Delay Based Effects

Many useful audio effects can be implemented using a **delay structure**:

- Sounds reflected off walls
  - In a cave or large room we hear an echo and also **reverberation** takes place – this is a different effect — **see later**
  - If walls are closer together repeated reflections can appear as parallel boundaries and we hear a modification of sound colour instead.
- **Vibrato**, **Flanging**, **Chorus** and **Echo** are examples of delay effects

Basic Delay Structure

The Return of IIR and FIR filters:

We build basic delay structures out of some very basic **IIR** and **FIR** filters:

- We use *FIR* and *IIR comb filters*
- Combination of FIR and IIR gives the **Universal Comb Filter**

FIR Comb Filter

FIR Comb Filter: A single delay

This simulates a **single delay**:

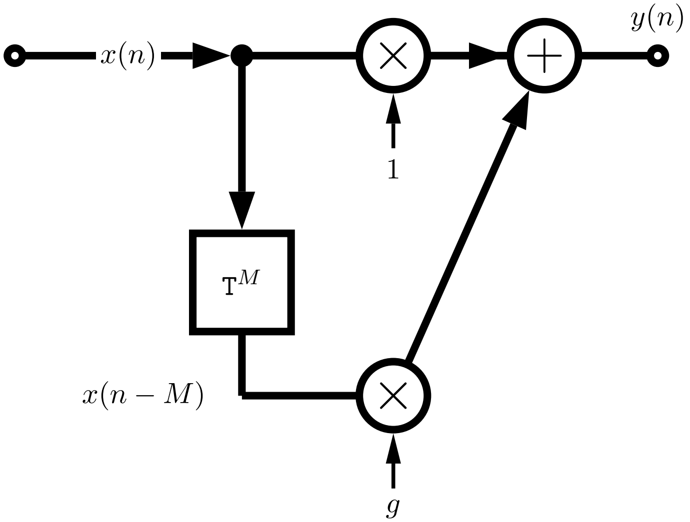
- The input signal is delayed by a given time duration, τ.
- The delayed (processed) signal is added to the input signal some amplitude gain, g
- The difference equation is simply:

$$y(n) = x(n) + gx(n - M) \qquad \text{with} \quad M = \tau / f_s$$

- The transfer function is:

$$H(z) = 1 + gz^{-M}$$

FIR Comb Filter Signal Flow Diagram



FIR Comb Filter MATLAB Code

fircomb.m:

```
x=zeros(100,1);x(1)=1; % unit impulse signal of length 100

g=0.5; %Example gain

Delayline=zeros(10,1); % memory allocation for length 10

for n=1:length(x);
  y(n)=x(n)+g*Delayline(10);
  Delayline=[x(n);Delayline(1:10-1)];
end;
```

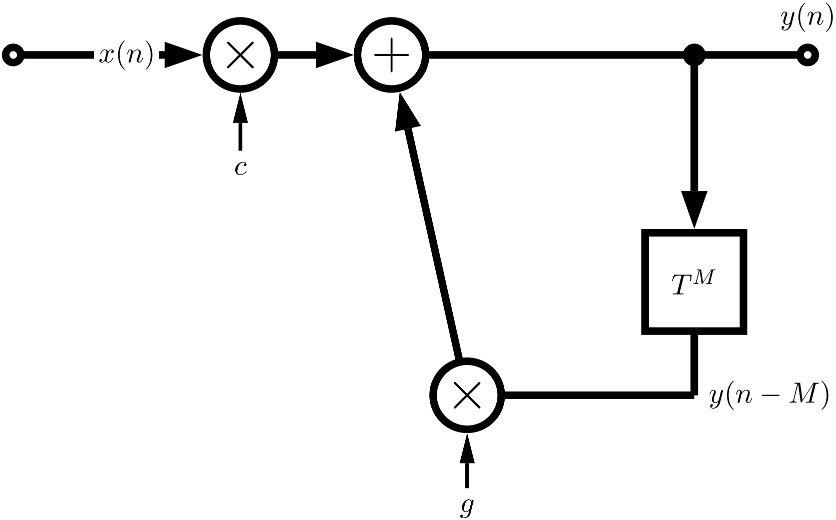
IIR Comb Filter

IIR Comb Filter

- Simulates *endless reflections* at both ends of cylinder.
- We get an endless series of responses, $y(n)$ to input, $x(n)$.
- The input signal circulates in delay line (delay time τ) that is fed back to the input.
- Each time it is fed back it is attenuated by g.
- Input sometime scaled by c to **compensate** for high amplification of the structure.
- The difference equation is simply:

$$y(n) = Cx(n) + gy(n-M) \qquad \textbf{with} \quad M = \tau / f_s$$

IIR Comb Filter Signal Flow Diagram



IIR Comb Filter MATLAB Code

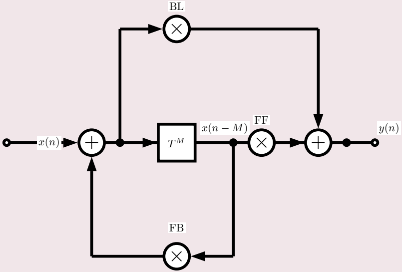
iircomb.m:

```
x=zeros(100,1);x(1)=1; % unit impulse signal of length 100

g=0.5;

Delayline=zeros(10,1); % memory allocation for length 10

for n=1:length(x);
   y(n)=x(n)+g*Delayline(10);
   Delayline=[y(n);Delayline(1:10-1)];
end;
```

Universal Comb Filter

Universal Comb Filter

- Combination of the FIR and IIR comb filters.
- Basically this is an **allpass filter** with an **M** sample delay operator and an additional multiplier, **FF**.



- Parameters:
  FF = **feedforward**, **FB** = **feedbackward**, **BL** = **blend**

Universal Comb Filter Parameters

Why is "Universal"?

- **Universal** in that we can form any **comb** filter, an **allpass** or a delay filter:

	BL	FB	FF
FIR Comb	1	0	g
IIR Comb	1	g	0
Allpass	a	$-a$	1
delay	0	0	1

Universal Comb Filter MATLAB Code

unicomb.m:

```
x=zeros(100,1);x(1)=1; % unit impulse signal of length 100

BL=0.5;
FB=-0.5;
FF=1;
M=10;

Delayline=zeros(M,1); % memory allocation for length 10

for n=1:length(x);
  xh=x(n)+FB*Delayline(M);
  y(n)=FF*Delayline(M)+BL*xh;
  Delayline=[xh;Delayline(1:M-1)];
end;
```

Vibrato - A Simple Delay Based Effect

Vibrato:

- **Vibrato** — **Varying** (**modulating**) the time delay periodically.
- If we **vary** the **distance** between an **observer** and a **sound source** (*cf. Doppler effect*) we hear a change in pitch.
- **Implementation**: A **Delay line** and a **low frequency oscillator** (LFO) to **vary** the **delay**.
- **Only listen** to the **delay** — no forward or backward feed.
- Typical delay time = **5–10** Ms and LFO rate = **5–14**Hz.

Vibrato MATLAB Code

vibrato.m function:

- See vibrato_eg.m for sample call this function

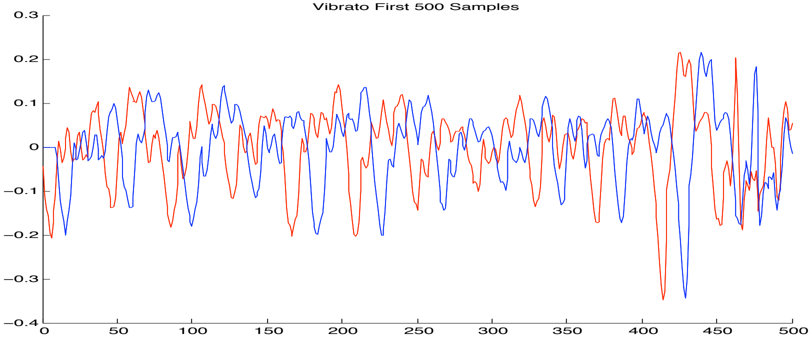
```
function y=vibrato(x,SAMPLERATE,Modfreq,Width)

ya_alt=0;
Delay=Width; % basic delay of input sample in sec
DELAY=round(Delay*SAMPLERATE); % basic delay in # samples
WIDTH=round(Width*SAMPLERATE); % modulation width in # samples
if WIDTH>DELAY
  error('delay greater than basic delay !!!');
  return;
end;

MODFREQ=Modfreq/SAMPLERATE; % modulation frequency in # samples
LEN=length(x);          % # of samples in WAV-file
L=2+DELAY+WIDTH*2;      % length of the entire delay
Delayline=zeros(L,1); % memory allocation for delay
y=zeros(size(x));       % memory allocation for output vector
```

