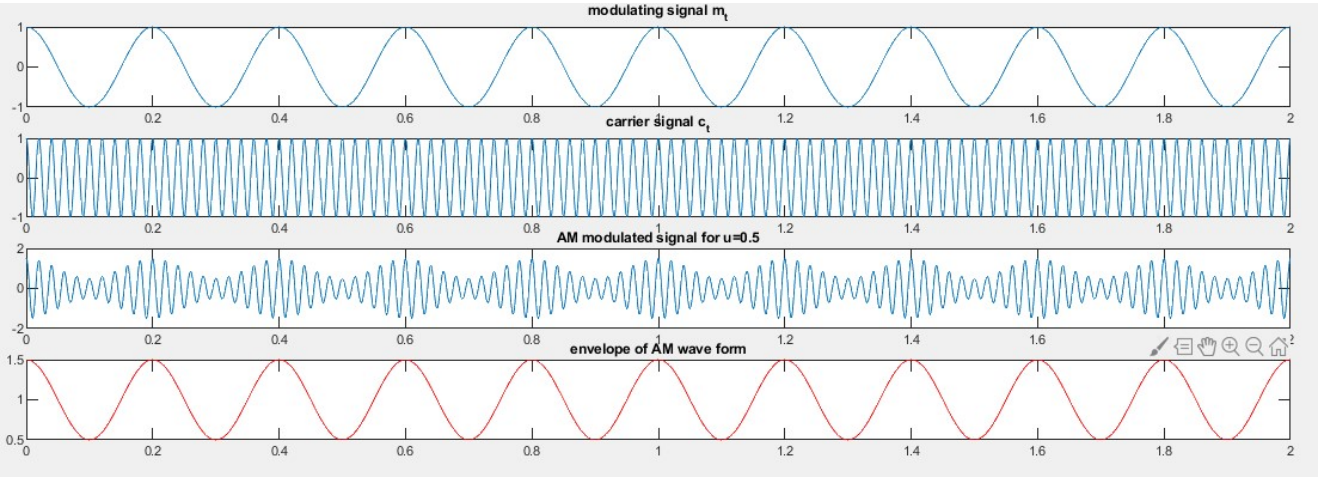
AM MODULATION-DEMODULATION

05 July 2021 14:55

% AIM: TO GENERATE AM SIGNAL WITH ENVELOPE

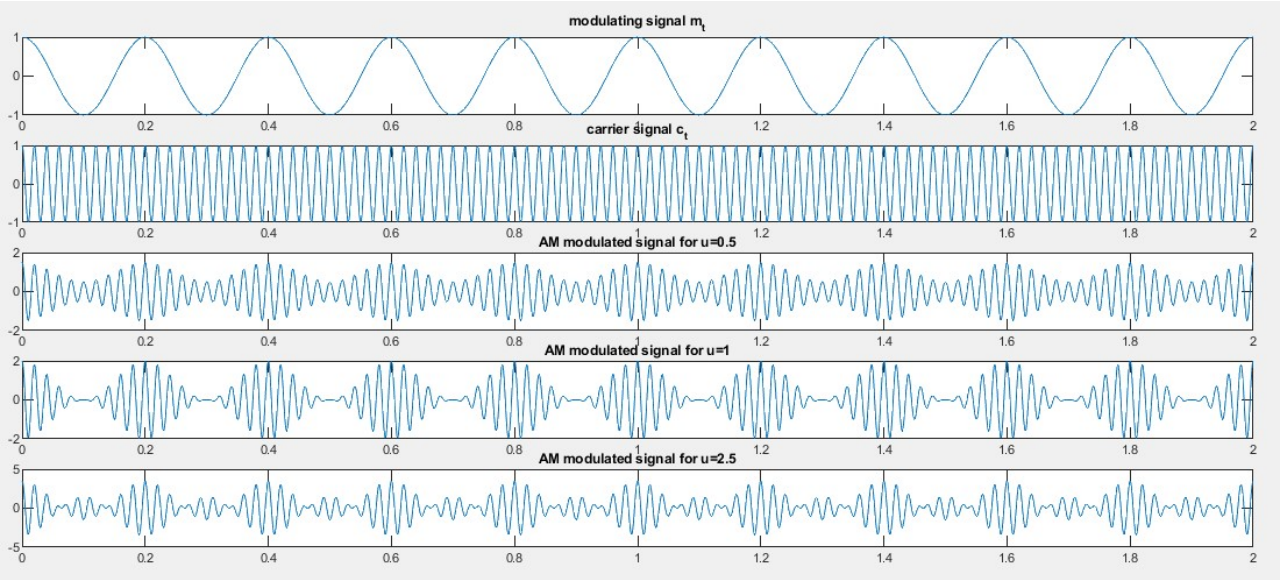
```
clc;
clear all;
close all;
t=0:0.001:2;
Am=1,fm=5;
Ac=1,fc=50;

% To plot modulating signal
m_t=Am*cos(2*pi*fm*t);
subplot(6,2,1:2)
plot(t,m_t);
title('modulating signal m_t');

% To plot carrier signal
c_t=Ac*cos(2*pi*fc*t);
subplot(6,2,3:4)
plot(t,c_t);
title('carrier signal c_t');

%To plot modulated signal
ka=0.5,u=ka*Am;
AM_SIGNAL=Ac*(1+u*cos(2*pi*fm*t)).*cos(2*pi*fc*t);
subplot(6,2,5:6);
plot(t, AM_SIGNAL);
title('AM modulated signal for u=0.5');

% ENVELOPE=abs(hilbert(AM_SIGNAL));
  ENVELOPE= Ac*(1+u*cos(2*pi*fm*t))
 subplot(6,2,7:8);
plot(t, ENVELOPE, 'r');
title('envelope of AM wave form');
```



% AIM:TO GENERATE DIFFERENT CASES OF AM

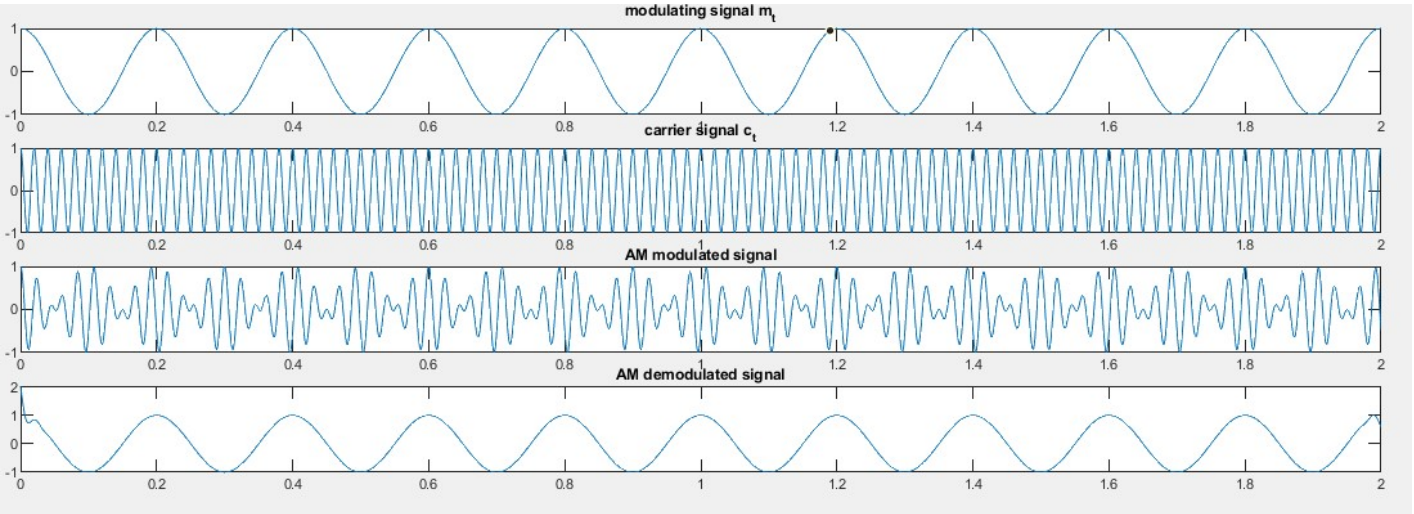
```
clc;
clear all;
close all;
t=0:0.001:2;
Am=1,fm=5;
Ac=1,fc=50;
% To plot modulating signal
m_t=Am*cos(2*pi*fm*t);
subplot(6,2,1:2)
plot(t,m_t);
title('modulating signal m_t');
% To plot carrier signal
c_t=Ac*cos(2*pi*fc*t);
subplot(6,2,3:4)
plot(t,c_t);
title('carrier signal c_t');
%To plot modulated signal
ka=0.5,u=ka*Am;
AM_SIGNAL1=Ac*(1+u*cos(2*pi*fm*t)).*cos(2*pi*fc*t);
```

```
subplot(6,2,5:6);
plot(t,AM_SIGNAL1);
title('AM modulated signal for u=0.5');
ka=0.5,Am=2,u=ka*Am;
AM_SIGNAL2=Ac*(1+u*cos(2*pi*fm*t)).*cos(2*pi*fc*t);
subplot(6,2,7:8);
plot(t,AM_SIGNAL2);
title('AM modulated signal for u=1');
ka=0.5,Am=5,u=ka*Am
AM_SIGNAL3=Ac*(1+u*cos(2*pi*fm*t)).*cos(2*pi*fc*t);
subplot(6,2,9:10);
plot(t,AM_SIGNAL3);
title('AM modulated signal for u=2.5');
```



