Experiment 9-Simulation of Various Digitization Techniques-DPCM, DM and ADM

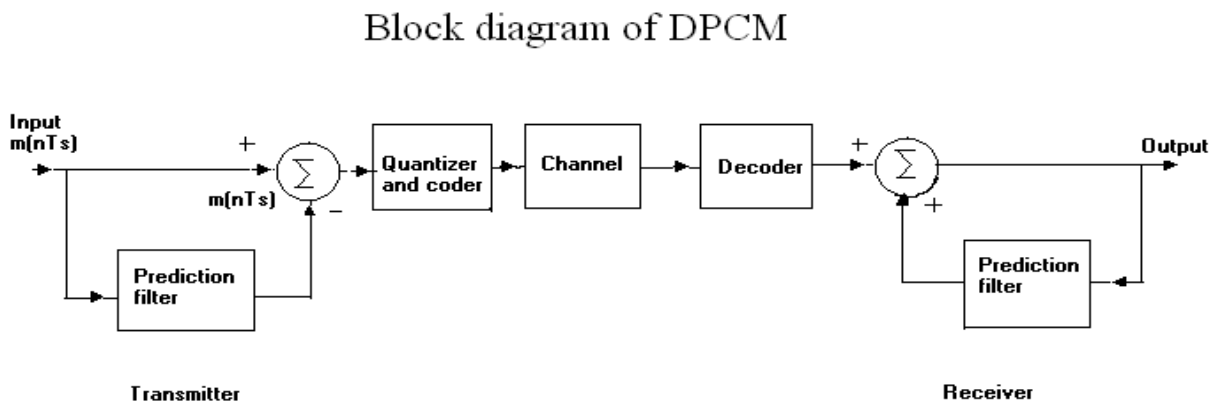
Objective:

Simulation of various digitization techniques used in audio, speech and music signals such as DPCM, DM and ADM.

Pre-Lab Questions

1. Mention the difference between PCM and DPCM?
2. Give the expressions for μ–law and A-law companding.
3. Mention the advantages and disadvantages of DPCM, DM and ADM.
4. What are SOLD and granular noise and how to overcome these in DM?
5. What is synchronization and how it is relevant to staircase approximation in these schemes?

DPCM Block Diagram



Differential PCM is quite similar to ordinary PCM. However, each word in this system indicates the difference in amplitude, positive or negative, between this sample and the previous sample. Thus the relative value of each sample is indicated rather than, the absolute value as in normal PCM.

DPCM is a good way to reduce the bit rate for voice transmission. However it causes some other problems that deal with voice quality. DPCM quantizes and encodes the difference between a previous sample input signal and a current sample input signal. DPCM quantizes the difference signal using uniform quantization. Uniform quantization generates an SNR that is small for small input sample signals and large for large input sample signals. Therefore, the voice quality is better at higher signals.

The input signal is sampled at a higher sampling frequency. Then these samples are modulated. At this point, the DPCM process takes over. The sampled input signals are stored in what is called a predictor. The predictor takes the stored sample signal and sends it through a differentiator. The differentiator compares the previous sample signal and sends its difference to the quantizing and coding phase of PCM (this phase can be a uniform quantizing or Companding with A-law or μ-law). After quantizing and coding, the difference signal is transmitted to its final destination. At the receiving end of the network, everything is reversed. First the difference signal is decoded and dequantized. This difference is added to the sample signal stored in the predictor and send through a low-pass filter that reconstructs the original input signal.

Predictor Model

The quantization requires no a priori knowledge about the transmitted signal. In practice, we often make educated guesses about the present signal based on past signal transmissions. Using such educated guesses to help quantize a signal is known as predictive quantization. The most common predictive quantization method is differential pulse code modulation (DPCM). The predictor is a function that the DPCM encoder uses to produce the educated guess at each step. A linear predictor has the form

y(k) = p(1)x(k-1) + p(2)x(k-2) + ... + p(m-1)x(k-m+1) + p(m)x(k-m)

where x is the original signal, y(k) attempts to predict the value of x(k), and p is an m-tuple of real numbers. Instead of quantizing x itself, the DPCM encoder quantizes the predictive error, x-y. The integer m above is called the predictive order. The special case when m = 1 is called delta modulation.

