

# Flanging and Phasing

Josh Reiss

Digital Audio Effects Tutorial

AES 126<sup>th</sup>

centre for digital music



Sources include information from Harmony Central and "Flanging " at <http://ccrma.stanford.edu/~jos/pasp/Flanging.html> by Julius Smith III , and Digital Audio Effects, edited by Udo Zoelzer. Software used in programming demonstrations is Max MSP. This is intended for education only.

From <http://www.avrev.com/music/revs/beatles/> The beatles and the double vibrocated splashing flange, Bill Biersach

1966, Beatles went to Abbey Road Studio to begin work on *Revolver*. "They would relate what sounds they wanted and we then had to go away and come back with a solution," recalls Ken Townsend, who was one of the studio technicians. "For example, they often liked to double-track their vocals, but it's quite a laborious process and they soon got fed up with it. So, after one particularly trying night-time session doing just that, I was driving home and suddenly had an idea."

Townsend's solution was Artificial Double Tracking. It was a process by which a recorded vocal track was re-recorded onto a separate machine which had been modified with a variable oscillator, causing the tape speed (and therefore the pitch) to fluctuate slightly up and down. This modulated signal was then fed back into the first machine to be combined with the original signal. The irregularities in the two signals imitated the natural variations in the human voice when overdubbed against itself. The resulting effect emulated a double-tracked vocal.

The moment Lennon heard it, he was overjoyed. Being a spontaneous musician, a "one-take man," he loathed the discipline of matching consonants and fricatives when double-tracking his own vocals. He asked Martin to explain it to him.

"I knew he'd never understand it," recalls Martin with amusement, "so I said, 'Now listen, it's very simple. We take the original image and split it through a double vibrocated splashing flange with double negative feedback ...' He said, 'You're pulling my leg, aren't you?' I replied, 'Well, let's flange it again and see.' From that moment on, whenever he wanted it he'd ask for his voice to be 'flanged,' or call out for 'Ken's flanger.' "

# Flanging

- characteristic sound referred to as *whooshing*
  - Similar to sound of jet plane flying overhead.
- Only delayed copy has pitch change, which is mixed in with unaltered signal.
- Delay change determined by LFO (Low Frequency Oscillator)
- No echo
  - Typical delay times are from 1 to 10 milliseconds
  - Human ear will perceive echo if delay > 50-70 milliseconds
- *Pitch modulation* caused by changing delay in flanger
  - *Read faster* or *read slower* from delayed signal
  - Perceived pitch *warbles*.
  - Increase delay between 2 signals → pitch drops
  - Decrease delay → increases pitch



**Drum loop dry, followed by  
same loop with flanging applied**

The flanging effect has been described as a kind of "whoosh" passing subtly through the sound. The effect is also compared to the sound of a jet passing overhead, in which the direct signal and ground reflection arrive at a varying relative delay.

The audio example presents a basic dry drum loop and that same drum loop with flanging applied.

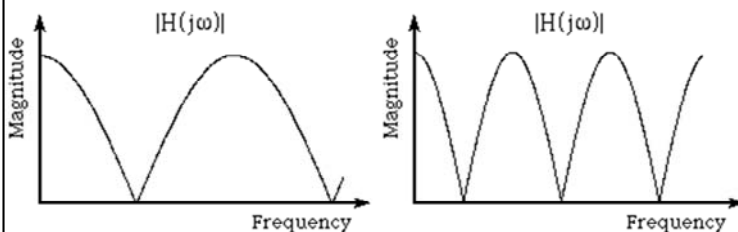
Flanging is created by mixing a signal with a slightly delayed copy of itself, where the length of the delay is constantly changing.

This changing of the delay in the flanger creates some **pitch modulation** - the perceived pitch 'warbles'. This happens because you have to 'read faster' or 'read slower' from the delayed signal. As you slow down, the pitch drops. To decrease the delay, you have to catch up - sort of like fast forwarding, which increases the pitch. Of course only the delayed copy of the sound has this pitch change, which is then mixed in with the unaltered signal.

When we listen to a flanged signal, we don't hear an echo because the delay is so short. In a flanger, the typical delay times are from 1 to 10 milliseconds (the human ear will perceive an echo if the delay is more than 50-70 milliseconds or so).

## Frequency Response

- delay has filtering effect on signal
- Creates equally spaced notches in frequency response.
- continuously changing amount of delay used
  - Notches sweep up and down frequency axis over time
  - As delay increases, notches slide down to lower frequencies
- Frequencies with magnitude response 0 are eliminated
- Other frequencies passed with amplitude change



Frequency response of 2 flangers, depth=1.  
Left has smaller delay than right.



Video of sweeping  
with an audio track

Instead of creating an echo, the delay has a filtering effect on the signal, and this effect creates a series of notches in the frequency response, as shown in the figure. Points at which the frequency response goes to zero means that sounds of that frequency are eliminated, while other frequencies are passed with some amplitude change. This frequency response is sometimes called a comb filter, as its notches resemble the teeth on a comb.

The characteristic sound of a flanger results when these notches sweep up and down the frequency axis over time. Picture the notches compressing and expanding like a spring between the two plots in the figure.

