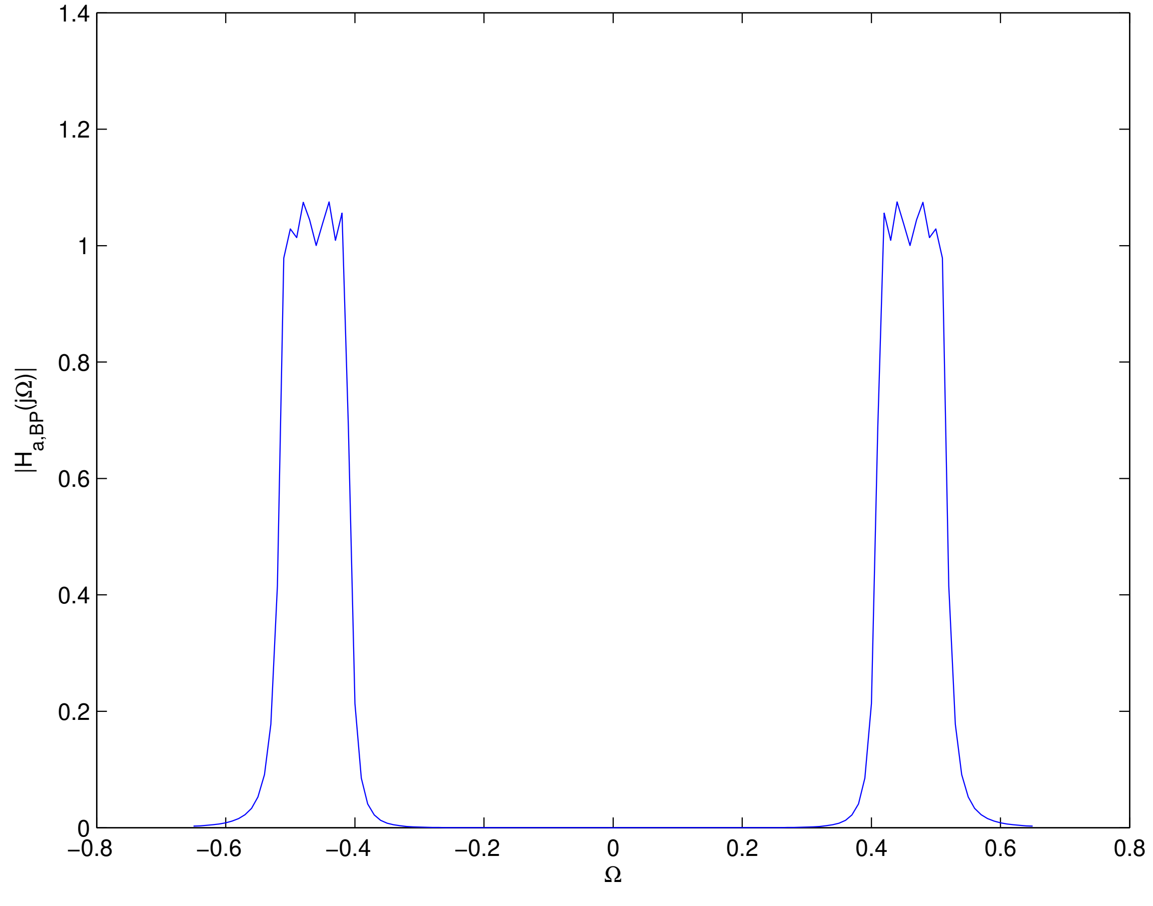


Figure 4.3: The analog bandpass magnitude response plot from Equation 4.2

$$N(z) = 1 - 4z^{-2} + 6z^{-4} - 4z^{-6} + z^{-8} \tag{4.5}$$

and

$$\begin{aligned} D(z) = 2.3609 - 12.0002z^{-1} + 31.8772z^{-2} - 53.7495z^{-3} + 62.8086z^{-4} \\ -51.4634z^{-5} + 29.2231z^{-6} - 10.5329z^{-7} + 1.9842z^{-8} \end{aligned} \tag{4.6}$$

The plot of $|H_{d,BP}(z)|$ with respect to the normalized angular freqency (normalizing factor π) is available in Fig. 4.4. Again we

find that the passband and stopband frequencies meet the specifications well enough.

B.4. The FIR Filter

We design the FIR filter by first obtaining the (non-causal) lowpass equivalent using the Kaiser window and then converting it to a causal bandpass filter.

B.4.1. The Equivalent Lowpass Filter

The lowpass filter has a passband frequency ω_l and transition band $\Delta\omega = 2\pi\frac{\Delta F}{F_s} = 0.0125\pi$. The stopband tolerance is δ.