If the guess for the kth value of the signal x, based on earlier values of x, is

y(k) = p(1)x(k-1) + p(2)x(k-2) +...+ p(m-1)x(k-m+1) + p(m)x(k-m) then the corresponding predictor vector for toolbox functions is predictor = [0, p(1), p(2), p(3),..., p(m-1), p(m)]

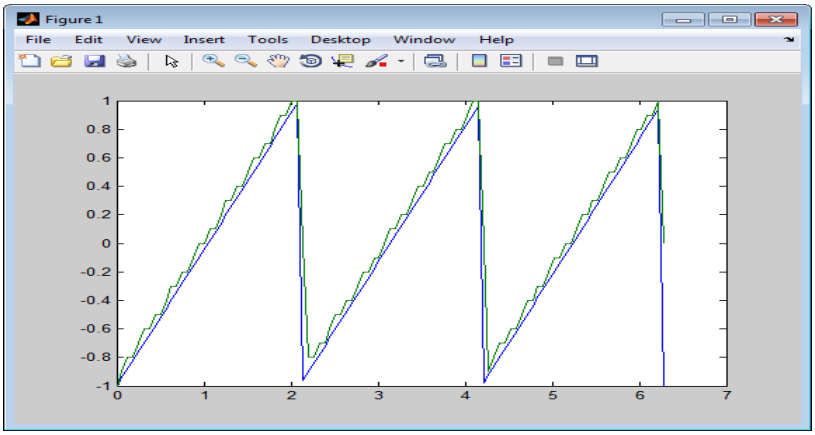
The initial zero in the predictor vector makes sense if you view the vector as the polynomial transfer function of a finite impulse response (FIR) filter.

The functions *dpcmenco, dpcmdeco, and dpcmopt* can implement a DPCM predictive quantizer with a linear predictor.

DPCM Encoding and Decoding

A simple special case of DPCM quantizes the difference between the signal's current value and its value at the previous step. Thus the predictor is just y(k) = x (k - 1). The code below implements this scheme. It encodes a saw tooth signal, decodes it, and plots both the original and decoded signals. The solid line is the original signal, while the dashed line is the recovered signals. The example also computes the mean square error between the original and decoded signals.

```
predictor = [0 1]; % y(k)=x(k-1)

partition = [-1:.1:.9];

codebook = [-1:.1:1];

t = [0:pi/50:2*pi];

x = sawtooth(3*t); % Original signal

% Quantize x using DPCM.

encodedx = dpcmenco(x,codebook,partition,predictor);

% Try to recover x from the modulated signal.

decodedx = dpcmdeco(encodedx,codebook,predictor);

plot(t,x,t,decodedx,'--')

legend('Original signal','Decoded signal','Location','NorthOutside');

distor = sum((x-decodedx).^2)/length(x); % Mean square errorThe output is distor =    0.0327
```



Optimize DPCM Parameters and Comparing

The section Optimize Quantization Parameters describes how to use training data with the *lloyds* function to help find quantization parameters that will minimize signal distortion. Similar procedures for using the *dpcmopt* function in conjunction with the two functions *dpcmenco* and *dpcmdeco*, are explained. The training data you use with *dpcmopt* should be typical of the kinds of signals you will actually be quantizing with *dpcmenco*.