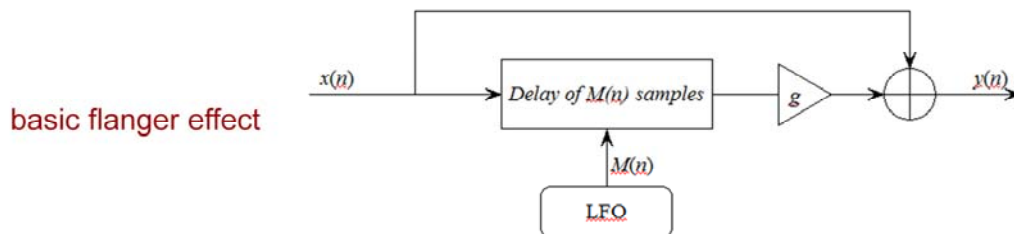
If you have Quicktime installed on your computer, view this movie of a flanger that demonstrates the sweeping with an audio track. The sweeping action of the notches is achieved by continuously changing the amount of delay used. As the delay increases, the notches slide further down into the lower frequencies. The manner in which the delay changes is determined by the LFO (Low Frequency Oscillator) waveform.

## Flanging Introduction

- input-output relation for basic flanger

$$y(n) = x(n) + gx[n - M(n)]$$

- $x(n)$  - input signal amplitude at time  $n=0,1,2,\dots$ ,
- $y(n)$  - output at time  $n$
- $g$  - mix parameter
- $M(n)$  - length of delay-line at time  $n$
- $M(n)$  must vary smoothly over time
  - *interpolated delay line* gives noninteger values of  $M$



Flanging is modeled quite accurately as a feedforward comb filter, in which the delay  $M$  is varied over time. Figure depicts such a model. The input-output relation for a basic flanger can be written as

$$y(n) = x(n) + gx[n - M(n)]$$

where  $x(n)$  is the input signal amplitude at time  $n=0,1,2,\dots$ ,  $y(n)$ , is the output at time  $n$ ,  $g$  is the mix parameter (the depth of the flanging effect), and  $M(n)$  is the length of the delay-line at time  $n$ . Since  $M(n)$  must vary smoothly over time, it is clearly necessary to use an *interpolated delay line* to provide non-integer values of  $M$  in a smooth fashion.

# Interference

Delay a sine wave and add it to original

- Destructive interference
  - Signals perfectly out of phase → no output signal.
  - Cause of notches in frequency response
- Constructive interference
  - Signals remain in phase → double magnitude of that freq.
- Depth control
  - When depth is 0, frequency response is flat,
  - Increase depth → notches appear and extend downward
  - When depth=1, notches extend down to 0
  - Even if notches > 0, they have audible effect.

[MSP Code](#)  
[flanger delay](#)

These notches in the frequency response are created by destructive interference. Picture a perfect tone - a sine wave. If you delay that signal and then add it to the original, the sum of the two signals may look quite different. At one extreme, where the delay is such that the signals are perfectly out of phase, as one signal increases, the other decreases the same amount, so the entire signal will disappear at the output. Of course, the two signals could still remain in phase after the delay, doubling the magnitude of that frequency (constructive interference). For any given amount of delay, some frequencies will be eliminated while others are passed through. In the flanger, you can control how deep these notches go by using the depth control. When the depth is at zero, the frequency response is flat, but as you increase the depth, the notches begin to appear and extend downward, reaching zero when the depth is one. Even if the notches do not extend quite all the way to zero, they will still have an audible effect.

## Derivation of Flanger's Frequency Response

**Delay Equation:**  $y(t) = x(n) + x(n - M)$

**Z-Transform:**  $Y(z) = X(z) + z^{-M} X(z), \quad z = e^{j\omega}$

**Transfer Function:**  $H(z) = Y(z) / X(z) = z^{-0} + z^{-M}$

$H(j\omega) = e^{-j0} + e^{-j\omega M}$

**Frequency Response:**  $H(j\omega) = e^{-j\omega M/2} \underbrace{(e^{-j\omega M/2} + e^{j\omega M/2})}_{2 \cos(\omega M/2)}$

**Amplitude Response:**  $|H(j\omega)| = |2 \cos(\omega M / 2)|$

These equations depict the derivation of the frequency response of a delay equation, which is a simplified model of the flanger.

