Lab 8 – CTFT, FFT, and Quantization

Name: K Sri Rama Rathan Reddy - 2022102072

Team Mate: B Karthikeya - 2022102042

Team: Noicifiers

8.1 Continuous-time Fourier transform:

a.

```
function X = continuousFT(t, xt, a, b, w)
    X = zeros(size(w));
    for i = 1:length(w)
        X(i) = int(xt*exp(-1j*w(i)*t), t, a, b);
    end
end
```

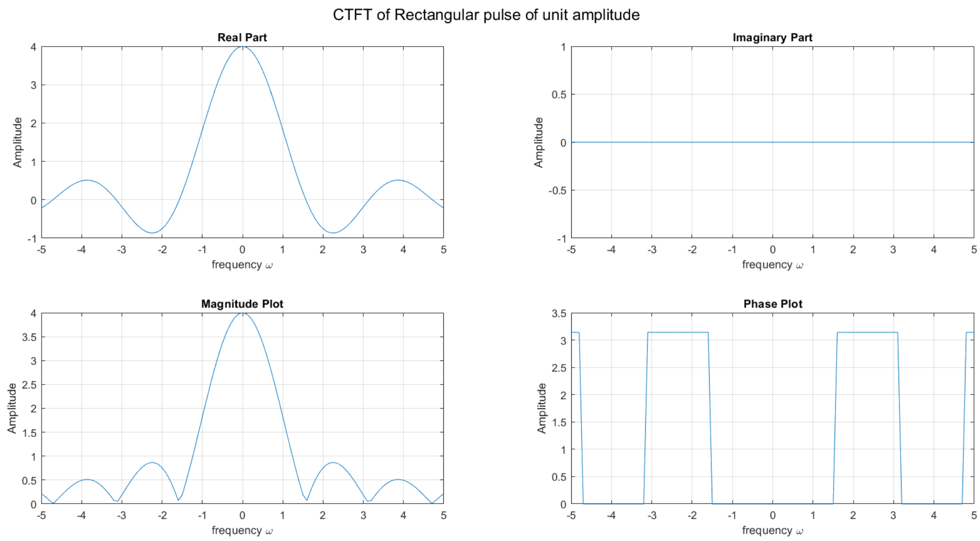
Inputs:

- **t –** symbolic variable
- **xt –** signal whose FT is to be computed (function of symbolic variable t)
- **a, b –** the signal is equal to xt in the interval [a, b] and zero outside
- **ω –** the vector ω contains the values of frequency where FT is to be computed.

Outputs:

- **X -** contains the FT of $x(t)$ for each of the frequencies in the input vector ω.

b.



Continuous Fourier Transform for Rectangular Pulse from [-2,2].

Explanation for the above plot:

$$x(t) = \begin{cases} 1 ; & -T \le t \le T \\ 0 ; & \text{otherwise} \end{cases}$$

$$X(\omega) = \int_{-\infty}^{\infty} x(t)\, e^{-j\omega t}\, dt = \int_{-T}^{T} e^{-j\omega t}\, dt = \frac{e^{-j\omega T} - e^{+j\omega T}}{-j\omega} = \frac{e^{j\omega T} - e^{-j\omega T}}{j\omega}$$

$$\Rightarrow X(\omega) = \frac{2 \sin(\omega T)}{\omega} = \underline{\underline{2T \cdot \operatorname{sinc}(\omega T)}}$$