% AIM: MODULATION AND DEMODULATION OF AM SIGNAL USING BUILT-IN FUNCTIONS

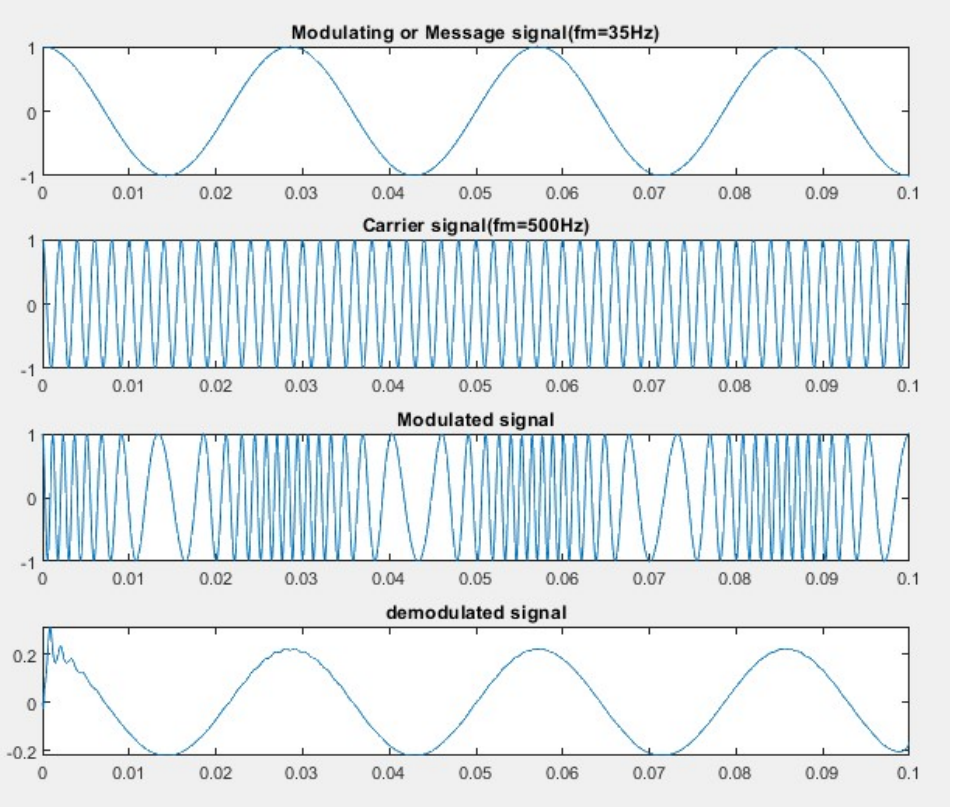
```
clc;
clear all;
close all;
t=0:0.001:2;
Am=1,fm=5;
Ac=1,fc=50;
% To plot modulating signal
m_t=Am*cos(2*pi*fm*t);
subplot(6,2,1:2)
plot(t,m_t);
title('modulating signal m_t');
% To plot carrier signal
c_t=Ac*cos(2*pi*fc*t);
subplot(6,2,3:4)
plot(t, c_t);
title('carrier signal c_t');
fs=2*(fc+(2*fm))*10;
AM_SIGNAL=ammod(m_t,fc,fs);
subplot(6,2,5:6)
plot(t, AM_SIGNAL);
title('AM modulated signal');
%To plot modulated signal
RECOVERED_SIGNAL=amdemod(AM_SIGNAL, fc,fs);
subplot(6,2,7:8);
plot(t, RECOVERED_SIGNAL);
title('AM demodulated signal');
```


modulating signal m_t
carrier signal c_t
AM modulated signal
AM demodulated signal

FM MODULATION

27 September 2021 15:06

```
close all
clear all
clc
%fm=35HZ,fc=500HZ,Am=1V,Ac=1V,B=10
fs=10000;
Ac=1;Am=1;fm=35;fc=500;B=10;
t=(0:.1*fs)/fs;

wm=2*pi*fm;
m_t=Am*cos(wm*t);
subplot(4,1,1);
plot(t,m_t);
title('Modulating or Message signal(fm=35Hz)');

wc=2*pi*fc;
c_t=Ac*cos(wc*t);
subplot(4,1,2);
plot(t,c_t);
title('Carrier signal(fm=500Hz)');

s_t=Ac*cos((wc*t)+B*sin(wm*t));
subplot(4,1,3);
plot(t,s_t);
title('Modulated signal');

d=demod(s_t,fc,fs,'fm');
subplot(4,1,4);
plot(t,d);
title('demodulated signal')
```



SAMPLING & PAM

21 July 2021 15:22

A band limited signal of finite energy which has no frequency components higher than f_m Hz, is completely described by specifying the values of the signal at instants of time separated by ½ f_m seconds.

The sampling theorem states that, if the sampling rate in any pulse modulation system exceeds twice the maximum signal frequency, the original signal can be reconstructed in the receiver with minimum distortion.

$F_s > 2f_m$ is called Nyquist rate.

Where f_s – sampling frequency

F_m – Modulation signal frequency.

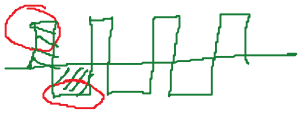
Reconstruction is possible

$f_s > 2f_m$

If we reduce the sampling frequency f_s less than f_m, the side bands and the information signal will overlap and we cannot recover the information signal simply by low pass filter. This phenomenon is called fold over distortion or aliasing.There are two methods of sampling. (1) Natural sampling (2) Flat top sampling.

$f_s < 2f_m$

Aliasing



Sample & Hold circuit holds the sample value until the next sample is taken. Sample & Hold technique is used to maintain reasonable pulse energy.The duty cycle of a signal is defined as the ratio of Pulse duration to the Pulse repetition period. The duty cycle of 50% is desirous taking the efficiency into account.

$$\text{Duty cycle} = \frac{T_{ON}}{T_{ON}+T_{OFF}} = \frac{1}{2}$$

T

When sampling frequency is greater than the nyquist rate then we can represent the discrete signal with more number of samples. These more number of samples are used to reconstruct (reproduce) the original signal without any distortion.