For a 1-tap FIR filter with coefficients equal to 1, i.e., the delay equation, whose output is described by:

(see 1<sup>st</sup> eq.),

the filter's Z transform is:

(see 2<sup>nd</sup> eq.) ,

The variable  $z$  in  $H(z)$  is a continuous complex variable, and we can describe it as:  $z=r \cdot e^{j\omega}$ , where  $r$  is a magnitude and  $\omega$  is the angle of  $z$ . If we let  $r=1$ , then  $H(z)$  around the unit circle becomes the filter's frequency response  $H(j\omega)$ . This means that substituting  $e^{j\omega}$  for  $z$  in  $H(z)$  gives us an expression for the filter's frequency response  $H(\omega)$ , which is:

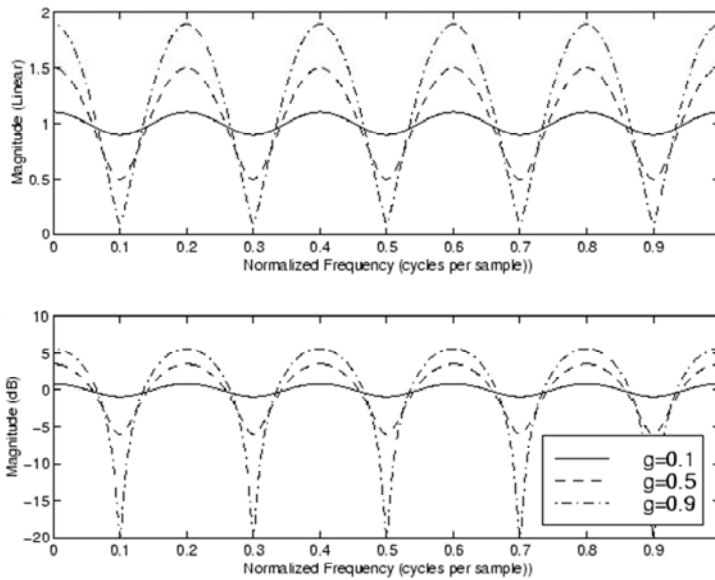
(see 3<sup>rd</sup> eq.) ,

Using Euler's identity,  $e^{-ja}=\cos(a) - j\sin(a)$ , we can write  $H(\omega)$  in rectangular form as:

(see 4<sup>th</sup>, 5<sup>th</sup> eq.) .

## Amplitude Response

- Peaks at frequencies  $\omega_k^{(p)} = 2\pi k / M, \quad k = 0, 1, \dots, M-1$
- $M$  notches at frequencies  $\omega_k^{(n)} = \omega_k^{(p)} + \pi / M$ 
  - Notches at intervals of  $f_s/M$  Hz



[MSP Code example F2](#)

- Comb structure
- 5 notches for  $M=5$
- Notch motion essential for effect
  - isolated notch may be inaudible

the frequency response has a comb shaped structure. For  $g > 0$ , there are  $M$  peaks in the frequency response. For  $g=1$ , the peaks are maximally pronounced, with  $M$  notches (notch refers to the elimination of sound energy at a single frequency or over a narrow frequency interval) occurring between them at frequencies. As the delay length  $M$  is varied over time, these “comb teeth” squeeze in and out like the pleats of an accordion. As a result, the spectrum of any sound passing through the flanger is “massaged” by a variable comb filter. At any given time there are  $M(n)$  notches in the flanger's amplitude response (counting positive- and negative frequency notches separately). The notches are thus spaced at intervals of  $f_s/M$  Hz, where  $f_s$  denotes the sampling rate. In particular, the notch spacing is inversely proportional to delay-line length. The time variation of the delay-line length  $M(n)$  results in a “sweeping” of uniformly-spaced notches in the spectrum. The flanging effect is thus created by moving notches in the spectrum. Notch motion is essential for the flanging effect. Static notches provide some coloration to the sound, but an isolated notch may be inaudible. Since the steady-state sound field inside an undamped acoustic tube has a similar set of uniformly spaced notches (except at the ends), a static row of notches tends to sound like being inside an acoustic tube. Amplitude responses of the feed forward comb-filter with  $M=5$ , linear amplitude scale & Decibel scale. The frequency axis goes from 0 to the sampling rate (instead of only half the sampling rate, which is more typical for real filters) in order to display the fact that the number of notches is exactly 5. For  $g=1$ , we obtain  $M$  nulls, or notches. The notches are points (frequencies) of zero gain in the amplitude response. Note that in flangers, these notches are moved slowly over time by modulating the delay length. Doing this smoothly requires interpolated delay lines.

# Flanger Parameters

- LFO Waveform
  - how delay varies in time
- Rate/Speed of LFO
  - # times per second notches sweep up and down
  - rate affects amount of pitch modulation
- Wet/dry mix control
  - The more wet the mix, more pronounced notches in flanger
- Minimum delay
  - Determines how high first notch will go
  - Delay  $\gg 0$ , first notch drops down.
  - Delay  $\rightarrow 0$ , notches sweep uppermost frequency range, and momentarily disappear



drum loop flanged with a delay of 1ms, and then 4 ms. Notches sweep downward, then upward before returning to initial location.



**LFO-** Some flangers will allow you to choose the LFO waveform. This waveform determines how the delay in the flanger varies in time. The triangle is probably the more common choice in flangers.

**Delay-** The delay parameter specifies the minimum delay used on the copy of the input signal - as the delay changes, this will be the smallest delay. Looking at the frequency response, this value determines how high the first notch will go. As the delay is increased, the first notch drops down. In some cases, the delay parameter can be set to zero, in which case the notches will sweep the uppermost frequency range, and essentially disappear momentarily. In other cases, you may not be able to control delay parameter. The sound example contains the same drum loop used above with two different delay settings. Try to picture the notches sweeping up and down.