$$\therefore \boxed{X(\omega) = 2T \cdot \operatorname{sinc}(\omega T)}$$

- **Real Part:**
  - It would be the Sinc function.
- **Imaginary Part:**

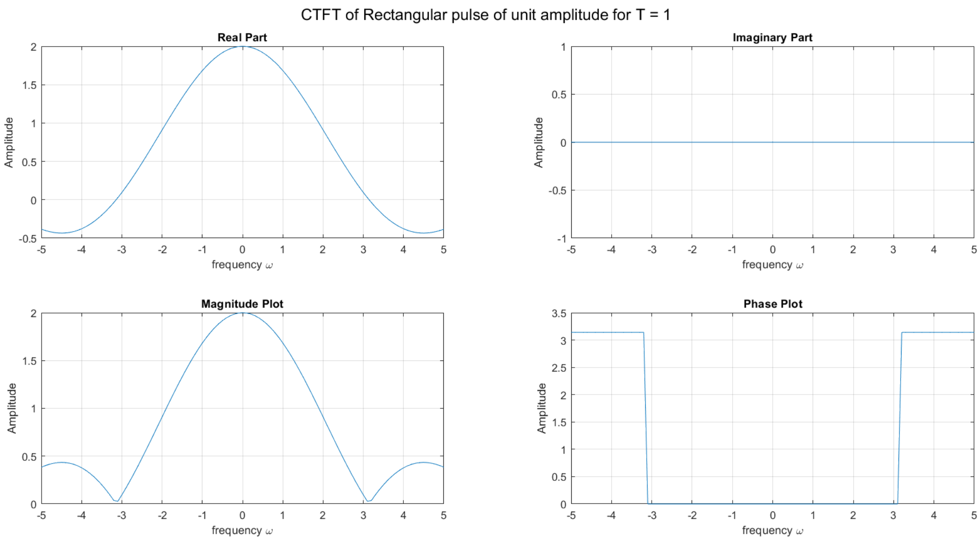
- All the terms are Real so the imaginary part would be 0

- **Magnitude Plot:**
  - $|x_i| = \sqrt{Re(x_i^2) + Im(x_i^2)}$

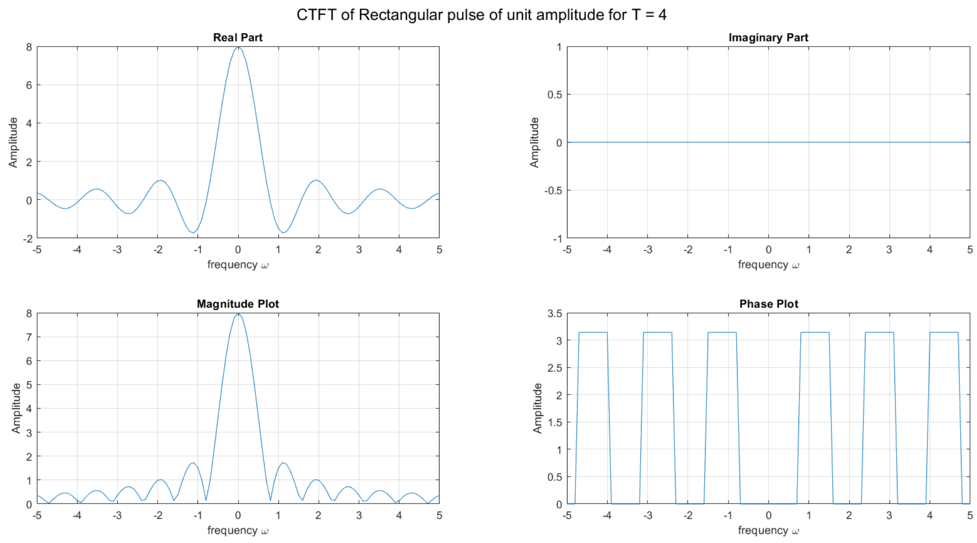
- **Phase Plot:**
  - The Phase would be π if x(i) = 1 and 0 if x(i) = 0 or π when ω = $4n\pi/2\ to\ 4n+1\pi/2\ and 4n+2\pi/2\ to\ 4n+3\pi/2$ and 1 when $\omega = 4n+1\pi/2\ to\ 4n+2\pi/2$

c.