Sampling Theorem :

Statement : It sates that "if the highest frequency in the signal spectrum is B, the signal can be reconstructed from its samples, taken at a rate not less than 2B samples per second".

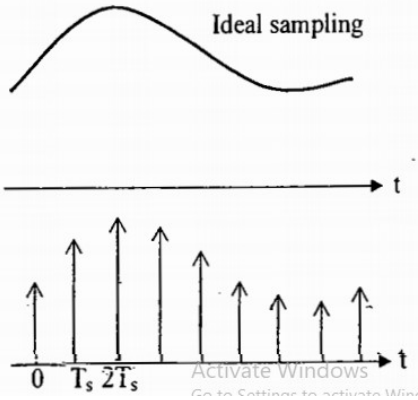
By sampling, the continuous time signal is converted into discrete time signal but the amplitude is constant.

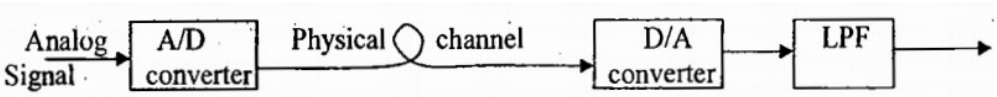
The signal is defined only at 0, T_s, $2T_s$, ...

where T_s is the sampling period.

A/D converter performs both sampling and quantization.

Disadvantage in having more number of samples is the pulse width decreases, it increases the bandwidth required to transmit the information.





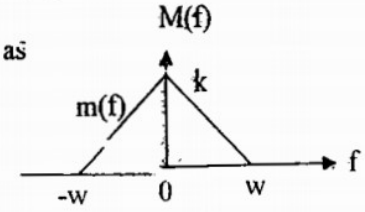
Condition on T_s to get a faithful reproduction of the input analog signal is

$$T_s \leq \frac{1}{2W} \quad ; \quad f_s \geq 2W$$

If the number of samples are more, the reconstructed signal is very close to the input signal.

$f_s = 2W$ is called the Nyquist rate.

Let us consider the spectrum as

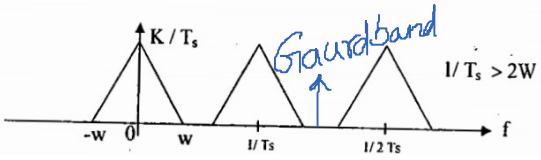


Fourier Transform of Ideally sampled signal is

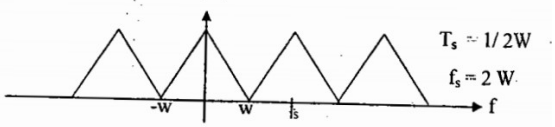
$$= (1/T_s) \sum_{n=-\infty}^{\infty} M(f - n/T_s) \qquad n = 0, \pm 1, \pm 2, \ldots\ldots$$

$$= (1/T_s)\, M(f) + (1/T_s)\, M(f - 1/T_s) + (1/T_s)\, M(f - 2/T_s) + \ldots\ldots$$

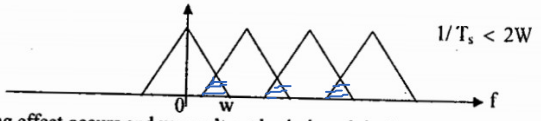
$$+ (1/T_s)\, M(f + 1/T_s) + (1/T_s)\, M(f + 2/T_s) + \ldots\ldots$$



Bandwidth of the signal after sampling is '∞'.

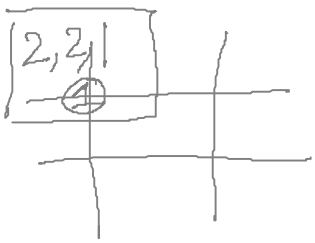


In this case, ideal LPF is used to get back the message signal.

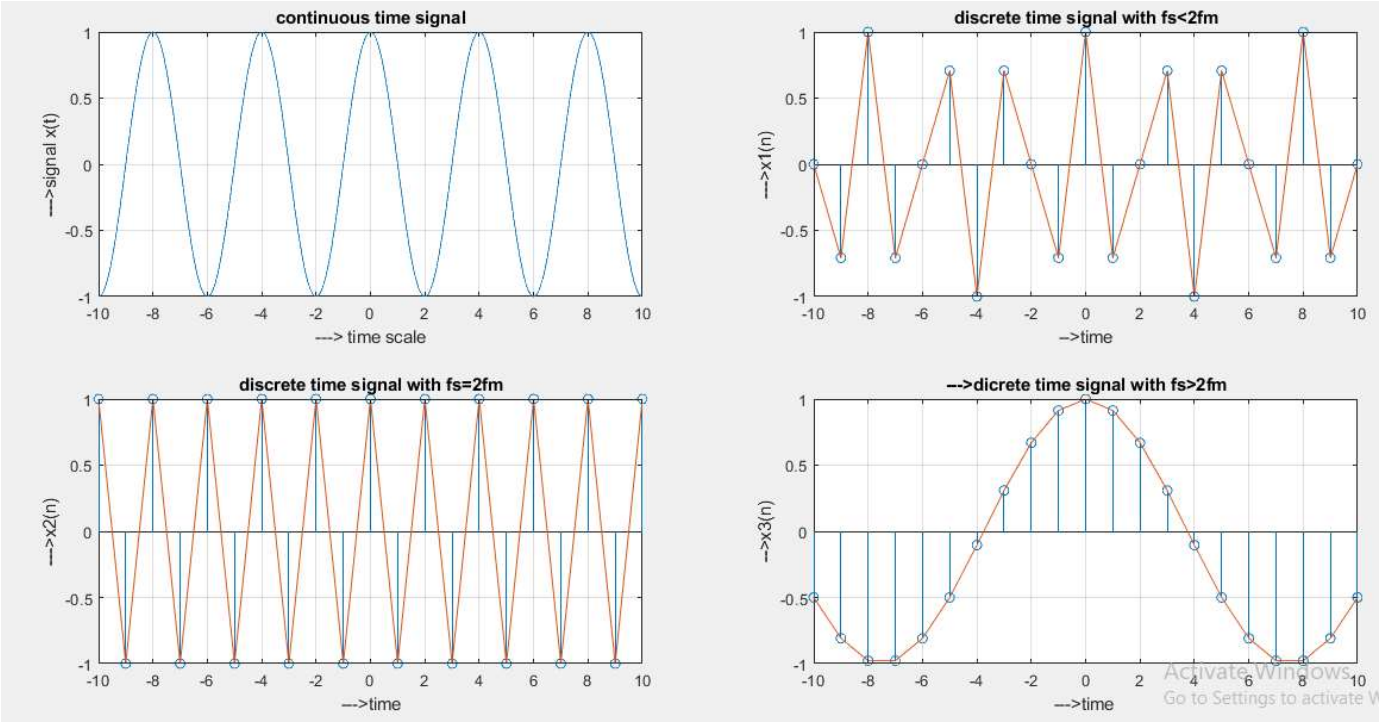


Here, Aliasing effect occurs and we can't get back the original message signal.

```
clc;
clear all;
close all;
t=-10:0.01:10;
T=4;fm=1/T;
x=cos(2*pi*fm*t);
subplot(2,2,1);
plot(t,x);
xlabel('---> time scale');
ylabel('--->signal x(t)');
title('continuous time signal');
grid

n1=-10:1:10;
fs1=1.6*fm;
x1=cos(2*pi*fm/fs1*n1);
subplot(2,2,2);
stem(n1,x1);
xlabel('-->time');
ylabel('--->x1(n)');
title('discrete time signal with fs<2fm');
hold on
subplot(2,2,2);
plot(n1,x1);
grid

n2=-10:1:10;
```