1. The passband frequency ω_l is defined as $\omega_l = \frac{\omega_{p1}-\omega_{p2}}{2}$. Substituting the values of ω_{p1} and ω_{p2} from section 2.1, we obtain $\omega_l = 0.025\pi$.

2. The impulse response $h_{lp}(n)$ of the desired lowpass filter with cutoff frequency ω_l is

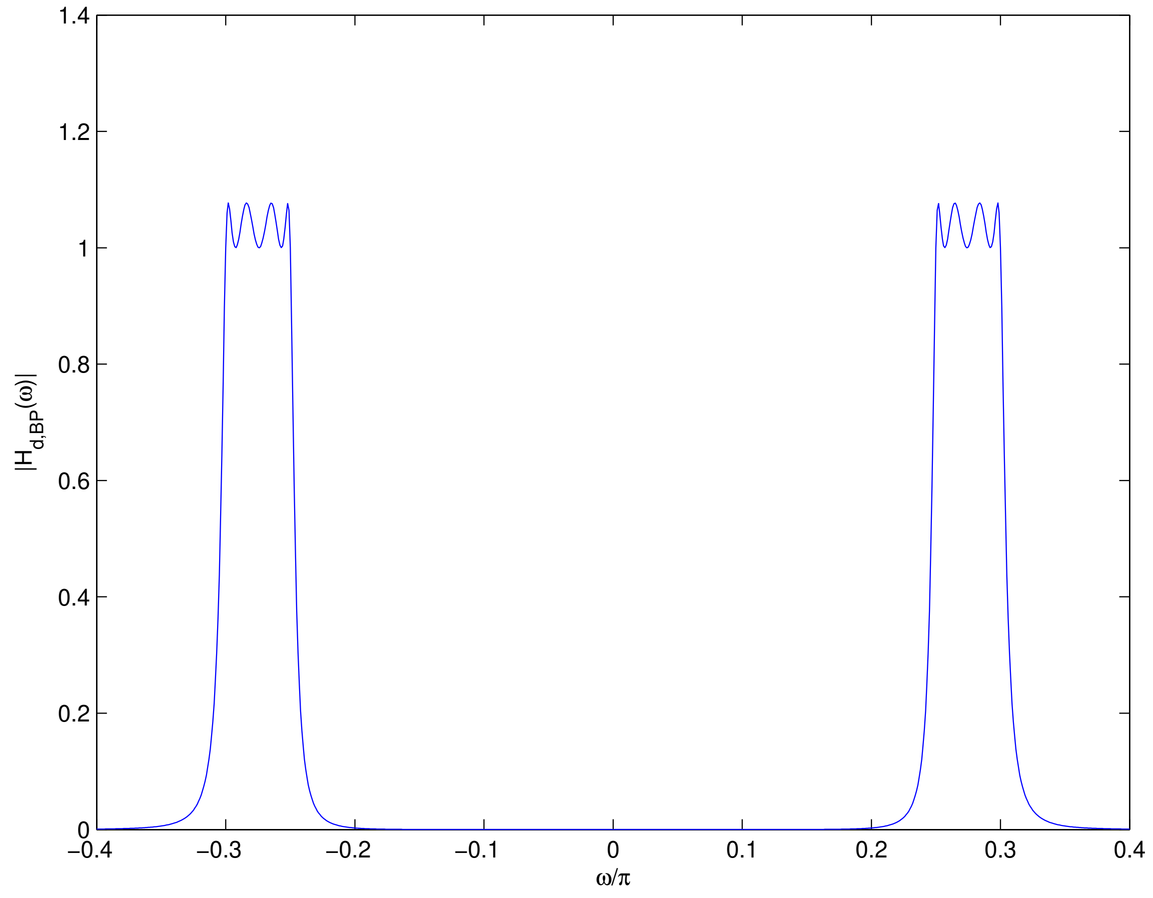


Figure 4.4: The magnitude response of the bandpass digital filter designed to meet the given specifications

given by

$$h_l(n) = \frac{\sin(n\omega_l)}{n\pi} w(n), \tag{2.1}$$

where $w(n)$ is the Kaiser window obtained from the design specifications.

B.4.2. The Kaiser Window

The Kaiser window is defined as

$$\begin{aligned} w(n) &= \frac{I_0\left[\beta N\sqrt{1-\left(\frac{n}{N}\right)^2}\right]}{I_0(\beta N)}, \qquad -N \le n \le N, \qquad \beta > 0 \\ &= 0 \qquad\qquad\qquad\qquad\qquad \text{otherwise}, \end{aligned} \tag{2.2}$$

where $I_0(x)$ is the modified Bessel function of the first kind of order zero in x and β and N are the window shaping factors. In the following, we find β and N using the design parameters in section 2.1.