T = 1



T = 4

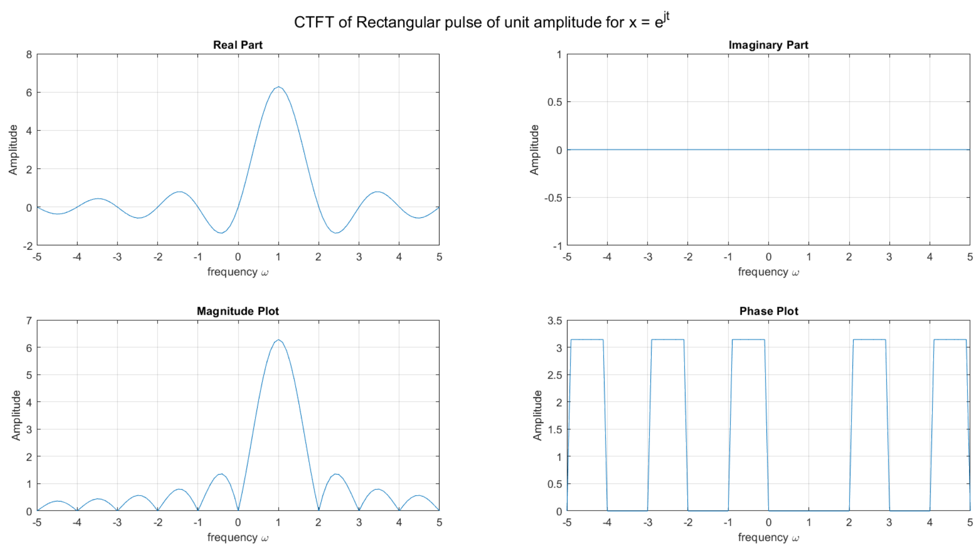


Explanation:

- We can see the the sinc function got shrunk
- The graph is Scaled by 1/4.
- Time scaling property can be observed: $x(at) = \frac{1}{|a|}X(\frac{f}{|a|})$

d.

For $x(t) = e^{jt}$:



$$x(t) = e^{jt}$$

$$T = [-\pi, \pi]$$

$$X(\omega) = \int_{-\infty}^{\infty} x(t)\, e^{-j\omega t}\, dt$$

$$= \int_{-\pi}^{\pi} e^{jt}\, e^{-j\omega t}\, dt$$

$$= \int_{-\pi}^{\pi} e^{-jt(\omega-1)}\, dt$$

$$= \left. \frac{e^{-jt(\omega-1)}}{-j(\omega-1)} \right|_{-\pi}^{\pi}$$

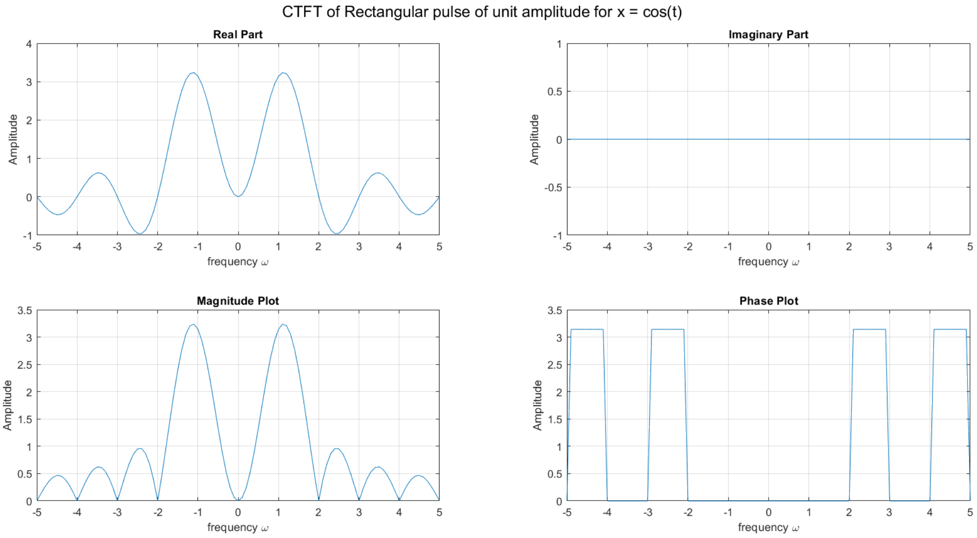
$$= \frac{e^{j\pi(\omega-1)} - e^{-j\pi(\omega-1)}}{j(\omega-1)}$$

$$= 2\pi \, \frac{\sin \pi(\omega-1)}{\pi(\omega-1)}$$

$$= 2\pi \, \operatorname{sinc}(\pi(\omega-1))$$

$\therefore$ It would shift π units to the right.