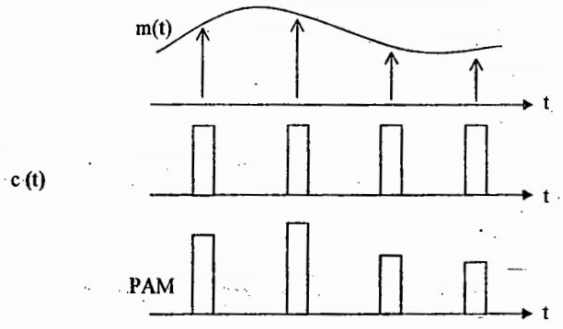


```
fs2=2*fm;
x2=cos(2*pi*fm/fs2*n2);
subplot(2,2,3);
stem(n2,x2);
xlabel('--->time');
ylabel('--->x2(n)');
title('discrete time signal with fs=2fm');
hold on
subplot(2,2,3);
plot(n2,x2);
grid

n3=-10:1:10;
fs3=15*fm;
x3=cos(2*pi*fm/fs3*n3);
subplot(2,2,4);
stem(n3,x3)
xlabel('--->time ');
ylabel('-->x3(n)');
title('--->discrete time signal with fs>2fm');
hold on
subplot(2,2,4)
plot(n3,x3);
grid
```

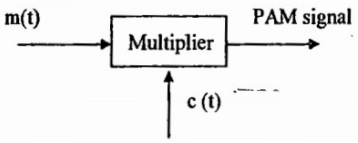


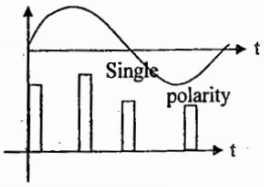
Pulse Analog Modulation (PAM) :

The carrier is high frequency periodic rectangular pulses. In PAM, the amplitude of the pulses is changed according to the sampled value.



Generation of PAM :

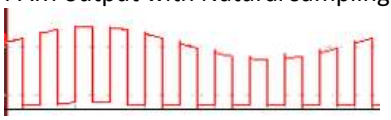




By adding a d.c. level to double polarity PAM, we can get single polarity PAM.

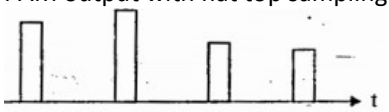
The distortion is minimum whenever the pulse width is very small. After sampling, the sampled value is held constant using sample and hold circuit.

PAM output with Natural sampling :



In natural sampling amplitude of each sample is propotional to amplitude of message signal . We can say that envelope of PAM follows the envelope of message signal.

PAM output with flat top sampling:



Flat top sampling can be generated by sample and hold circuit. Amplitude of each sample in PAM remains flat .

Pulse modulation is used to transmit analog information. In this system continuous wave forms are sampled at regular intervals. Information regarding the signal is transmitted only at the sampling times together with syncing signals.

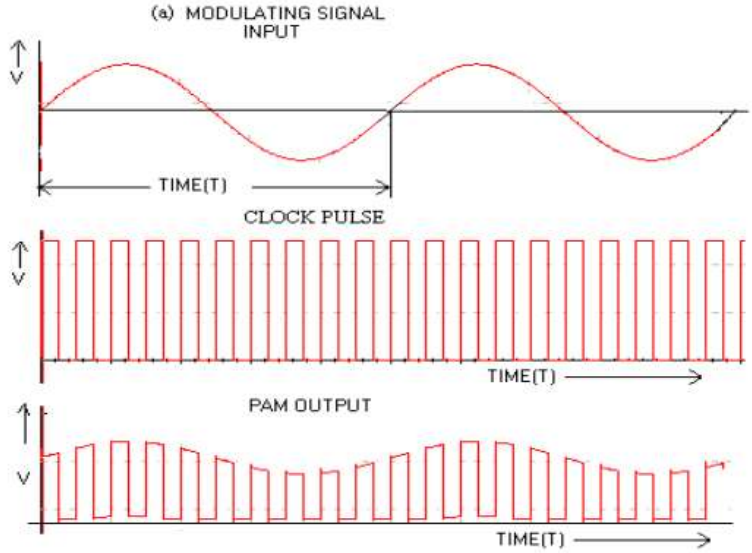
At the receiving end, the original waveforms may be reconstituted from the information regarding the samples.

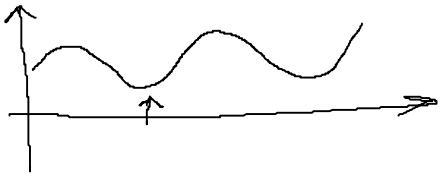
The pulse amplitude modulation is the simplest form of the pulse modulation. PAM is a pulse modulation system is which the signal is sampled at regular intervals, and each sample is made proportional to the amplitude of the signal at the instant of sampling. The pulses are then sent by either wire or cables are used to modulated carrier.

The two types of PAM are i) Double polarity PAM, and ii) the single polarity PAM, in which a fixed dc level is added to the signal to ensure that the pulses are always positive. Instantaneous PAM sampling occurs if the pulses used in the modulator are infinitely short.

Natural PAM sampling occurs when finite-width pulses are used in the modulator, but the tops of the pulses are forced to follow the modulating waveform.

Flat-topped sampling is a system quite often used because of the ease of generating the modulated wave.