**Mix-** The more wet the mix (amount of delayed signal added to original), the more pronounced the notches in the flanger. In multieffects units, the depth may only be controllable in the mixer section, and not available within the flanging processor. Some people use the term 'depth' interchangeably with 'mix'. To obtain maximum effect, the mix control,  $g$  should be set to 1.  $g=0$  gives no effect.

The **rate** control is pretty straightforward. This parameter refers to the rate at which the LFO waveform repeats itself, or equivalently, how many times per second the notches sweep up and down. The rate also affects the amount of pitch modulation. By increasing the speed, the flanger will have to sweep through the depth in less time.

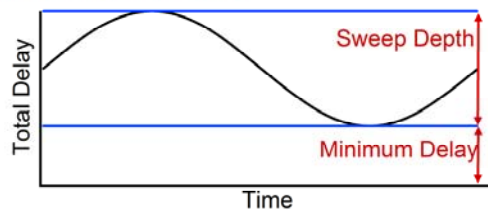
# Sweep Depth

- how wide sweep is in terms of delay time
- maximum delay = minimum delay + sweep depth
- Notch positions determined by total delay
- large depth →
  - notches in frequency response sweep over large area
  - pitch modulation effect grows
  - flanger reads faster or slower to change delay in same amount of time
- As delay increases, notch frequencies decrease,
  1. set minimum delay for maximum position of first notch
  2. adjust sweep depth to set minimum position

plucked electric guitar string, first flanged with depth of 2 ms, then with 6 ms. Latter varies more in pitch



relationship between sweep depth and delay parameters



The sweep depth determines how wide the sweep is in terms of delay time - essentially the width of the LFO. This sweep depth is the maximum additional delay that is added to the signal in addition to the delay in the delay parameter. It determines how low the first notch in the frequency response will reach. A small value for the depth will keep the variance in the delay time small, and a large value will cause the notches in the frequency response to sweep over a larger area. The figure shows how the delay and sweep depth parameters are related to the LFO. The maximum delay is the sum of the delay and sweep depth parameters.

As the sweep depth is increased, the pitch modulation effect mentioned above will become more noticeable. The flanger needs to read even faster or slower to change the delay in the same amount of time. The sound examples show how the pitch modulation effect varies with the depth.

Note that when you vary the minimum delay, both the upper and lower limits of the first notch are changed, but when you adjust the depth, only the lower limit is affected. So when you are setting up a flanger to sweep over a particular range, first set the delay so that the high point of the sweep is where you want it, and then adjust the sweep depth to set the low point of the sweep. When I refer to the sweep here, I'm talking about the notches, not the delay time. Remember that the parameters you set are controlling the delay, and that the notches result from this delay. As the delay increases, the notch frequencies decrease.

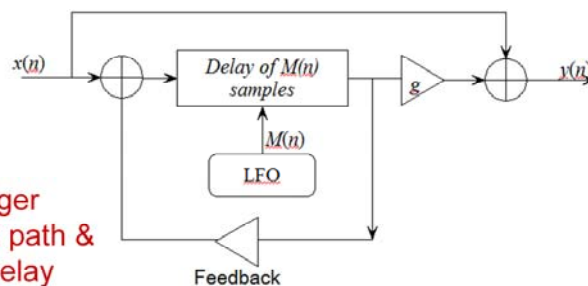
## Feedback/Regeneration

- Take portion of flanger's output and routing it to input
  - sets level of feedback from output to input of delay line
  - creating *feedback comb filter* in addition to feedforward comb filter
- large amount of feedback creates very 'metallic' and 'intense' sound
- as feedback gain  $\rightarrow 1$ , system can become unstable
  - Feedback in small amounts



Flanging with heavy feedback applied to strummed chord

more complex flanger including feedback path & LFO control over delay



Some units will give you an option for taking a portion of the flanger's output and routing it to the input. This control sets the level of feedback from the output to the input of the delay line, thereby creating a *feedback comb filter* in addition to the feedforward comb filter, in the same manner as in the creation of a Schroeder allpass filter.

A large amount of feedback can create a very 'metallic' and 'intense' sound, as the sound example demonstrates. A diagram for the flanger with feedback is shown.

Of course as the feedback gain approaches one, the system can become unstable, possibly resulting in overflow or clipping.

In some cases, you can also specify whether to add or subtract the feedback signal.

More generally, outputs of *any* sections can be fed back to the input (in small amounts) to produce different sounding effects.

## Stereo Flangers

- use 2 monophonic flangers in **quadrature phase**.
  - LFO in each monophonic flanger differ in phase by 90 degrees ( $\frac{1}{4}$  of wavelength).
- creates 'wider' sound
  - sound arriving at each of your ears is different



•mono source file processed by a flanger with stereo outputs.

One method for creating a stereo flanger is to use two monophonic flangers in **quadrature phase**. This simply means that the LFO in each monophonic flanger differ in phase by ninety degrees (or one-quarter of a wavelength). This technique creates a 'wider' sound because the sound arriving at each of your ears is different. The sound example presents a mono source file processed by a flanger with stereo outputs. Compare it to the flanged audio clips from the previous examples. If you have trouble hearing any difference, make sure your computer is generating a stereo output, and use a pair of headphones.