For $x(t) = cos(t)$:


CTFT of Rectangular pulse of unit amplitude for x = cos(t)
Real Part
Imaginary Part
Magnitude Plot
Phase Plot
Amplitude
frequency ω

e.

$$x(t) = e^{jt}$$

$$T = [-\pi, \pi]$$

$$X(\omega) = \int_{-\infty}^{\infty} x(t)\, e^{-j\omega t}\, dt$$

$$= \int_{-\pi}^{\pi} e^{jt}\, e^{-j\omega t}\, dt$$

$$= \int_{-\pi}^{\pi} e^{-jt(\omega-1)}\, dt$$

$$= \left. \frac{e^{-jt(\omega-1)}}{-j(\omega-1)} \right|_{-\pi}^{\pi}$$

$$= \frac{e^{j\pi(\omega-1)} - e^{-j\pi(\omega-1)}}{j(\omega-1)}$$

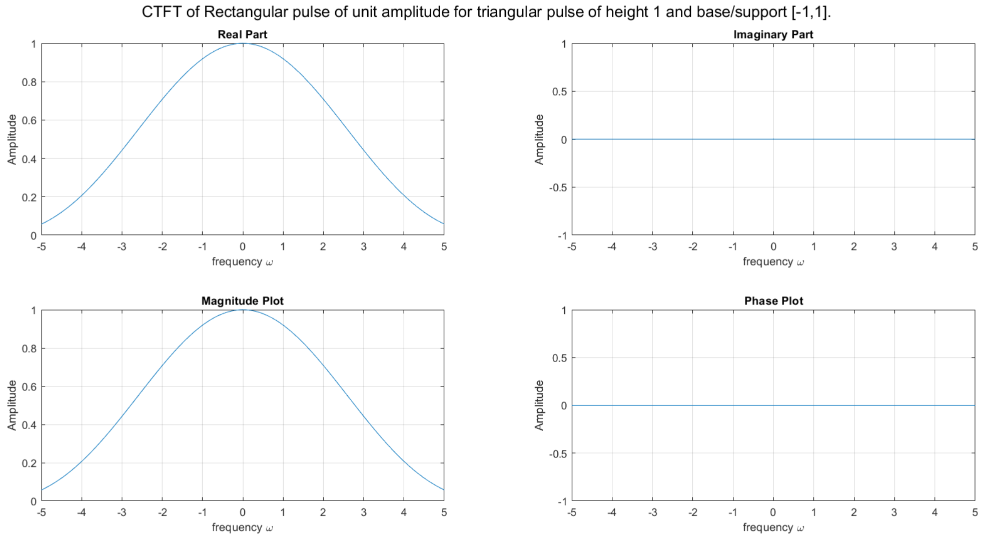
$$= 2\pi \, \frac{\sin \pi(\omega-1)}{\pi(\omega-1)}$$

$$= 2\pi \, \mathrm{sinc}\left(\pi(\omega-1)\right)$$

∴ It would shift π units to the right

e.

Triangular pulse: `xt = piecewise(-T<=t<=0,t+1,0<=t<=T,-t+1,0)`



x(t) can be expressed as convolution of two signals: $x_1(t) = x_2(t) = f(x) = \begin{cases} 1 & ; -\frac{1}{2} \leq x < \frac{1}{2} \\ 0 & ; otherwise \end{cases}$

$$CTFT\{x(t)\} = CTFT\{x_1(t) * x_2(t)\} = CTFT\{x_1(t)\} \cdot CTFT\{x_2(t)\}$$

i.e., the Convolution property of CTFT has been used to solve

Let $x_1(t) = x_2(t) = \begin{cases} 1; & -\frac{1}{2} \le t \le 1/2 \\ 0; & \text{otherwise} \end{cases}$

$$x(t) = x_1(t) * x_2(t) = \int_{-\infty}^{\infty} x_1(\tau) \cdot x_1(t-\tau) d\tau$$

$$= \int_{-1/2}^{1/2} \frac{x_1(t-\tau)}{2} d\tau$$

$$= \begin{cases} t+1; & -1 \le t \le 0 \\ -t+1; & 0 \le t \le 1 \\ 0; & \text{otherwise} \end{cases}$$

CTFT of Convolution = Convolution of product of CTT's

$$= \left(\int_{-1/2}^{1/2} e^{-jt} dt \right)^2$$

$$= \left[2(1/2) \ \text{sinc}(\omega/2) \right]^2$$

$$= \underline{\underline{\text{sinc}^2(\omega/2)}}$$

8.2 Fast Fourier Transform (Radix-2):

```
function X = radix2fft(x)
    N = length(x);
    if N == 1
            X = x;
    elseif N == 2
        X(1) = x(1)+x(2);
        X(2) = x(1)-x(2);
    else
        xe = radix2fft(x(1:2:N));
        xo = radix2fft(x(2:2:N));
        W = exp(-1j*2*pi/N).^(0:N/2-1);
        X = [xe + W.*xo, xe - W.*xo];
```

```
    end
end
```