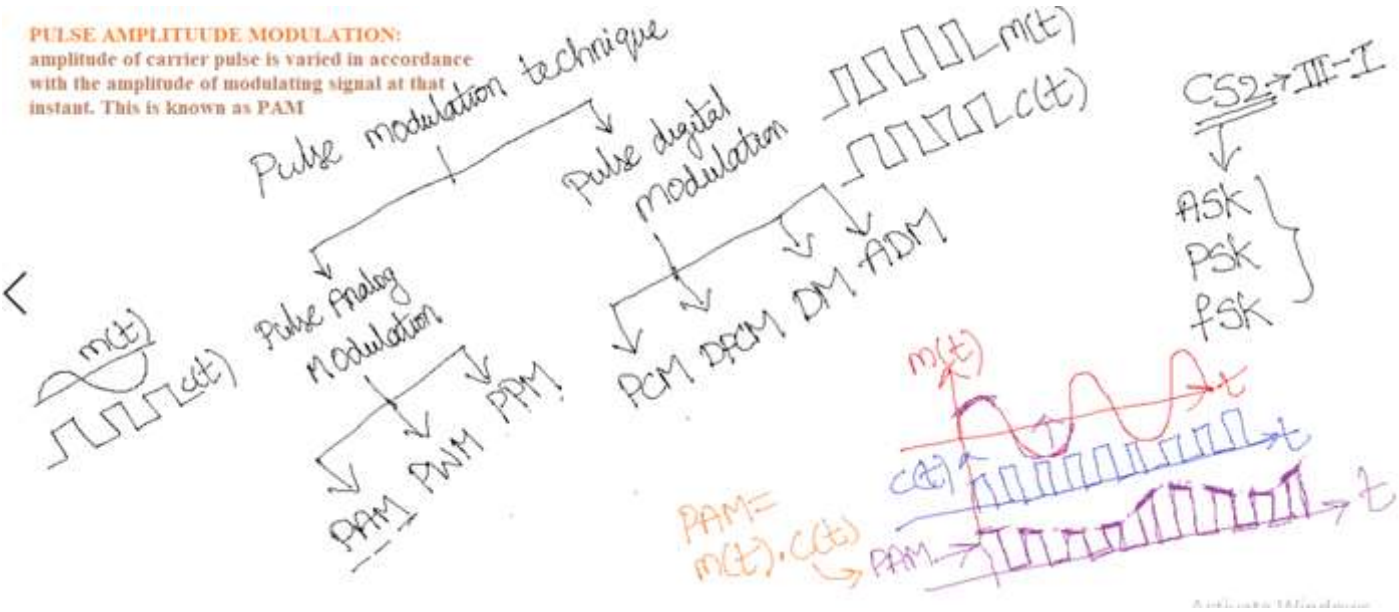




$m(t) \cdot c(t) \rightarrow PAM$

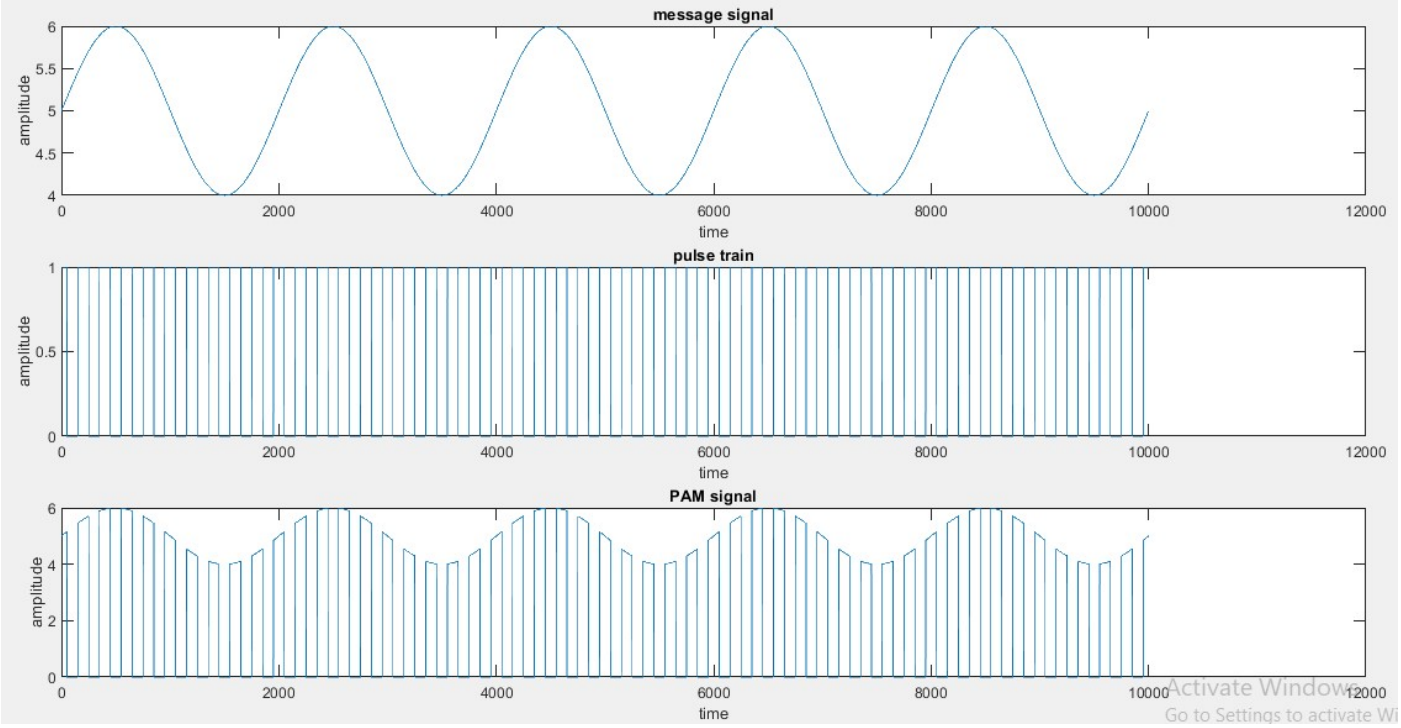
PAM signals are very rarely used for transmission purposes directly. The reason for this lies in the fact that the modulating information is contained in the amplitude factor of the pulses, which can be easily distorted during transmission by noise, crosstalk, other forms of distortion. They are used frequently as an intermediate step in other pulse-modulating methods, especially where time-division multiplexing is used.

PULSE AMPLITUUDE MODULATION:
amplitude of carrier pulse is varied in accordance with the amplitude of modulating signal at that instant. This is known as PAM



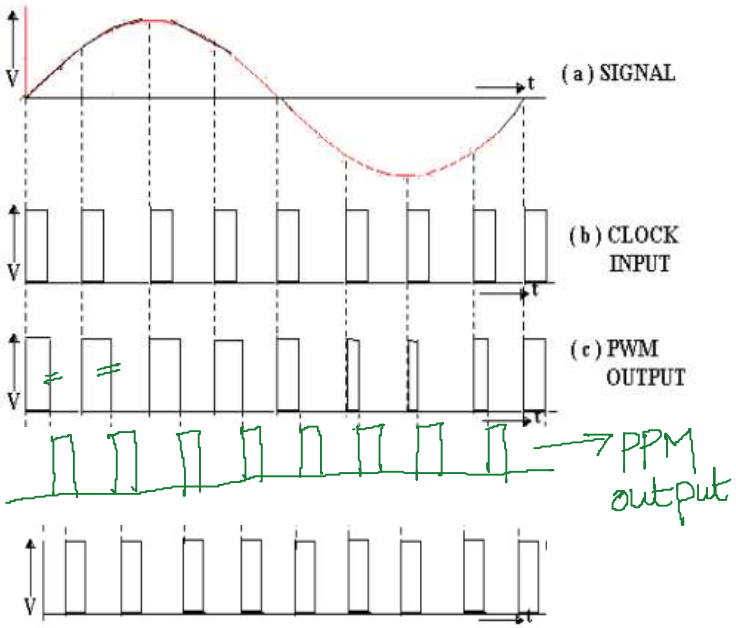

```
clc;
clear all;
close all;
fs=1000;fm=0.5;
t=0:1/fs:10;
d=0:1/5:10;
x=5+sin(2*pi*fm*t);  % message signal is shifted upwards from origin by the
                     % addition of DC component
subplot(3,1,1)
plot(x);
xlabel('time');
ylabel('amplitude');
title('message signal');

y=pulstran(t,d,'rectpuls',0.1);  %generation of pulse train carrier input
```

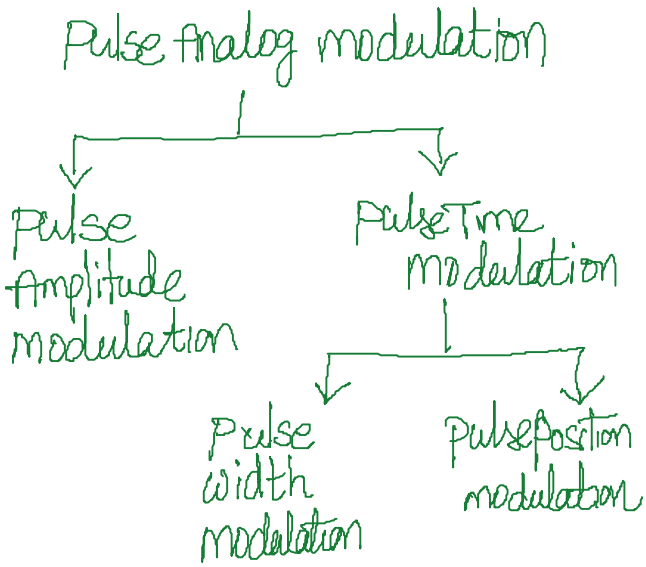
```
title( message signal );

y=pulstran(t,d,'rectpuls',0.1);  %generation of pulse train carrier input
subplot(3,1,2);
plot(y);
xlabel('time');
ylabel('amplitude');
title('pulse train');

z=x.*y;  %PAM output
subplot(3,1,3);
plot(z);
xlabel('time');
ylabel('amplitude');
title('PAM signal');
```



