Waveshaping & Karplus-Strong Synthesis

Prof Eduardo R Miranda
Varèse-Gastprofessor
eduardo.miranda@btinternet.com

Electronic Music Studio TU Berlin
Institute of Communications Research
http://www.kgw.tu-berlin.de/

- Topics:
  - Waveshaping Synthesis
  - Karplus-Strong Synthesis

Waveshaping Synthesis

• Also termed *non-linear distortion* or *non-linear processing*

• Creates composite spectra by applying distortion to a simple sound (e.g., sinusoid)

• Works by passing a sound through a unit that distorts its waveform, according to a user-specified directive

• The distortion unit is called *waveshaper* and the user-specified directive is called *transfer function*

• Metaphor to understand the technique: imagine a note played on an electric guitar connected to a vacuum-tube amplifier

Metaphor to understand the technique:

• Imagine a note played on an electric guitar connected to a vacuum-tube amplifier

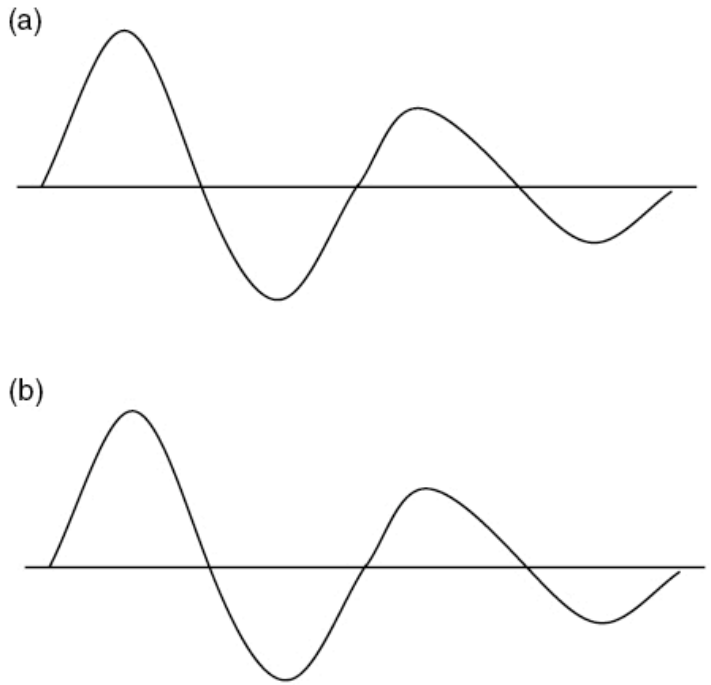
• If the volume of the amplified is increased to its maximum, the vacuum-tubes will be saturated and the sound will clip

• If the amplitude of the note is increased at its origin, before entering the amplifier, then the output will clip even more

• If the note is a sinusoid, the louder the input, the more squared the output wave will be

• If the note has a complex spectrum, then the output will be a signal blurred with distortion

If the “instrument” produces a sustained note, gradually losing amplitude, then the output will be a sound whose amplitude remains (almost) constant, but the waveform will vary continuously.



Amplitude sensitivity: a key feature of waveshaping synthesis.

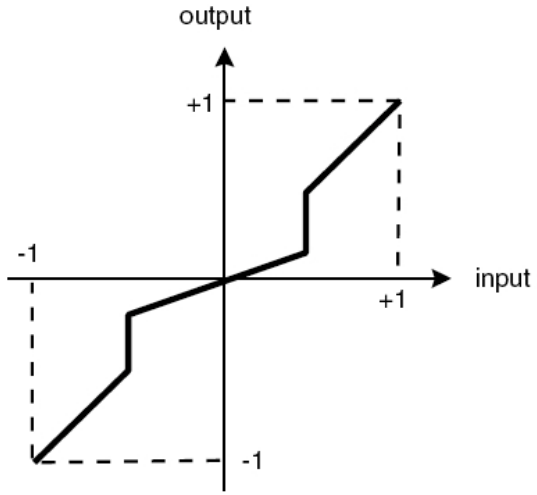
The amount of amplification or distortion is proportional to the level of the input signal.