1. N is chosen according to

$$N \ge \frac{A-8}{4.57\Delta\omega}, \tag{1.1}$$

   where $A = -20\log_{10}\delta$. Substituting the appropriate values from the design specifications, we obtain $A = 16.4782$ and $N \ge 48$.

2. β is chosen according to

$$\beta N = \begin{cases} 0.1102(A-8.7) & A > 50 \\ 0.5849(A-21)^{0.4} + 0.07886(A-21) & 21 \le A \le 50 \\ 0 & A < 21 \end{cases} \tag{2.1}$$

In our design, we have $A = 16.4782 < 21$. Hence, from (2.1) we obtain $\beta = 0$.

3. We choose $N = 100$, to ensure the desired low pass filter response. Substituting in (2.2) gives us the rectangular window

$$\begin{aligned} w(n) &= 1, -100 \leq n \leq 100 \\ &= 0 \quad \text{otherwise} \end{aligned} \tag{3.1}$$

From (2.1) and (3.1), we obtain the desired lowpass filter impulse response

$$\begin{aligned} h_{lp}(n) &= \frac{\sin(\frac{n\pi}{40})}{n\pi} \qquad -100 \leq n \leq 100 \\ &= 0, \qquad\qquad \text{otherwise} \end{aligned} \tag{3.2}$$

The magnitude response of the filter in (3.2) is shown in Figure 5.

B.4.3. The FIR Bandpass Filter

The centre of the passband of the desired bandpass filter was found to be $\omega_c = 0.275\pi$ in Section 2.1. The impulse response of the desired bandpass filter is obtained from the impulse response of the corresponding lowpass filter as

$$h_{bp}(n) = 2h_{lp}(n)cos(n\omega_c) \tag{3.3}$$

Thus, from (3.2), we obtain

$$\begin{aligned} h_{bp}(n) &= \frac{2\sin(\frac{n\pi}{40})\cos(\frac{11n\pi}{40})}{n\pi} \qquad -100 \leq n \leq 100 \\ &= 0, \qquad\qquad \text{otherwise} \end{aligned} \tag{3.4}$$