Inputs:

- **x:** Input Vector Array for which FFT needs to be calculated

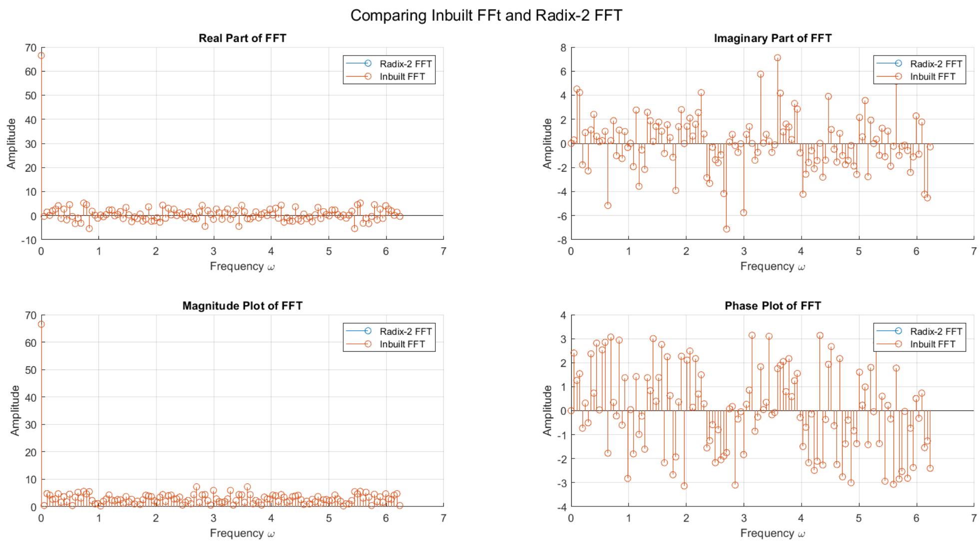
Outputs:

- **X:** Output Containing the DFT values of The input array

For N = 2:

- It is the base case of the recursion
- It has to be explicitly written in the Function as `X(1) = x(1) + x(2)` and `X(2) = x(1) - x(2)`
- i.e., the First element of FFT would be the sum of the input elements, and the second element of FFT difference between the input elements.

Verification of the Function with FFT:



FFT and Radix-2 FFT comparison plots for 128-length randomly generated sequence.

8.3 Quantization:

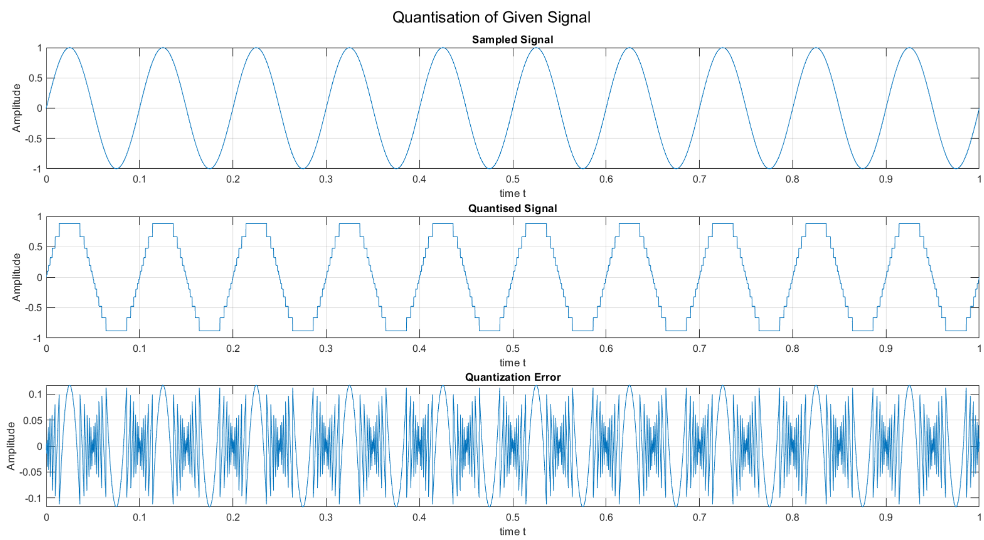
```
function y = quadratic_quant(x,B,a)
    L = 2^(B-1);
    r = linspace(0,1,L+1);
    r = r.^2;
    r = [-fliplr(r),r(2:end)];
    y = zeros(size(x));
```

```
    for i=1:length(x)
        if x(i)>=a*r(end)
            y(i) = a;
        elseif x(i)<a*r(1)
            y(i) = -a;
        else
            for j=2:length(r)
                if x(i)>=a*r(j-1) && x(i)<a*r(j)
                    y(i) = (a*(r(j-1)+r(j)))/2;
                    break;
                end
            end
        end
    end
end
```