Waveshaper: a processor that can alter the shape of the waveform passing through it

• Non-linear processor: the relationship between input and output signals depends upon the amplitude of the input signal and the nature of the non-linearity

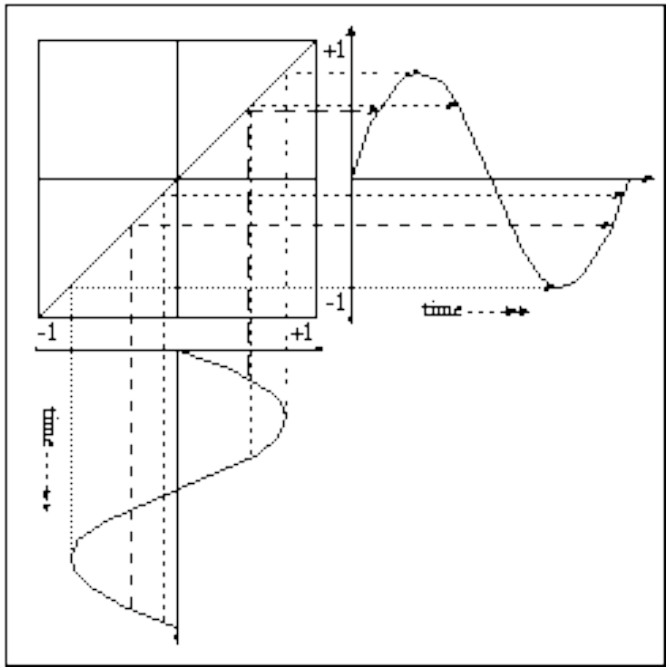
• In most cases, non-linear processors modify the waveform of the input signal, generally resulting in an increase in the number and intensity of its partials

• The waveshaper is characterised by a transfer function, which relates the amplitude of the signal at the output to the signal at the input

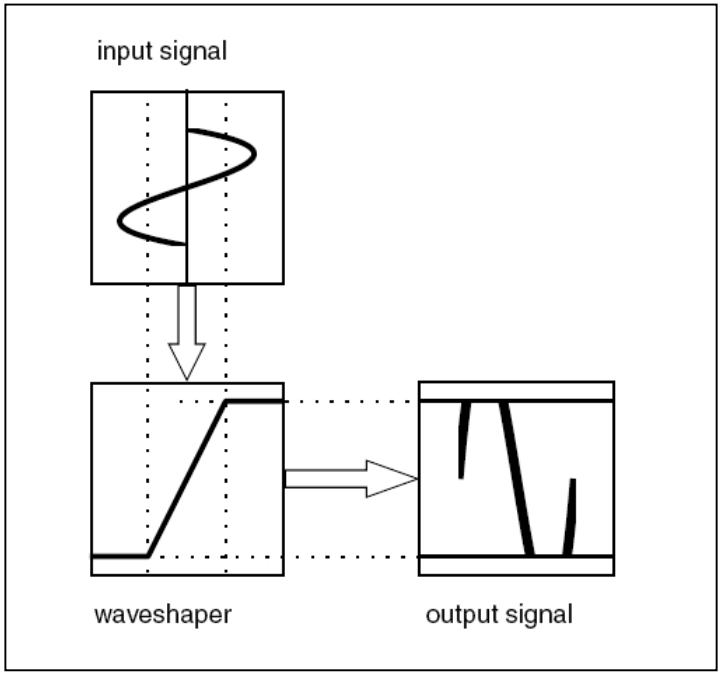


If the transfer function is a straight diagonal line from -1 to +1, the output is an exact replica of the input. Any deviation from a straight diagonal line introduces modifications

A transfer function: the amplitude of the input signal is plotted on the horizontal axis and the output on the vertical



(Fischman)



• The success of a waveshaping instrument depends upon the choice of a suitable transfer function

• Drawing transfer functions would intuitively be one way to experiment with waveshaping, but it might be difficult to predict the results systematically

• Mathematical tools for constructing waveshapers: *polynomials* and *trigonometric functions*

• Well known polynomials for spectral design: *Chebyshev polynomials of the first kind*