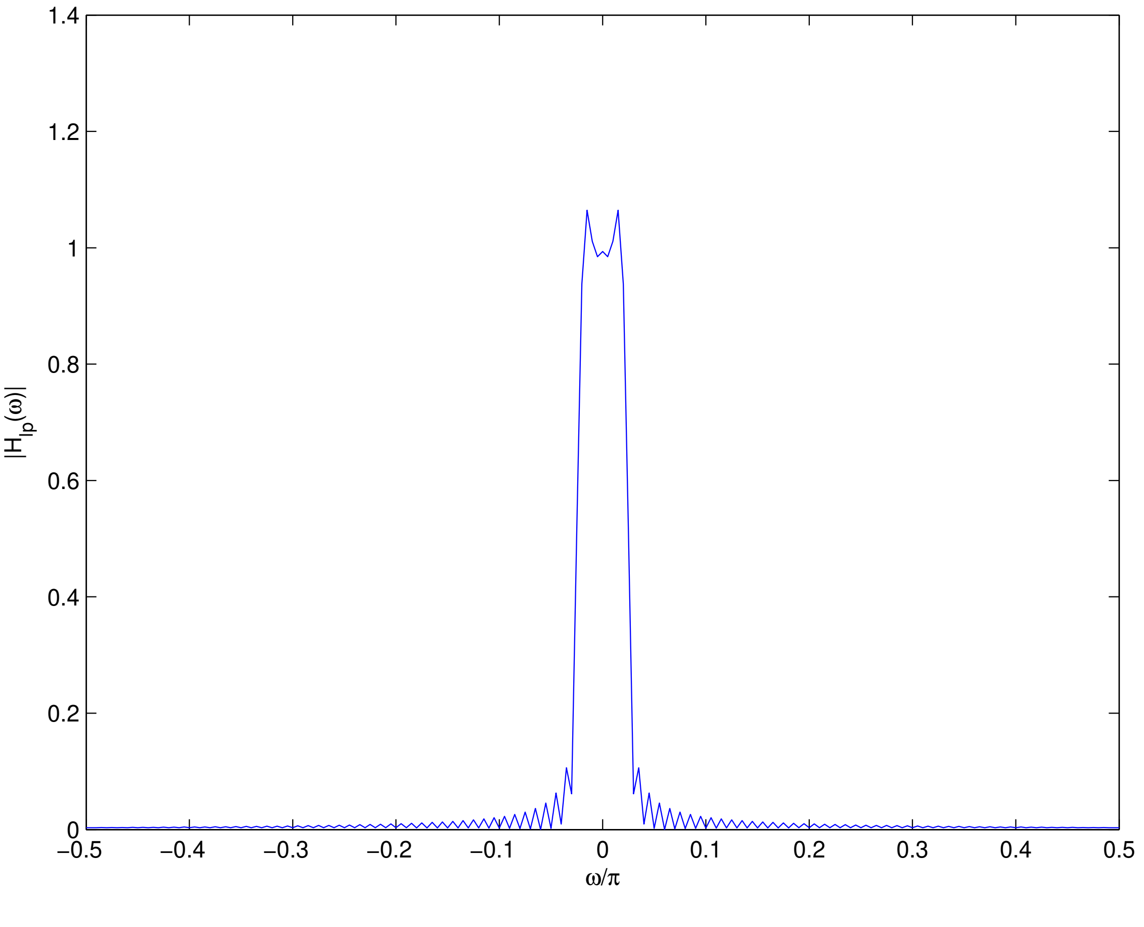


Figure 3.1: The magnitude response of the FIR lowpass digital filter designed to meet the given specifications

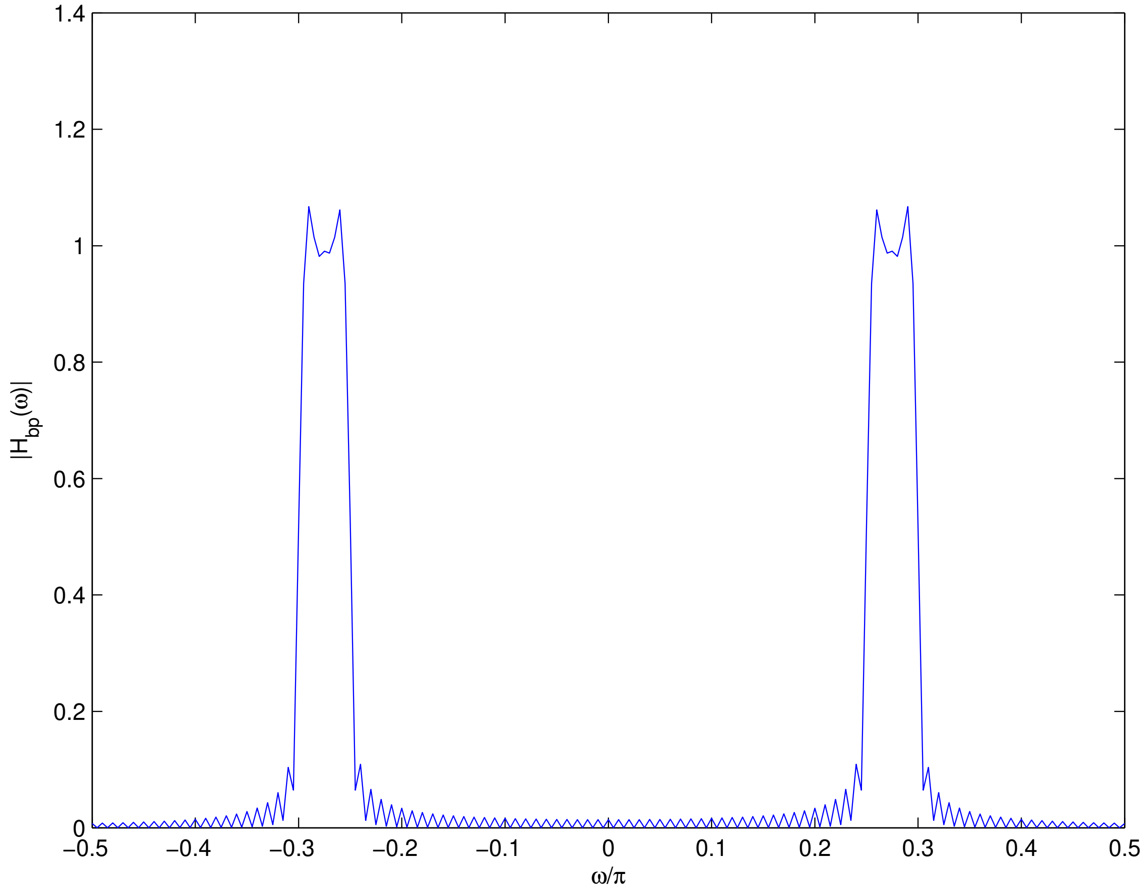


Figure 3.2: The magnitude response of the FIR bandpass digital filter designed to meet the given specifications

The magnitude response of the FIR bandpass filter designed to meet the given specifications is plotted in Fig. 3.2.