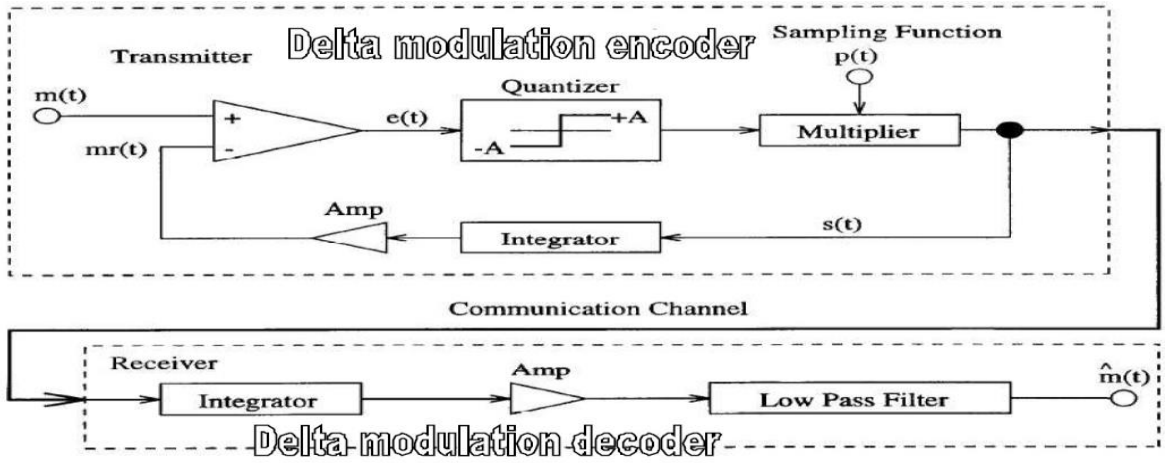
Comparing Optimized and Nonoptimized DPCM Parameters

This uses the same codebook (now called initcodebook) as in the previous code as an initial guess for a new optimized codebook parameter. This example also uses the predictive order, 1, as the desired order of the new optimized predictor. The *dpcmopt* function creates these optimized parameters, using the saw tooth signal x as training data. The example goes on to quantize the training data itself; in theory, the optimized parameters are suitable for quantizing other data that is similar to x. Notice that the mean square distortion here is much less than the distortion in the previous example.

```
t = [0:pi/50:2*pi];

x = sawtooth(3*t); % Original signal

initcodebook = [-1:.1:1]; % Initial guess at codebook

% Optimize parameters, using initial codebook and order 1.

[predictor,codebook,partition] = dpcmopt(x,1,initcodebook);

% Quantize x using DPCM.

encodedx = dpcmenco(x,codebook,partition,predictor);

% Try to recover x from the modulated signal.

decodedx = dpcmdeco(encodedx,codebook,predictor);

distor = sum((x-decodedx).^2)/length(x) % Mean square error
```

The output is distor = 0.0063

Delta Modulation

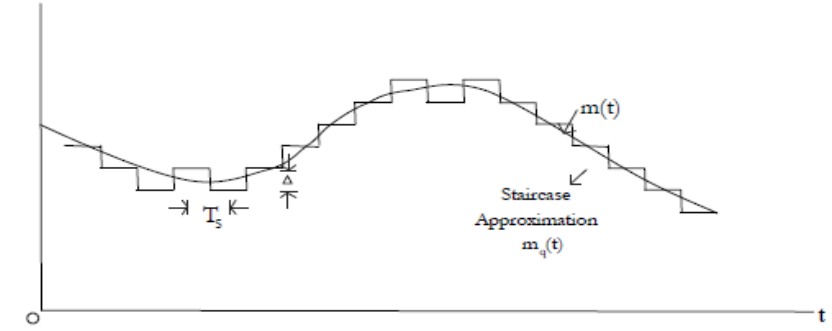


The modulating message signal (t) is applied to the non-inverting input of a high gain differential amplifier. The input analog signal (t) is compared with the loop estimated signal $mr(t)$ to generate the error signal $e(t)$. Then, this error signal is fed into a two level quantizer. The quantized error signal is multiplied by the sampling function (t) to generate the output $s(t)$. The signal $(t)$is integrated and amplified to generate the loop estimate $mr(t)$. In this implementation, the sampling function is a train of narrow pulses with unit amplitude (most practical implementations also). The multiplier becomes a switch and the sampling function becomes the switch enable input. The output (t) is a sequence of pulses with amplitude $+A$ or $-A$. It is very important to note that the DM process samples the quantized error function and not the input signal itself and for narrow sampling pulses the estimated signal $m(t)$ is a stepwise approximation of $m(t)$.