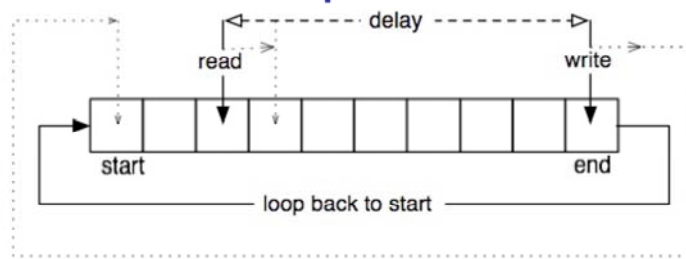
## Implementation issues

- Delay lines or circular buffers
  - chunk of memory used to store values
  - continuously read and write to that space
- Continuously changing delay time makes it interesting
- Samples are one sampling period apart
  - can't create delay line whose delay is not multiple of sampling period
- ✗ Read values out of delay line closest to required point
  - *zipper noise* - clicking in output as delay length changes.
- ✓ Use **interpolation**
  - estimate value of function between two points
  - Linear interpolation
    - read desired value from line between points
    - noisy
  - Cubic interpolation
    - more computationally expensive

In the digital realm, delays are implemented using delay lines or circular buffers (a chunk of memory is used to store a number of values, and you continuously read and write to that space). But implementing a continuously changing delay time makes it interesting. The changing delay time will require delay times that are not integer multiples of the sampling period (and the input signal is being sampled at multiples of this sampling period). Since the samples are always taken one sampling period apart, you can't create a delay line whose delay is not an integral multiple of the sampling period. You can try reading values out of the delay line closest to the required point, you will hear a clicking in the output as the delay length changes, also called "zipper noise".

You can get around this by using **interpolation**. Interpolation is simple estimating a value of a function between two points. In this case, we have two samples and we want to get a value in between them. A simple method of interpolating is to connect the points with a line, and then read the desired value from that line. Although linear interpolation is simple, it can be noisy due to aliasing. It may be acceptable with very high sampling rates, but in some cases, a more computationally expensive interpolation method may be needed.

## Aside: linear interpolation on delay lines



```

writepos=0 //write pointer position, start at beginning of buffer
readpos=(writepos+D-initialdelay) //read pointer position
For n=0 to N-1 do
  prev=floor(readpos) //get previous integer index
  next=(prev+1)%D //get next integer index; need modulo for wraparound
  t=readpos - prev //get fractional position of read pointer
  y[n]=(1-t) delayline[prev]+tdelayline[next] //get fractional
  position of read pointer
  readpos=(readpos+updateread)%D //update read pointer for next time
  delayline[writepos]=x[n] //write to delay line
  writepos=(writepos+1)%D //update write pointer for next time

```

We can implement a delay using a buffer of samples as a holding area, called a delay line. If the buffer has  $D$  samples, it can implement delays up to  $D$  samples long. The circular buffer is the standard way to accomplish this. Note the separate read and write pointers. This diagram depicts a variable length delay line, with a maximum possible delay of  $D=10$  samples. Memory is finite, hence the wraparound trick. The delay is the separation of the read and write pointers. Stepping off the end is equivalent to going back to the first slot in the buffer. The dotted arrows depict an update of the pointer positions by one sample, as everything shuffles right and around the loop. In general, read pointers and write pointers can be independent. This could also describe reading from a preloaded sound file in memory, with just a read pointer moving around the data array.

In standard operation of a delay line, both pointers increment by one sample in the buffer with each sample of output calculation, wrapping around to the start whenever they 'fall off the end' of the delay line. If an exact delay of  $D$  samples, the whole length of the line, is all we ever need, the pointers can point to the same place, reading an output sample before replacing the buffer contents at the current index with a new input sample. A variable length delay line is implemented by letting the read pointer shift about to different separations from the write pointer. Chasing at different distances is equivalent to particular delays. This gives us tools for basic delay effects units; by using multiple read pointers, at a set of desired delays or 'taps' you'd get a multi-tap delay line.

If a read pointer overtakes a write pointer, you are trying to read data that hasn't yet been written. Since the buffer is circular, there will be a jump from reading back data in the near past, to reading back data in the distant past. If the write pointer overtakes the read pointer, then you suddenly jump from having a long delay to a very short delay. This establishes limits on any variation of the delay, via the update speeds for the pointers. Possible rates of change depend on the relative position. The read pointer can always be safely shifted to new offsets which correspond to new delay times up to the length of the delay line as the update rate of the read pointer is the same as that of the write pointer. If the read pointer moves at a constant rate that is any slower or faster than the update rate of the write pointer, then there will eventually be an overtake. If slower, write overtakes read, if faster, read overtakes write. To avoid this, the read pointer must be working without the interference of a write pointer (simple playback) or be varying its rate either side of the update rate while maintaining the same mean speed (a type of modulation which leads to various standard effects).

The code gives an example of reading and writing from a delay line with linear interpolation on the read pointer position.