PWM SIMULATION

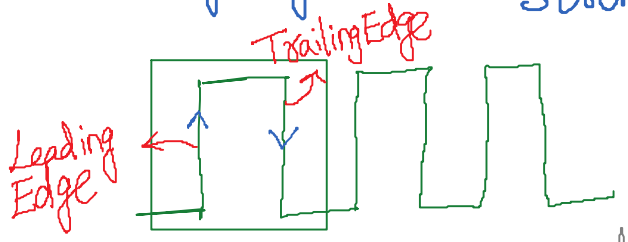
27 July 2021 14:53





PWM: width of each pulse is made propotional to amplitude of a signal at that instant

PWM: Amplitude Varying message signal → convert into → Square Wave (Constant Amplitude, Frequency) width of pulse changes as per strength of message signal.



PPM: Position of pulse signal is Varied with respect to amplitude of message signal

In PWM, Leading Edge is Fixed; Trailing is not fixed. Trailing Edges of PWM pulses are position Modulated, to get PPM.

modulate

Modulation for communications simulation

Syntax

```
y = modulate(x,fc,fs,'method')
```

Leading edge correspond to rise of pulse or positive going pulse.
Trailing edge correspond to fall of pulse or negative going pulse.

Leading edge correspond to rise of pulse or positive going pulse.
Trailing edge correspond to fall of pulse or negative going pulse.
PPM signal will start at trailing edge pulse of PWM.

Syntax

```
y = modulate(x,fc,fs,'method')
```

m(t) PWM

demod

Demodulation for communications simulation

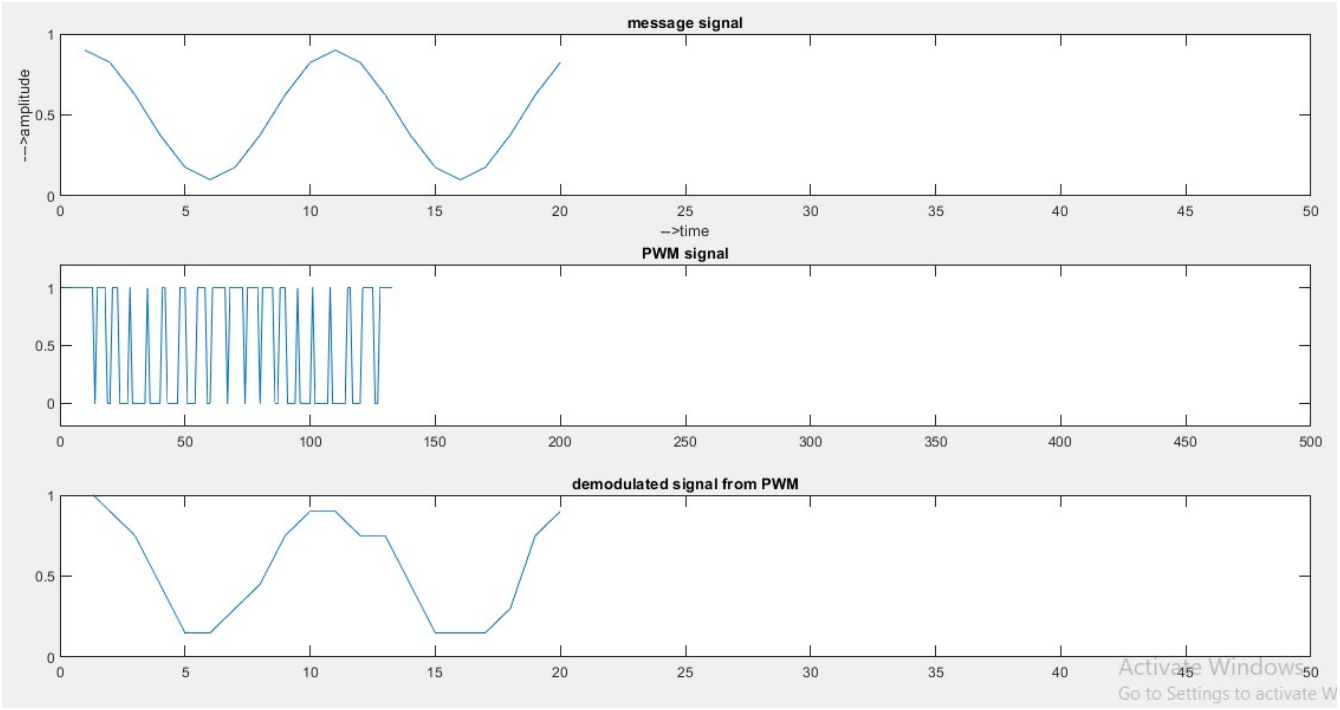
Syntax

```
x = demod(y,fc,fs,'method')
```

modulated PWM

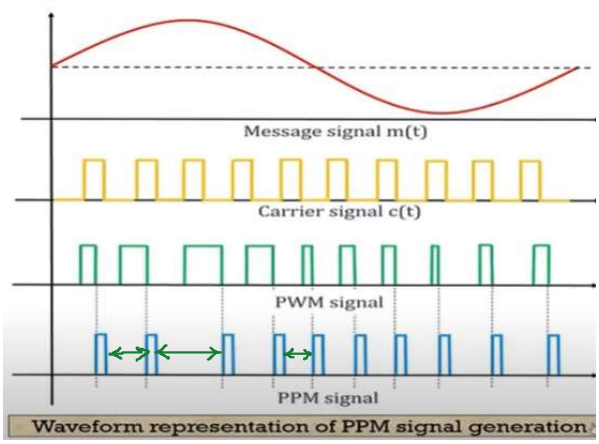
```
% MATLAB PROGRAM TO GENERATE PWM & RECONSTRUCT MESSAGE SIGNAL
clc;
clear all;
close all;

fc=300;
fs=2000;
fm=200;
t=0:1/fs:((2/fm)-(1/fs));

m_t=0.4*cos(2*pi*fm*t)+0.5;
subplot(3,1,1);
plot(m_t);
xlabel('-->time');
ylabel('--->amplitude');
title('message signal');
axis([0 50 0 1]);

MODULATED_SIGNAL=modulate(m_t,fc,fs,'pwm');
subplot(3,1,2);
plot(MODULATED_SIGNAL);
axis([0 500 -0.2 1.2]);
title('PWM signal');

RECONSTRUCTED_SIGNAL=demod(MODULATED_SIGNAL,fc,fs,'pwm');
subplot(3,1,3);
plot(RECONSTRUCTED_SIGNAL);
title('demodulated signal from PWM');
axis([0 50 0 1 ]);
```

SIMULATED OUTPUT:



PPM SIMULATION

03 August 2021 10:28



Waveform representation of PPM signal generation

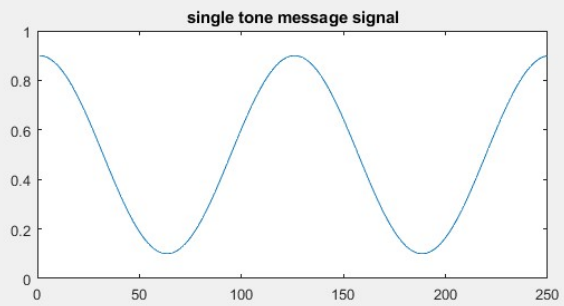
- In pulse width modulation the width of the carrier pulse is being varied as per the amplitude variation in message signal.
- In pulse position modulation the position of the carrier pulse is being varied as per the amplitude variation in message signal.