The DM process is a process of analog to digital conversion. Quantization error is present at the output. Two types of quantization noise are present in delta modulation, granular nose and slope overload nose. Granular noise is due to the use of finite quantization steps to reconstruct (approximate) the input signal. It is similar to the quantization noise observed in pulse code modulation (PCM) systems.

Slope overload noise occurs whenever the slope of the input signal exceeds the DM maximum slope or the input signal changes between samples, by an amount greater than the step size of the DM.

DM approximate a waveform by a linear staircase function, the waveform must change slowly relative to the sampling rate. This requirement implies that waveform must be oversampled, i.e., at least five times the Nyquist rate. "Oversampling" means that the signal is sampled faster than is necessary. In the case of Delta Modulation this means that the sampling rate will be much higher than the minimum rate of twice the bandwidth. Delta Modulation requires "oversampling" in order to obtain an accurate prediction of the next input. Since each encoded sample contains a relatively small amount of information Delta Modulation systems require higher sampling rates than PCM systems.



Binaries sequece at modulator o/p 0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0

DM Simulation

The simulation in this lab includes the modulation (encoding) and demodulation (decoding) steps.

1. Construct a time array from 0 to 1/25 seconds with ts=1/fs interval (sample frequency fs=2000). Name it 'tn'.
2. Set the step size 1/15. Name it 'StepSize'.
3. Construct a signal 'm' which is a sine wave with 0.5 amplitude and frequency of 50, time is 'tn'. The signal ‘m’ is our modulating message signal.
4. Call function Delta modulation encoder 's=DM_Encod(m, StepSize)'With input signal 'm' and step size "StepSize' as the parameter to the function. 's' is the output. Thefollowing steps describe the procedure to write this function.

4.1 Use 'length' to find the size of the signal m, name it 'xlen'.

4.2 Set accumulation at start is zero.

4.3 Compare input signal(s) with accumulation (i) from beginning to end. When signal is greater than accumulation, encoder out s(i)=+1, next accumulation (i+1) equals accumulation (i) plus step size. When signal less than accumulation,, encoder out s(i)=-1, next accumulation(i+1) equals accumulation(i) minus step size.

5. Call function Delta modulation decoder '[Si]=DM_Decod (StepSize,s)'

With input signal 's' and step size "StepSize'. Output 'Si'.

5.1 Use 'length' find size of signal (s), name it 'xlen'

5.2 Set accumulation at start is zero.

5.3 Compare input signal s(i) with zero from beginning to end. When s(i) is greater than 0, next accumulation(i+1) equals accumulation(i) plus step size. Else, next accumulation (i+1) equals accumulation (i) minus step size.

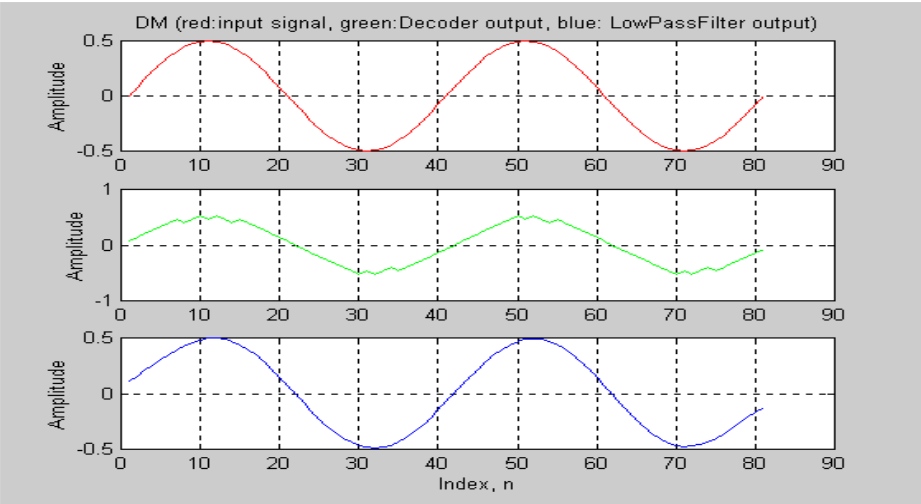
5.4 Set output 'Si' equal equals accumulation (2:i+1)