## Issues with Flanging

- *Only* uniformly spaced notches
  - ear processes sound over logarithmic frequency scale
  - exponentially spaced notches sound more uniform perceptually
- notches flanger produces could coincide exactly with fundamental and overtones
  - severe amplitude modulation
  - use flangers on noise-like and percussive signals
    - notch effect is much clearer
  - flanging often used on entire mix, rather than specific instrument within mix
  - For harmonic signals, create *non-uniform* moving notches

Phaser → *Arbitrary spaced notches*

Note that flanging provides only uniformly spaced notches. This can be considered non-ideal for several reasons. First, the ear processes sound over a frequency scale that is more nearly logarithmic than linear. Therefore, exponentially spaced notches (uniformly spaced on a log frequency scale) should sound more uniform perceptually. Secondly, the uniform peaks and notches of the flanger can impose a discernible "resonant pitch" on the program material, giving the impression of being inside a resonant tube. Third, it is possible for a periodic tone to be completely annihilated by harmonically spaced notches if the harmonics of the tone are unlucky enough to land exactly on a subset of the harmonic notches. In practice, exact alignment is unlikely; however, the signal loudness can be modulated to a possibly undesirable degree as the notches move through alignment with the signal spectrum. For this reason, flangers are best used with noise-like or inharmonic sounds. For harmonic signals, it makes sense to consider methods for creating non-uniform moving notches.

There is an interesting possibility when using a flanger with a musical instrument. For non-percussive instruments that generate a pitch and the harmonic overtones, the notches that the flanger produces could in theory coincide exactly with the fundamental and the overtones, eliminating the sound of the instrument altogether. In practice, an instrument won't disappear entirely, but there can be severe amplitude modulation. For this reason, some favor using flangers on percussive and noise-like signals (and this is partly why a drum loop was used in the sounds above. The notch effect is much clearer). In fact, flanging is often used on an entire mix, rather than a specific instrument within a mix.

Flanging uses a delay line to create a set of equally spaced notches in the audio spectrum. Phasing uses a set of notches as well, but the spacing of them can be arbitrary. The phaser uses allpass filters or notch filters. A set of notches is used to process the input signal. Although this effect does not rely on a delay line, it is often considered to go along with delay-line based effects because the sound effect is similar to that of flanging. In this respect, flanging is a type of phasing, but using the simplest allpass filters.

# Phasing

- Creates 1 or more notches in frequency domain that eliminate sounds at notch frequencies
  - ⊗ Sometimes called phase shifter
- notches created by filtering signal
  - mixing filter output with input signal
- filters can be designed so we can independently control
  - location of each notch
  - number of notches
  - width of notches

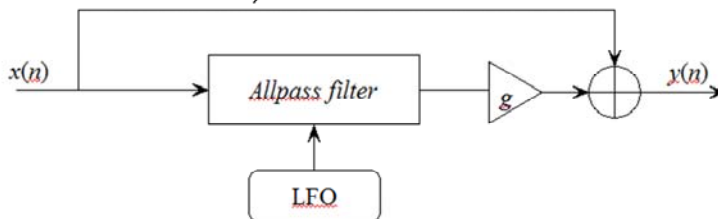


Picked chord sequence  
dry followed by same  
sequence with phase  
shifting applied

The phase shifter (or phaser) achieves its distinctive sound by creating one or more notches in the frequency domain that eliminate sounds at the notch frequencies. The notches are created by simply filtering the signal, and mixing the filter output with the input signal. The filters can be designed so that we can independently control the location of each notch, the number of notches, and even control the width of the notches. This can lead to many interesting sonic possibilities as presented in the sound example.

## How it Works

- notches implemented using *allpass filters*
- allpass filter passes all frequencies
  - all frequencies appear in output with no attenuation or amplification
  - sine wave into allpass filter → see sine wave at output with same amplitude
- add filter output to input signal.
  - amount of filtered signal that appears in output set by *depth* (also called *mix* or *level*) control



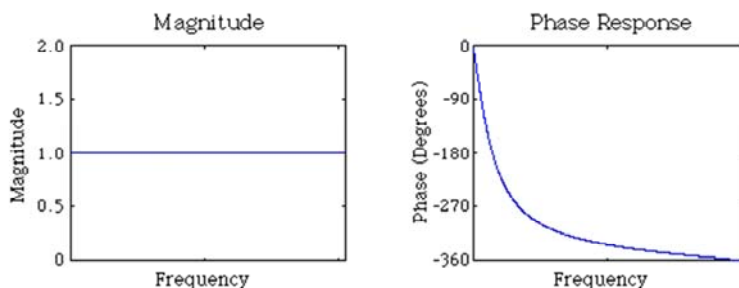
- filter passes all frequencies equally
  - how does allpassfilter alter sound?
  - where do notches come from?

The notches needed for phase shifting (or simply called phasing) are most often implemented using a special group of filters called **allpass filters**. As the name implies, the allpass filter passes all frequencies - that is it allows all frequencies to appear in the output with no attenuation or amplification. So if you were to put any sine wave into the allpass filter, you would see a sine wave at the output, with the same amplitude it was at the input. To complete the phase shifter, we just add the filter output to the input signal, as in the Figure. The amount of the filtered signal that appears in the output is set by the mix level control.