Polynomials

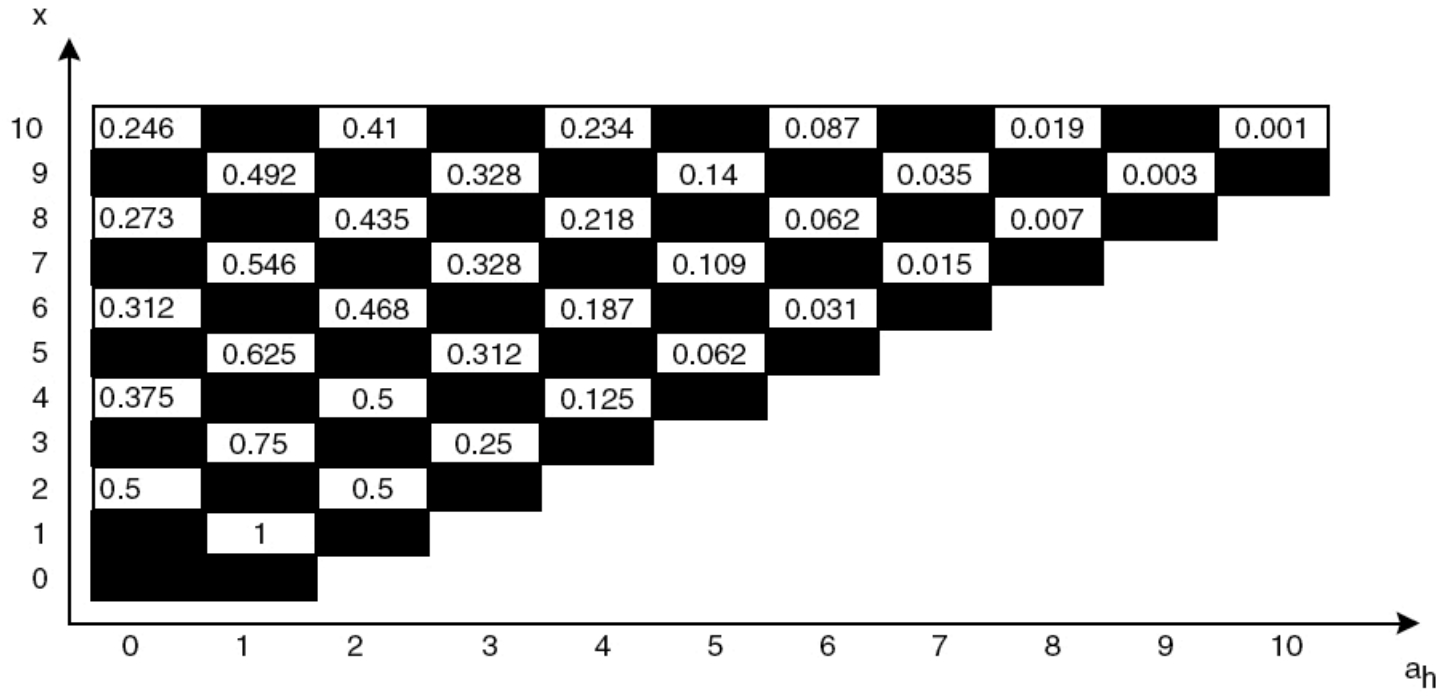
Transfer functions using polynomials are generally the best.

The amplitude of the signal input to the waveshaper is represented by the variable *x* and the output is denoted by *F*(*x*), where *F* is the function and *d* are amplitude coefficients:

$$F(x) = d_0 + d_1 x + d_2 x^x + \ldots + d_n x^n$$

It is possible to predict the highest partial that will be generated by the waveshaper: if the input is a sinusoid and the transfer function a polynomial of order *n*, then the waveshaper produces partials up to the n^{th} partial.