6. Call function low pass filter 'So=LowPassFilter(100, 4*50/fs, Si);' Input signal 'Si', Output 'So'.

This function program is given in following.

```
function So=LowPassFilter(fod, cf, Si)
%lowpass filter
%So=LowPassFilter(100, .1, Si);
%
%fod: filter order.
%cf: cut-off frequency.
%So: output.
b=fir1(fod,cf);
size(b)
So = conv2(Si,b,'same');
```

7. Plot the signal of input 'm', Decoder output 'Si' and Lowpassfilter output 'So' vs time. Use 'subplot' function plot them in same template.
8. By change the (1) fs to 500, 1000, rerun program. What happen? Why?
9. By change the (2) step size to 1/20, 1/40 rerun program. What happen? Why?
10. Submit a report and the soft copy of your codes.



Top sinewave signal is given as the input to the modulator. Next signal is the decoder output. The last signal is the low pass filter. So it is faithful reproduction of the original message signal

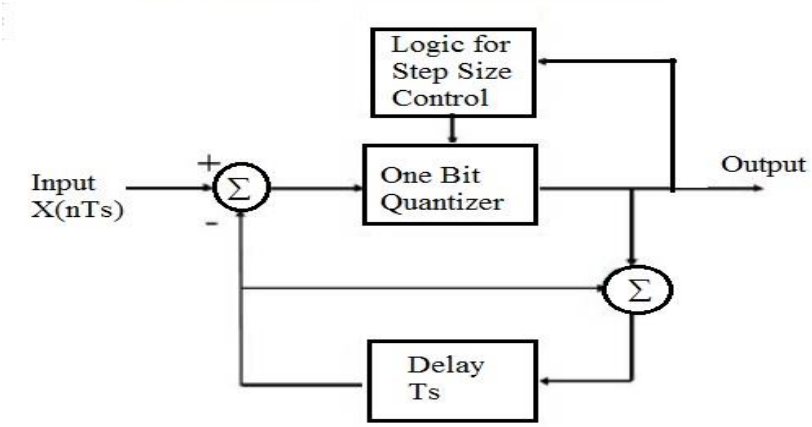
Hardware Implementation of DM

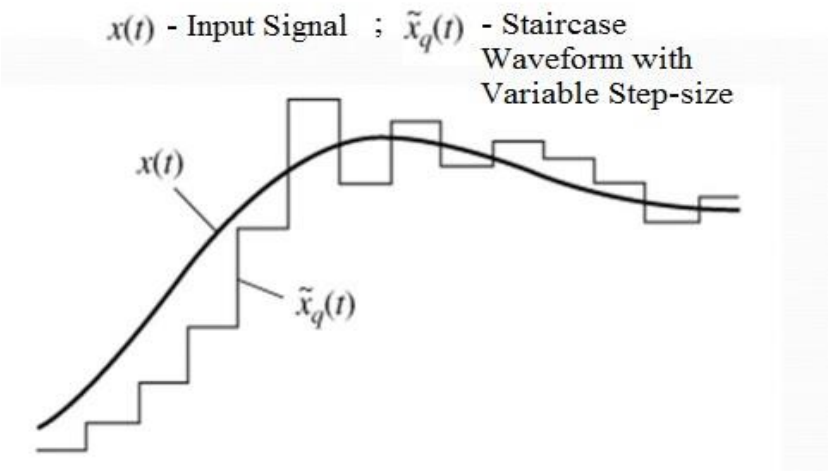
For the hardware experiment we will focus on the encoding as well as estimation process of the Delta Modulation (DM).