Well that's interesting, but if the filter passes all frequencies equally, how does it alter the sound, and where do the notches come from? Well there's one other characteristic of the filter we haven't mentioned yet, and that is the filter's phase response.

## Allpass filter response

- two important characteristics of any linear filter
  - magnitude response
    - relative amplitudes of input & output signals
    - for allpass filter, magnitude response = 1 for all frequencies.
  - Phase response (phase lag)
    - relative alignment of 2 signals in time
    - do they both cross zero or reach maximum values at same time



MSP Code:  
[allpass](#)  
[zeros &](#)  
[poles](#)

magnitude and frequency response for 2<sup>nd</sup> order allpass filter.  
 Only phase varies, all frequencies passed with gain 1.

If the filter passes all frequencies equally, how does it alter the sound, and where do the notches come from? Well there's one other characteristic of the filter we haven't mentioned yet, and that is the filter's phase response. The **phase response** is kind of difficult to explain, so let's keep things relatively simple here. Let's say you're given a block box, and you don't know what circuit is inside of it. You can test it by putting a sine wave generator on the input and then look at the output and the input together on an oscilloscope. There are two important characteristics that can be observed. The first is the relative amplitudes of the two signals (which is referred to as the magnitude response). Again, for the allpass filter, these amplitudes are equal, so the magnitude response is one for all frequencies. The other important characteristic is the relative alignment of the two signals in time, i.e. do they both cross zero or reach the maximum values at the same time. The difference between the two signals is the **phase lag**, or the phase response. The black box is altering, or shifting, the phase of the input signal - hence the name 'phase shifting.'

The phase lag is usually measured in fractions of a wavelength rather than an amount of time. A single cycle of the wave is 360 degrees (regardless of the period or frequency of the sine wave), so 90 degrees is a quarter of a cycle, 180 degrees is half a cycle, and so on.

All practical filters have a phase response that changes with frequency, and the figure is a plot of the magnitude and phase response of one possible allpass filter. One interesting case is a linear phase response (i.e. the phase response is a straight line, angled downwards). In this case, doubling the frequency means doubling the phase lag. The wavelength of the doubled frequency is half that of the original. This basically keeps all the frequency components aligned in time - it delays the signal. So the pure delay (no feedback or mixing with the original signal) is one type of an allpass filter. Linear phase is desirable in some applications so that all frequencies are kept together in time and the sound isn't colored by 'phase distortion.'

## Notches

- To create notches, mix allpass filter output with input
  - At some frequencies, phase lag will be  $180^\circ$ 
    - equivalent to taking negative of input.
    - cancel frequency components in input
  - # notches determined by filter complexity
  - Frequencies near notch also attenuated
- Filter with nonlinear phase response delays signal
  - not all frequencies delayed by same amount
- Linear phase case - pure delay
  - phase response hits values of  $-180^\circ$  at equally spaced frequencies
  - mix delayed copy with original → notches at equally spaced frequencies
  - exactly how flanger operates

Now we are ready to understand where the notches come from. To create the notches, we only need to mix the allpass filters output with the input. Why? At some frequencies, the phase lag introduced by the filters will be 180 degrees, which is equivalent to taking the negative of the input. When you mix this signal with the input, those frequencies that experience 180 degrees of phase lag will exactly cancel with those frequency components in the input, and that is the notch. Frequencies near the notch will also be attenuated somewhat. Essentially, a filter with a non-linear phase response, though it may be hard to believe, is delaying the signal, but not all frequencies are delayed by the same amount.

By using allpass filters that do not have a linear phase characteristic, we can distort the phase response to produce a notch at whatever frequency we choose. The number of notches is determined by how complex the filter is.

Going back to the linear phase case (the pure delay), the phase response hits values of -180 degrees minus multiples of 360 degrees (-180, -540, -900, -1260, etc.) at equally spaced frequencies. So when we mix a delayed copy of the signal with the original, there will be notches at equally spaced frequencies. This is exactly how the flanger operates, and thus, the flanger is merely one type of a phase shifter.

## Cascaded allpass filters

- chain additional filters to create more notches
  - series combination of allpass filters is allpass filter
  - phase response for chain of filters is sum of each filter's phase response
  - unlike flanger, can control width of notches if we want to
- notch frequencies should change exponentially
  - when notches move up, double notch frequency at each step, taking larger and larger steps
  - halve notch frequency at each step to return to low point

### MSP Code allpass cascade

### MSP Code driven allpass cascade & driven exponential poles & zeros



- 2 segments of phase shift applied to distorted guitar
  1. notches sweep exponentially along frequency axis
  2. notch sweep is linear
- Listen to notches at beginning (low point)
  - In 1, notches move more slowly at first compared to 2<sup>d</sup> clip

We can also exploit another property of allpass filters to increase the number of notches - a series combination of allpass filters is itself an allpass filter. So we can start with a simple filter, and then chain additional filters to create more notches, and the phase response for the chain of filters is simply the sum of each filter's phase response. Also, unlike the flanger, we can control the width of the notches if we want to. But this parameter is often determined in the design stage of a product and can't be adjusted by the user.