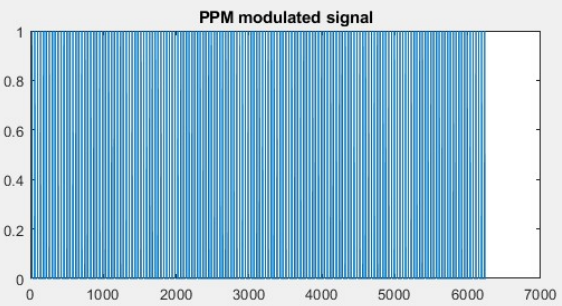
AT THE FALLING EDGE(TRAILING EDGE) OF PWM , PPM SIGNAL IS START.

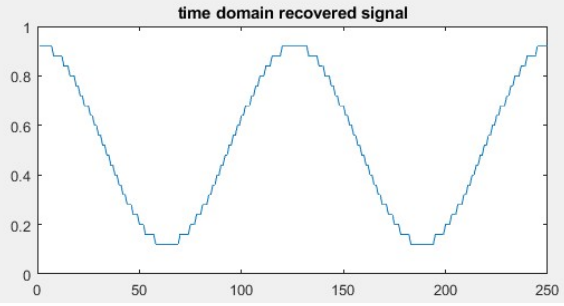
spacing between the pulses in PPM, is not same;
Position of Pulse varies according to message signal strength(amplitude)

%MATLAB PROGRAM TO SIMULATE PPM

```
clc;
close all;
clear all;

fc=1000;
fs=25000;
fm=200;
t=0:1/fs:((2/fm)-(1/fs));

x1=0.4*cos(2*pi*fm*t)+0.5;
y1=modulate(x1,fc,fs,'ppm');
subplot(2,2,1);
plot(x1);
title('single tone message signal');

 subplot(2,2,2)
 plot(y1);
 title('PPM modulated signal');

 x1_recover=demod(y1,fc,fs,'ppm')
 subplot(2,2,3)
 plot(x1_recover);
 title('time domain recovered signal');
```

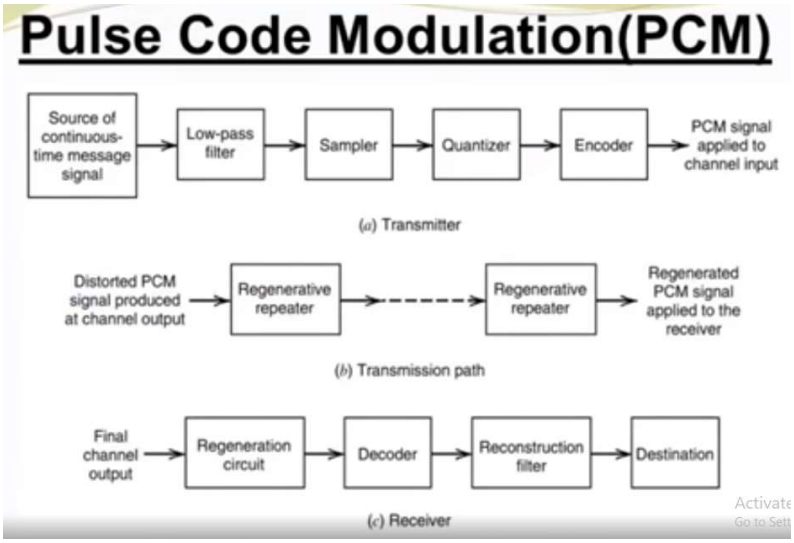






PCM SIMULATION

03 August 2021 17:38



BASIC ELEMENTS OF PCM

PCM digitizes the continuous time signal.

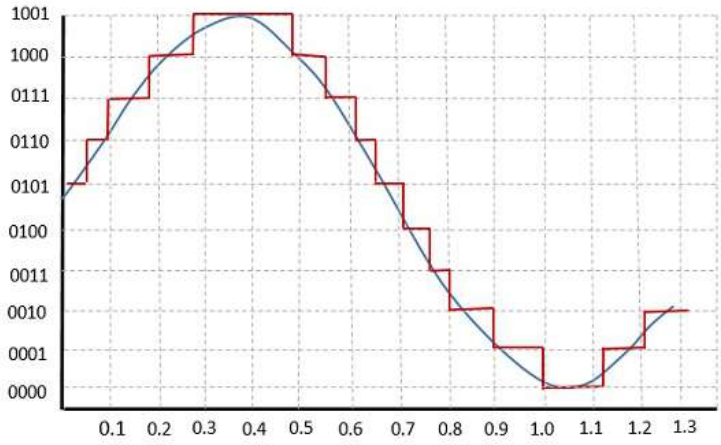
Message signal is compressed using compression laws (U-Law , A-Law)
U=255 , A-78.5

Encoding operation is done after quantizing at the transmitter.

At the receiver signal is decoded.

Decoded signal is expanded using compander technique.

The following figure shows how an analog signal gets quantized. The blue line represents analog signal while the brown one represents the quantized signal.



Both sampling and quantization result in the loss of information. The quality of a Quantizer output depends upon the number of quantization levels used. The discrete amplitudes of the quantized output are called as **representation levels** or **reconstruction levels**. The spacing between the two adjacent representation levels is called a **quantum** or **step-size**.

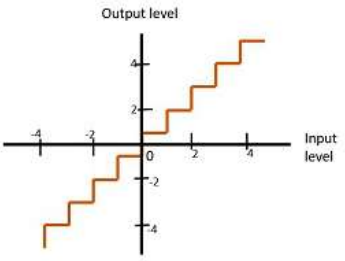


Fig 1 : Mid-Rise type Uniform Quantization

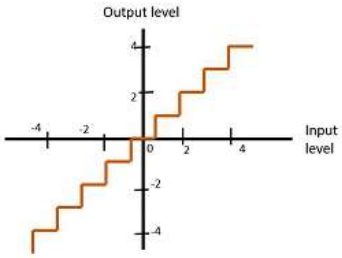


Fig 2 : Mid-Tread type Uniform Quantization

Figure 1 shows the mid-rise type and figure 2 shows the mid-tread type of uniform quantization.

- The **Mid-Rise** type is so called because the origin lies in the middle of a raising part of the stair-case like graph. The quantization levels in this type are even in number.
- The **Mid-tread** type is so called because the origin lies in the middle of a tread of the stair-case like graph. The quantization levels in this type are odd in number.
- Both the mid-rise and mid-tread type of uniform quantizers are symmetric about the origin.

Algorithms

In certain applications, such as speech processing, using a logarithmic computation (called a compressor) before quantizing the input data is common. The inverse operation of a compressor is called an *expander*. The combination of a compressor and expander is called a *compander*.

For a given signal, x, the output of the (μ-law) compressor is

$$y = \frac{\log(1+\mu|x|)}{\log(1+\mu)}\operatorname{sgn}(x).$$

μ is the μ-law parameter of the compander, `log` is the natural logarithm, and `sgn` is the signum function (`sign` in MATLAB®).

μ-law expansion for input signal x is given by the inverse function y^{-1},

$$y^{-1} = \operatorname{sgn}(y)\left(\frac{1}{\mu}\right)((1+\mu)^{|y|}-1) \qquad \text{for } -1 \le y \le 1$$

For a given signal, x, the output of the (A-law) compressor is

$$y = \begin{cases} \dfrac{A|x|}{1+\log A}\operatorname{sgn}(x) & \text{for } 0 \le |x| \le \dfrac{1}{A} \\ \dfrac{(1+\log(A|x|))}{1+\log A}\operatorname{sgn}(x) & \text{for } \dfrac{1}{A} < |x| \le 1 \end{cases}$$

A is the A-law parameter of the compander, `log` is the natural logarithm, and `sgn` is the signum function (`sign` in MATLAB).