The circuit for DM consists of an LM741 operate amplifier, CD4013 dual 'D' flip flop and CD4016 bilateral switch. The LM 741 is operated open loop as a comparator between the input signal (t) and the feedback signal $mr(t)$. The function of the CD4013 is to hold the value of the quantized error signal constant (+ or – Vcc) during the sampling period. Both the flip-flop and the bilateral switch are enabled by same clock pulse. Propagation delay in the flip-flop may be considered negligible in comparison with the pulse width. A dc integrator is used in the feedback loop.

Adaptive Delta Modulation

ADM is the refined form of delta modulation. This method was introduced to solve the granular noise and slope overload error caused during Delta modulation. This Modulation method is similar to Delta modulation except that the step size is variable according to the input signal. Here first the difference between the present sample value and previous approximation is calculated. This error is quantized, if the present sample is smaller than the previous approximation, quantized value is high or else it is low. The one-bit quantizer o/p is given to the Logic step size control where the step size is decided





Adaptive Delta Modulation plot

Post-Lab Questions

1. What is the role of predication filter in DPCM?
2. Give the SNR expressions for DPCM, error in single sample for DM?
3. Mention the applications of DPCM, DM and ADM?
4. Compare DPCM with Delta Modulation?
5. Why an integrator required and what is the effect of frequency in ADM?

Model Codes:

Applying uniform quantization (DPCM) on a sound signal

```
%quantizing a sound signal
function []=demo_quant(inname,outname, N);
if nargin < 3
disp('Usage: sampl_quant(inname,outname, N)');
disp('inname: input .wav file name');
disp('outname: output .wav file name');
disp('N: quantization level, N>1');
return;
end;
%read in input signal
[x,fs,N0]=wavread(inname);
xmin=min(x); xmax=max(x);
Q=(xmax-xmin)/N;
disp('N0,xmin,xmax,N,Q');
disp([N0,xmin,xmax,N,Q]);
%apply uniform quantization on each sample
xq=sign(x).*(floor((abs(x)+Q/2)/Q)*Q);
%compare sound quality
wavwrite(xq,fs,N0,outname);
sound(x,fs);
pause;
sound(xq,fs);
%plot waveforms over the entire period
t=1:length(x);
```

```
figure; plot(t,x,'r:');
hold on; plot(t,xq,'b-');
axis tight; grid on;
%plot waveform over a selected period
t=5000:5100;
figure; plot(t,x(5000:5100),'r:');
hold on; plot(t,xq(5000:5100),'b-');
axis tight; grid on;
```

Delta Modulation on a sinusoidal signal

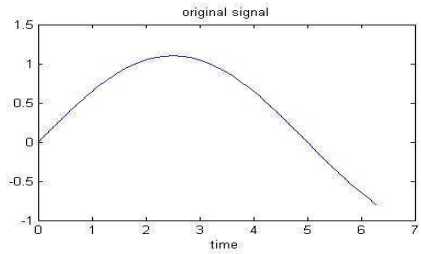
```
function [t,x,xx]=sindm(Q, xmean);
if nargin < 1
disp('Usage: sindm(Q, xmean)');
disp('Q: stepsize');
disp('xmean: mean value of the signal');
return;
end;
% construct a quantized sinusoid wave signal using 8-bit quantizer
% { ranged at (0,255) } for sampling time interval t=(0,1).
% given a sampling frequency.
fs=33;
t=[0:1/fs:1];
L=length(t);
f=2;
x=(sin(2*pi*f*t)+1.0)/2*255;
x=round(x); %the round operation essentially quantizes to 8 bits,
because the range
% of x is 0-255.
% Delta Modulation
D=Q*ones(L); % fixed stepsize=30
%xmean=128;
% given the initial condition.
d(1)=x(1); %difference signal
c(1) =0; %coded signal
dd(1)=D(1); %quantized difference signal
xx(1)=xmean+dd(1); %reconstructed signal
%sindm.m
% calculate the delta modulation.
for i = 2:L,
d(i)=x(i)-xx(i-1);
if d(i) > 0
c(i) = 0;
dd(i)=D(i);
else
c(i) = 1;
dd(i)=(-1)*D(i);
end
xx(i)=xx(i-1)+dd(i);
end
figure;
t1=[0:1/(100*fs):1];
x1=(sin(2*pi*f*t1)+1.0)/2*255;
plot(t1,x1,t,x,'*',t,xx,'x',t(2:L),xx(1:(L-1)),'o');
title('Illustration of the linear delta modulation');
legend('original signal', 'original sample','reconstructed value','predicted value');
```