When the amplitude of the input sinusoid is = 1, the amplitudes of the partials can be calculated according to the following table:

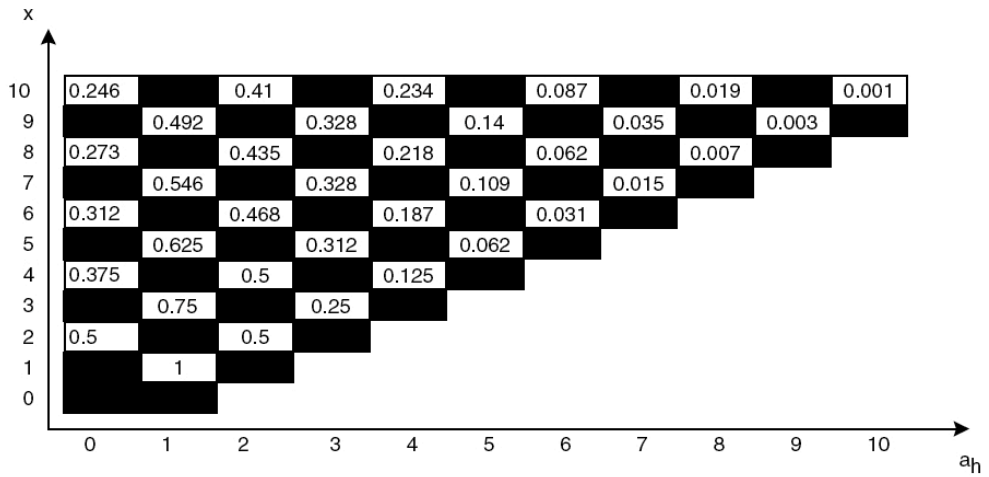


A term does not produce more harmonics than its exponent.

Example: the function below produces the fundamental + 1st, 2nd, 3rd and 4th partials

$$F(x) = x + x^2 + x^3 + x^4$$

The amplitudes are calculated as the sum of the contribution of each term:



$$a_{h0} = 0.5 + 0.375 = 0.875$$

$$a_{h1} = 1 + 0.75 = 1.75$$

$$a_{h2} = 0.5 + 0.5 = 1$$

$$a_{h3} = 0.25$$

$$a_{h4} = 0.125$$

Chebyshev polynomials of the first kind

$T_k(x)$ k = order of the polynomial
x = sinusoid

When a cosine wave with amplitude = 1 is applied to $T_k(x)$, the resulting signal is a sinewave at the k^{th} harmonic.

Example: if a sinusoid of amplitude = 1 is applied to a transfer function given by k = 7, the result will be a sinusoid at 7 times the freq of the input. (Amplitude = 1 refers to the max input amplitude that the waveshaper can manage.)

Because each separate polynomial produces a particular harmonic of the input signal, a certain spectrum composed of various harmonics can be obtained by summing a weighted combination of polynomials.

Chebyshev polynomials from T_1 to T_{10} are given as follows:

$$T_1(x) = x$$
$$T_2(x) = 2x^2 - 1$$
$$T_3(x) = 4x^3 - 3x$$
$$T_4(x) = 8x^4 - 8x^2 + 1$$
$$T_5(x) = 16x^5 - 20x^3 + 5x$$
$$T_6(x) = 32x^6 - 48x^4 + 18x^2 - 1$$
$$T_7(x) = 64x^7 - 112x^5 + 56x^3 - 7x$$
$$T_8(x) = 128x^8 - 256x^6 + 160x^4 - 32x^2 + 1$$
$$T_9(x) = 256x^9 - 576x^7 + 432x^5 - 120x^3 + 9x$$
$$T_{10}(x) = 512x^{10} - 1280x^8 + 1120x^6 - 400x^4 + 50x^2 - 1$$

Each separate polynomial produces a particular harmonic of the input signal. A spectrum containing various harmonics can be obtained by adding a weighted combination of polynomials, one for each desired harmonic.

Example: A spectrum containing the 1st, 2nd 3rd and 5th partials:

$$F(x) = w_1T_1(x) + w_2T_2(x) + w_3T_3(x) + w_5T_5(x)$$

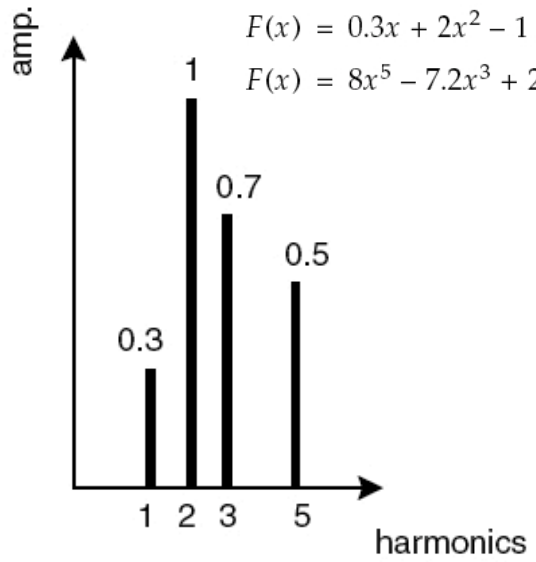
The weighting values (w_1, w_2, w_3 and w_5) correspond to the relative amplitudes of the respective harmonic components.