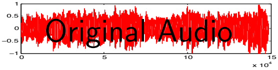
Vibrato MATLAB Code (Cont.)

vibrato.m (Cont.)

```
for n=1:(LEN-1)
  M=MODFREQ;
  MOD=sin(M*2*pi*n);
  ZEIGER=1+DELAY+WIDTH*MOD;
  i=floor(ZEIGER);
  frac=ZEIGER-i;
  Delayline=[x(n);Delayline(1:L-1)];
  %---Linear Interpolation-----------------------------
  y(n,1)=Delayline(i+1)*frac+Delayline(i)*(1-frac);
  %---Allpass Interpolation------------------------------
  %y(n,1)=(Delayline(i+1)+(1-frac)*Delayline(i)-(1-frac)*ya_alt);
  %ya_alt=ya(n,1);
end
```

The output from the above code is (red plot is original audio):





Click image or here to hear: original audio, vibrato audio.

Comb Filter Delay Effects: Flanger, Chorus, Slapback, Echo

- A few other popular effects can be made with a comb filter (FIR or IIR) and some modulation.
- Flanger, Chorus, Slapback, Echo same basic approach but *different sound* outputs:

Effect	Delay Range (ms)	Modulation
Resonator	$0 \ldots 20$	None
Flanger	$0 \ldots 15$	Sinusoidal (≈ 1 Hz)
Chorus	$10 \ldots 25$	Random
Slapback	$25 \ldots 50$	None
Echo	> 50	None

- **Slapback** (or doubling) — quick repetition of the sound,
  Flanging — continuously varying LFO of delay,
  Chorus — **multiple copies** of sound delayed by small random delays

Flanger MATLAB Code

flanger.m:

```
% Creates a single FIR delay  with the delay time oscillating from
%  Either 0-3 ms or 0-15 ms at 0.1 - 5 Hz

infile='acoustic.wav';
outfile='out_flanger.wav';

% read the sample waveform
[x,Fs] = audioread(infile);

% parameters to vary the effect %
max_time_delay=0.003; % 3ms max delay in seconds
rate=1; %rate of flange in Hz

index=1:length(x);

% sin reference to create oscillating delay
sin_ref = (sin(2*pi*index*(rate/Fs)))';

%convert delay in ms to max delay in samples
max_samp_delay=round(max_time_delay*Fs);
```

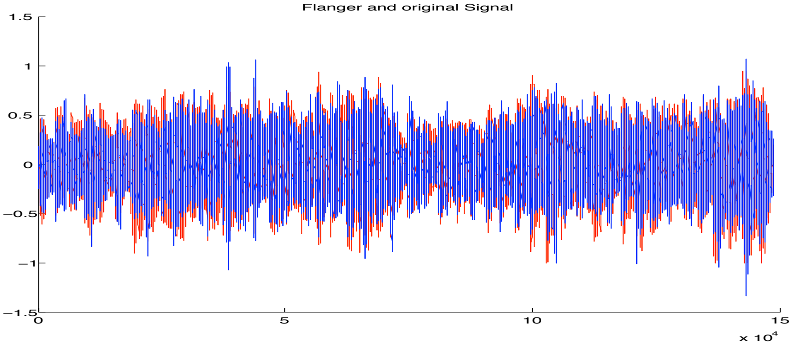
Flanger MATLAB Code (Cont.)

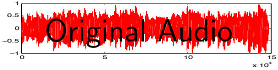
flanger.m (Cont.):

```
% create empty out vector
y = zeros(length(x),1);

% to avoid referencing of negative samples
y(1:max_samp_delay)=x(1:max_samp_delay);

% set amp suggested coefficient from page 71 DAFX
amp=0.7;

% for each sample
for i = (max_samp_delay+1):length(x),
  cur_sin=abs(sin_ref(i));     %abs of current sin val 0-1
  % generate delay from 1-max_samp_delay and ensure whole number
  cur_delay=ceil(cur_sin*max_samp_delay);
  % add delayed sample
  y(i) = (amp*x(i)) + amp*(x(i-cur_delay));
end

% write output
audiowrite(outfile, y, Fs);
```

Flanger MATLAB Example (Cont.)

The output from the above code is (red plot is original audio):





Click here to hear: original audio, flanged audio.