A-law expansion for input signal x is given by the inverse function y^{-1},

$$y^{-1} = \operatorname{sgn}(y)\begin{cases} \dfrac{|y|(1+\log(A))}{A} & \text{for } 0 \le |y| < \dfrac{1}{1+\log(A)} \\ \dfrac{\exp(|y|(1+\log(A))-1)}{A} & \text{for } \dfrac{1}{1+\log(A)} \le |y| < 1 \end{cases}$$

Syntax

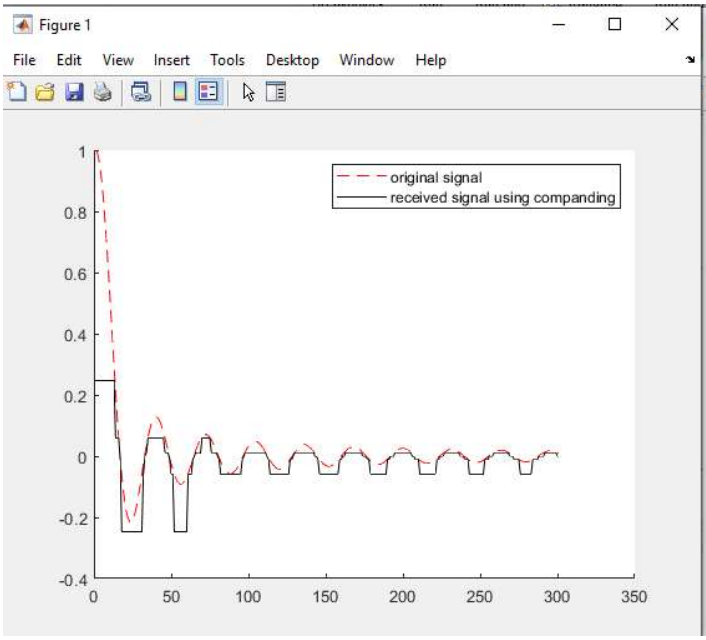
```
out = compand(in,param,v)
out = compand(in,param,v,method)
```

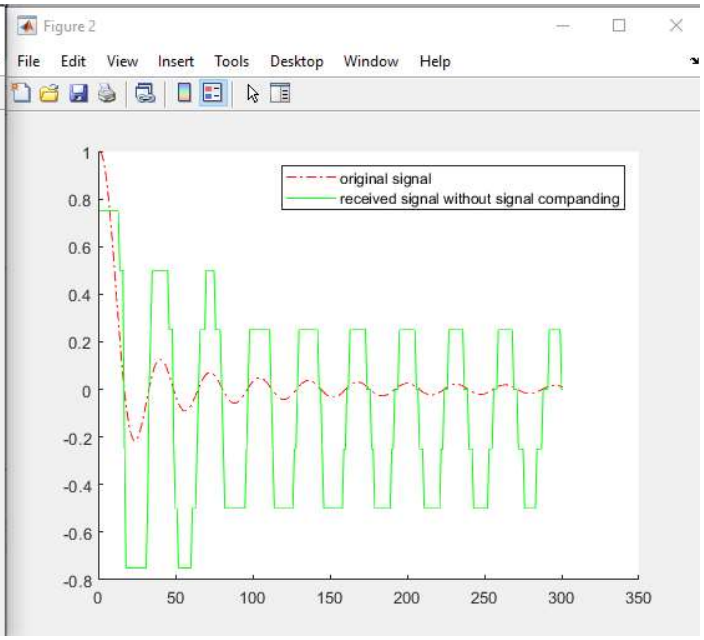
Description

`out = compand(in,param,v)` performs mu-law compression on the input data sequence. The `param` input specifies the mu-law compression value and must be set to a mu value for mu-law compressor computation (a mu-law value of 255 is used in practice). `v` specifies the peak magnitude of the input data sequence.

`out = compand(in,param,v,method)` performs mu-law or A-law compression or expansion on the input data sequence. `param` specifies the mu-law compander or A-law compander value (a mu-law value of 255 and an A-law value of 87.6 are used in practice). `method` specifies the type of compressor or expander computation for the function to perform on the input data sequence.

```
% PCM SIMULATION USING MU LAW COMPANDER
clc;
close all;
clear all;
t=0:0.01:3;
m_t=sinc(2*pi*t);
c=compand(m_t,255,1,'mu/compressor')
e=uencode(c,3)
d=udecode(e,3)
o=compand(d,255,1,'mu/expander')

%PCM with companding
figure(1)
hold on
plot(m_t,'--red');
plot(o, 'black');
legend('original signal','received signal using companding')
hold off

%PCM without companding
e1=uencode(m_t,3)
d1=udecode(e,3)
figure(2)
hold on
plot(m_t,'-.red');
plot(d1,'green');
legend('original signal','received signal without signal companding');
hold off
```

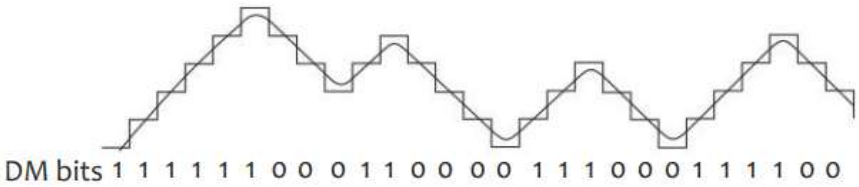




DELTA MODULATION

27 September 2021 15:14

In delta modulation 1bit per sample is transmitted.
N=1 => No. of levels L=2^N =>L=2 -->level 0 or Level1



$\Delta = 0 \leftrightarrow 2$

$\Delta = 0.4$

Let X = message signal ⟶ X(i) = Present sample of Message signal

X_r = staircase signal ⟶ $X_r(i)$ = Previous sample of staircase signal

if message signal sample > staircase signal sample ↓

$x(i) > X_r(i)$

logic level 1 ① ← bit 1 is Transmitted

② Staircase signal is incremented by stepsize;

$x(i) > X_r(i)$

error = $X(i) - X_r(i) = +ve$

Step size = $+\Delta$

$x(i) < X_r(i)$

if message signal sample < staircase signal sample

① bit 0 is Transmitted

② Staircase signal is decremented by Stepsize

slope overload error occurs $\Delta < \frac{d(m(t))}{dt}$ ← stepsize; ↳ slope of message

granular noise occurs when $\Delta > \frac{d}{dt} m(t)$ ↳ slope of message signal

```
clc;
clear all;
close all;
t=0:2*pi/100:2*pi  %Total samples=101
x=5*sin(2*pi*t/5); %message signal with peak voltage 5v & frequency 5Hz
plot(x,'--red');
hold on          %To show multiple graphs in same window
y=[0];  %output of DM signal initially 0, which have bit pattern Ex:1 1 1 0 0 1...
xr=0;   % stair case approximation having initial value=0
del=0.4; %del=stepsize
for i=1:length(x)-1
   if x(i) >=xr(i) %if present sample of a message signal > previous sample of staircase signal
      d=1;       %set d to 1
    xr(i+1)=xr(i)+del; %increment staircase signal by step size
   else
      d=0;
    xr(i+1)=xr(i)-del; %decrement staircase signal by step size
   end
  y=[y d]        % display the bit pattern
end
stairs(xr) % shows stair case approximated signal
hold off
y
del1=0.1;              %chose stepsize as 0.1 instead of 0.4

del2=2;                  % chose stepsize as 2 instead of 0.4
```