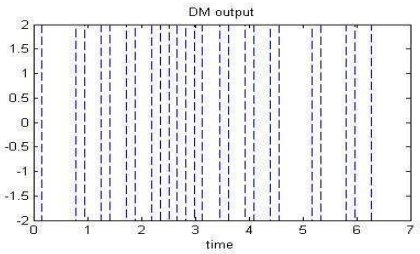
Adaptive Delta Modulation (ADM) with sine as input signal

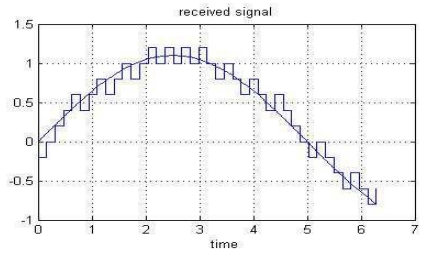
```
function [t,x,xx]=sinadm(P,xmean,dmin,dmax)
```

```
if nargin < 1
disp('Usage: sinadm(P,xmean,dmin,dmax)');
disp('P: Adaptation Parameter, 1<=P<=3');
disp('xmean: mean value of x');
disp('dmin, dmax: min and max of stepsize');
return;
end;
% given a sampling frequency.
fs=33;
% construct a quantized sinusoid wave signal using 8-bit quantizer
% { ranged at (0,255) } for sampling time interval t=(0,1).
t=[0:1/fs:1];
L=length(t);
f=2;
x=(sin(2*pi*f*t)+1.0)/2*255;
x=round(x);
% Adaptive Delta Modulation
%P=1.8;
%dmin=2;
%dmax=40;
%xmean=128;
Q=1.0/P;
% given the initial condition.
d(1)=x(1);
c(1) =0;
dd(1)=(dmin+dmax)/2;
xx(1)=xmean+dd(1);
% calculate the adaptive delta modulation.
for i = 2:L,
d(i)=x(i)-xx(i-1);
if d(i) > 0
c(i) = 0;
else
c(i) = 1;
end
if c(i) == c(i-1)
M=P;
else
M=Q;
end
dd(i)=M*dd(i-1);
%dd(i)=round(dd(i));
if dd(i) < dmin
dd(i)=dmin;
elseif dd(i) > dmax
dd(i)=dmax;
end
if c(i) == 0
xx(i)=xx(i-1)+dd(i);
elseif c(i) == 1
xx(i)=xx(i-1)-dd(i);
end
end
% graph of the fixed stepsize delta modulation of a sinusoid wave signal.
figure;
t1=[0:1/(100*fs):1];
x1=(sin(2*pi*f*t1)+1.0)/2*255;
plot(t1,x1,'-',t,x,'*',t,xx,'x',t(2:L),xx(1:(L-1)),'o');
title('Illustration of the adaptive delta modulation');
legend('original signal', 'original sample', 'reconstructed value', 'predicted value');
```

Waveforms output for Delta Modulation







Conclusion

The digitization of analog signals is simulated and analysed using DPCM, DM and ADM schemes.