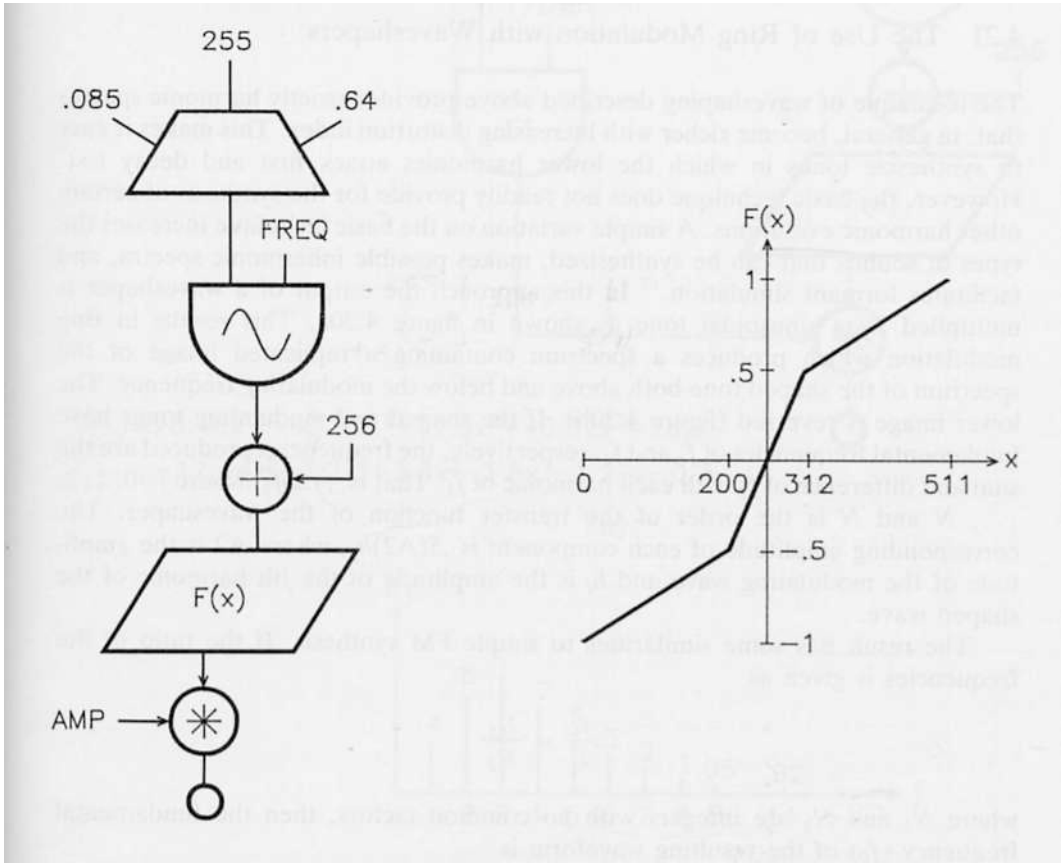
$$F(x) = 0.3T_1(x) + 1T_2(x) + 0.7T_3(x) + 0.5T_5(x)$$