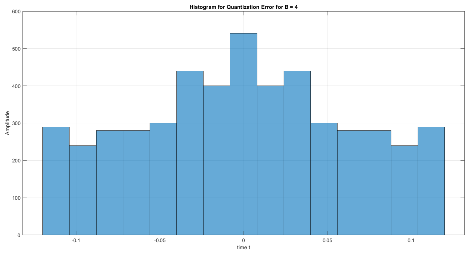
Explanation:

- `y = linspace(x1,x2,n)` generates `n` points. The spacing between the points is `(x2-x1)/(n-1)`.
- Each element of the array is squared to get the Quadratic separation.
- The array is flipped and concatenated properly to extend it even to negative numbers.
- For every element in x(t), the value has been checked for its range and the average value of the extrema is assigned to it and stored in its respective place in y(t)

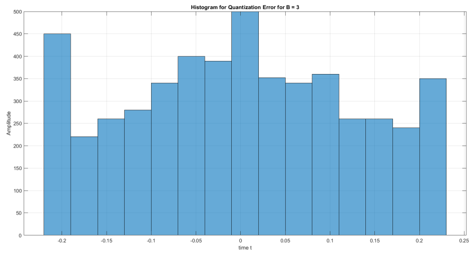
a,b,c:



d.

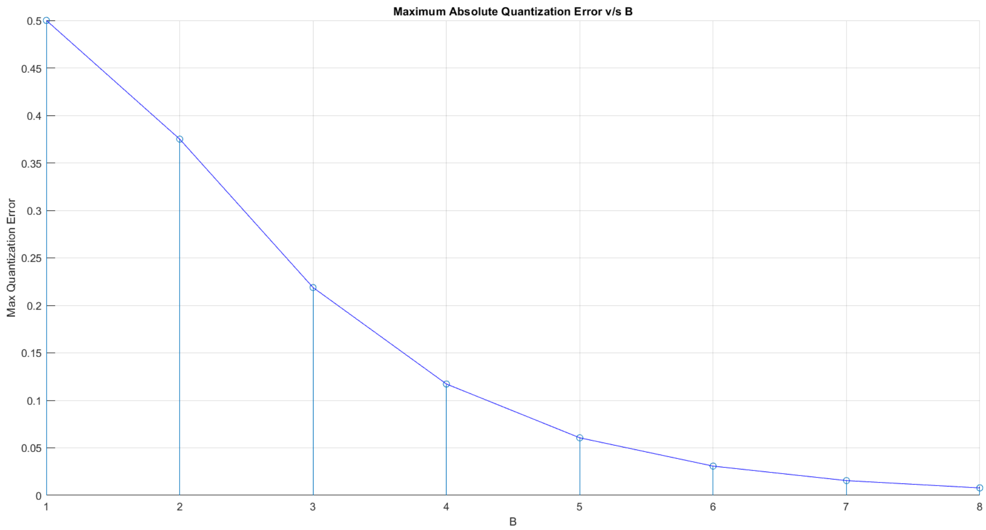


for B = 4



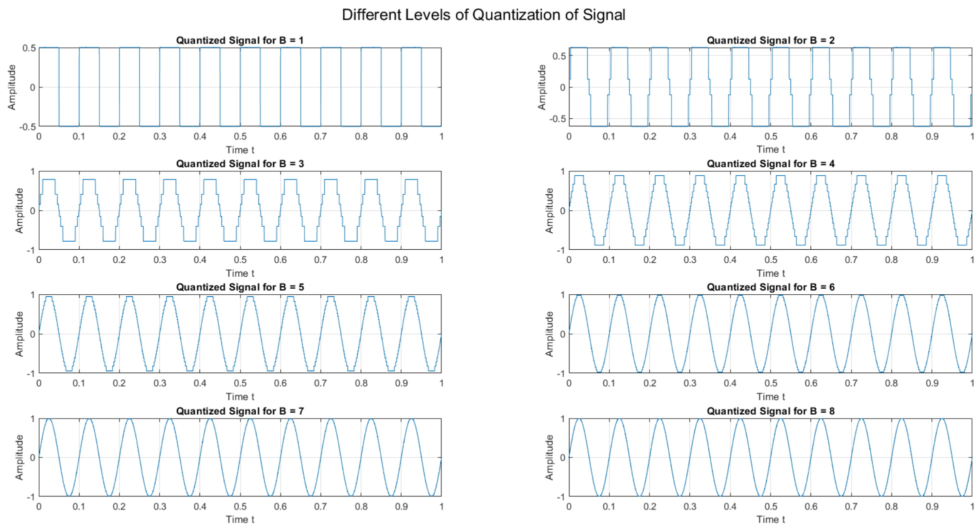
for B = 3

e.

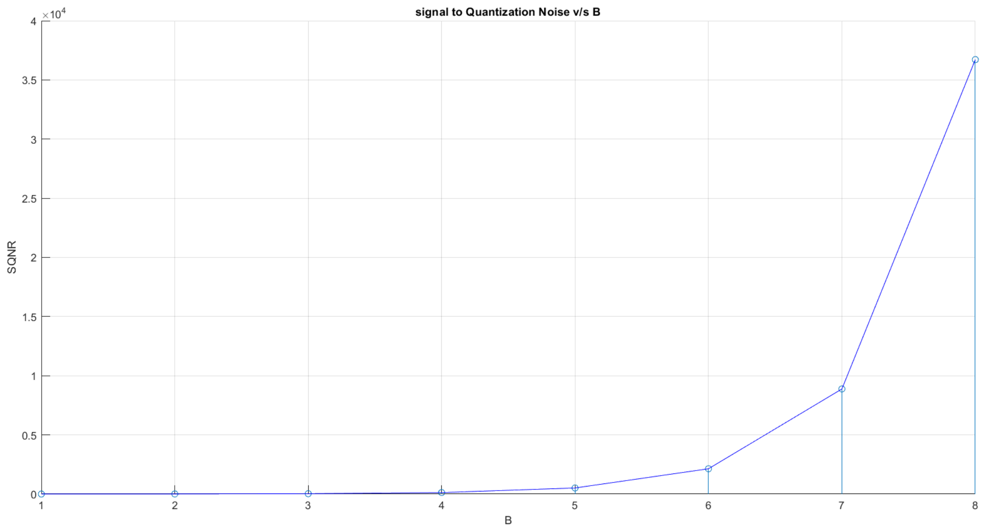


- As B increases, No. of levels of Quantization increases.
- The Signal would become closer to the original one.
- So, Quantization Error error decreases.

Different Levels of Quantization:



f.



Observations:

- As B increases, No. of Quantization levels increase.
- The Signal becomes closer to the original one.
- Therefore noise decreases.
- So, Power corresponding to noise decreases.
- But the Power of the Signal remains the same.
- So, **SQNR increases with B.**

g.

In the intervals nearer to 0, the accuracy is high but as the interval lies far away from the origin, the accuracy decreases.

For a uniform quantizer, every interval is of the same length.

I feel that a Quadratic Quantizer is better than a linear Quantizer because at least for some intervals nearer to zero, the error would be close to zero unlike the linear one.

8.4 Quantization of Audio Signals:

```
[y,fs] = audioread("Audio Files\9.wav");
t = 1:fs:(length(y)/fs);
B = 3;
a = 1;
yq = quadratic_quant(y,B,a);

sound(y);
pause(2);
sound(yq);
pause(2);
```

Comparing the sound quality of these two signals:

- The quality of the Original signal is better than the Quantized signal.
- Quantization adds additional unwanted Noise to the original signal which degrades the Quality of the signal.

```
b = 1:1:8;
for k = 1:1:length(b)
    yq = quadratic_quant(y,b(k),a);
    sound(yq);
    pause(2);
end
```

Observation as Quantization Level Increases:

- As the Quantization level increases, The Quantized signal becomes closer to the original one.
- So, the sound quality becomes better (close to the original one) as levels of Quantization increase.

The effect of Quantization on Frequency Content the Signal:

- The frequency of the signal does not change.
- Quantization adds extra noise to the signal but it does not change the signal.
- As B increases, Noise decreases but the frequency of the signal remains the same irrespective of B.