One remaining point to cover is how the notches sweep over time. For the chorus and flanging effect, we use a simple LFO to control the delay time. But with the phaser, it's generally preferred to have the notch frequencies change exponentially. For example, when the notches are moving up, you might double the notch frequency at each step, taking larger and larger steps, and then once you've reached the high point of the sweep, you halve the notch frequency at each step to return to the low point.

You won't have control over how the notches sweep in most commercial products, just how quickly the notches sweep back and forth and the range they sweep over. Of course, there is not really a right or wrong way to do things, and for the circuit hacker or programmer, there are plenty of possibilities to explore. In this spirit, the sound example offers two clips of phasing on a distorted guitar with an exponential and linear sweep.

## Common Parameters

- **Mix level (dry to wet)**
  - amount of filter output added to sound
  - as it increases, depth of notches increases
  - When mix is completely wet, notches reach down to 0
- **Sweep Depth (range)**
  - how far notches sweep up and down in frequency
- **Speed or Rate**
  - Frequency of LFO
  - sets how many times notches sweep up and back down per second only



Distorted guitar sound, processed by phaser with 1 octave sweep depth, then 2 octave sweep depth. rate is same in both cases

[MSP Code simple phaser with exponential sweep](#)

The mix level controls the amount of the filter output that is added to the sound. As it increases, the depth of the notches increases as well. When the mix is completely wet, then the notches reach all the way to zero.

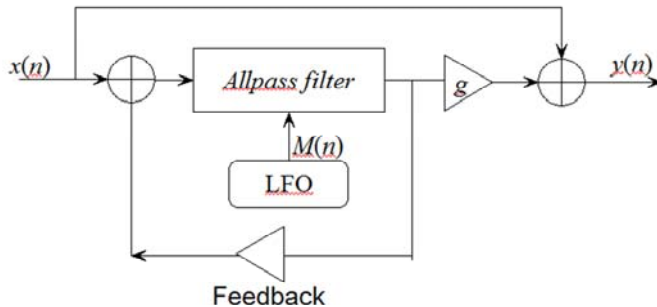
The sweep parameter is used to control how far the notches sweep up and down in frequency. In some cases, you may be able to select actual frequency values, and in other cases, the base frequency may be set to some value and you can only control how far from that frequen

The speed parameter simply controls how quickly the notches sweep up and down over the frequency range. The rate sets how many times the notches sweep up and back down per second only. The speed at which the actual notches are moving is determined by this rate control, the sweep depth, and the sweep pattern.

the base frequency may be set to some value and you can only control how far from that frequency the sweep will go The sound example demonstrates the sweep depth on the phaser.

# Feedback/Regeneration

- phase shifting effects made more intense by using feedback
  - adding part of filter output to input
- Some units allow negative feedback gains
  - subtract output from input
- When several stages of allpass filters, can use feedback on each individual stage as well as entire structure



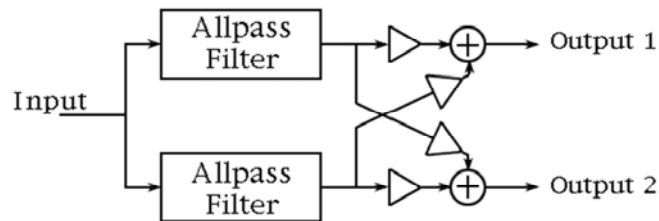
phase shifter with added feedback path. feedback gain can be negative as well as positive

strummed guitar using phaser with mild feedback, followed by higher feedback setting

The phase shifting effects can be made more intense by using feedback - adding part of the filter output to the input, as shown, and in the audio clip. Some units may also allow you to have negative feedback gains, which is equivalent to subtracting the output from the input. When working with several stages of allpass filters, you can conceivably use feedback on each individual stage as well as the entire structure.

# Stereo Phase Shifter

- like stereo flanger using 2 filters with notches at different frequencies
- more complex sound by selectively mixing outputs of 2 filters
  - Creates additional notches that can exist on single output.



generalized diagram of a stereo phaser

- spatial phaser
  - feed-across gains set to 0
  - each output fed to speaker
  - notches exist only in air and heard by moving about room



3 different stereo phaser configurations applied to distorted guitar sound

A stereo phaser could be constructed along the lines of a stereo flanger/chorus unit by using two filters with notches falling at different frequencies. It's also possible to create more complex sounds by selectively mixing the outputs of the two filters as well which can create additional notches that can exist on a single output. This general structure is depicted. If the feed-across gains are set to zero and each output fed to a speaker, you will have a 'spatial phaser' where the notches exist only in the air and can be heard by simply moving about the room. The sound example gives a sampling of some stereo phase shifting effects.

## Source material

- “Effects explained” available online at Harmony Central
- “Flanging“ by Julius O. Smith III  
<http://ccrma.stanford.edu/~jos/pasp/Flanging.html>  
(and related web pages)
- Digital Audio Effects, edited by Udo Zoelzer
- Software used in demonstrations is Max MSP
- This presentation intended for education only.