$$F(x) = 0.3(x) + 1(2x^2 - 1) + 0.7(4x^3 - 3x) + 0.5(16x^5 - 20x^3 + 5x)$$

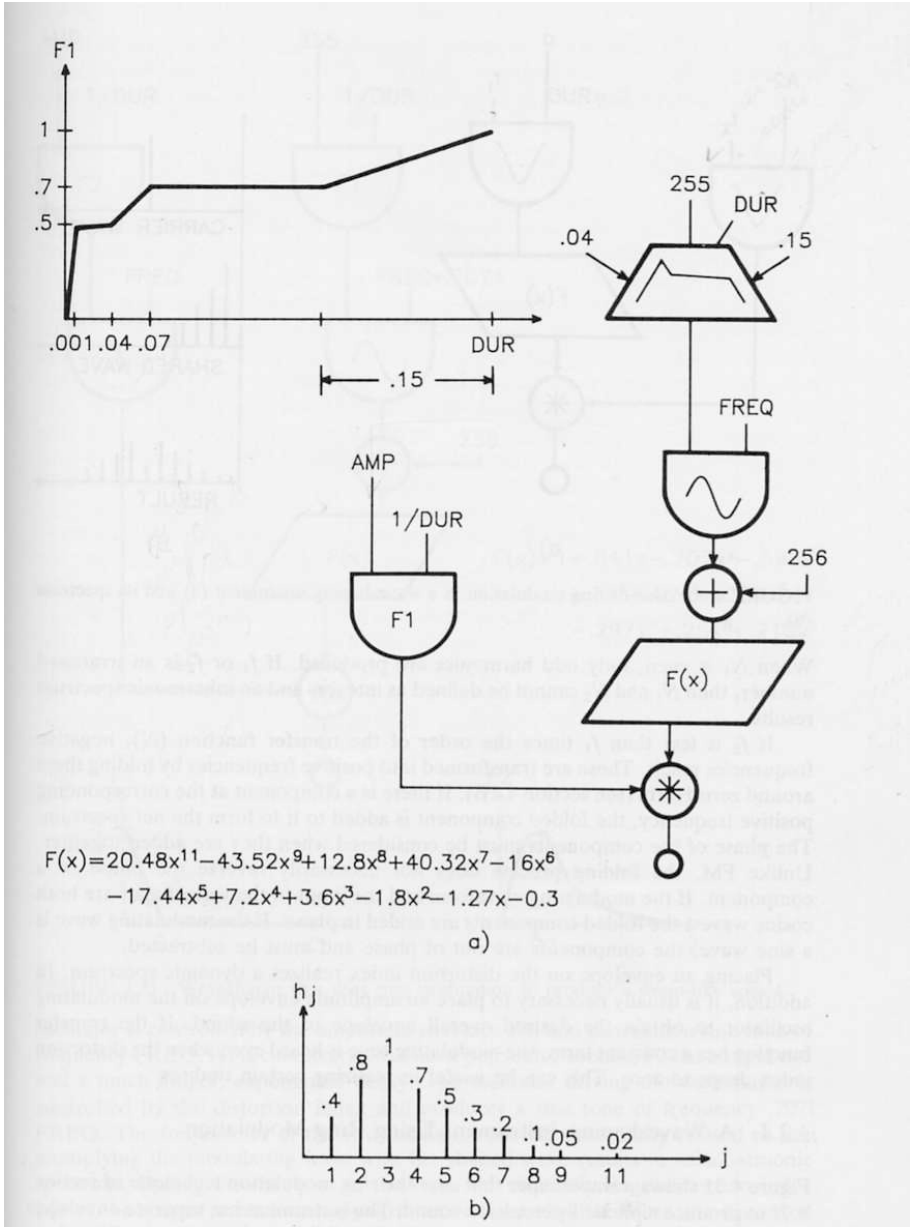
$$F(x) = 0.3x + 2x^2 - 1 + 2.8x^3 - 2.1x + 8x^5 - 10x^3 + 2.5x$$

$$F(x) = 8x^5 - 7.2x^3 + 2x^2 - 0.7x - 1$$





(Dodge & Jerse)


F1
1
.7
.5
.001 .04 .07
DUR
.15
255
DUR
.04
.15
FREQ
AMP
1/DUR
F1
256
F(x)
F(x)=20.48x^11−43.52x^9+12.8x^8+40.32x^7−16x^6
−17.44x^5+7.2x^4+3.6x^3−1.8x^2−1.27x−0.3
a)
hj
.4
.8
1
.7
.5
.3
.2
.1
.05
.02
1 2 3 4 5 6 7 8 9 11
j
b)

(Dodge & Jerse)

How to plot Chebyshev transfer function?

Most synthesis programming languages provide a “unit” for implementing the waveshaper as a table, or array, whose size is compatible with the maximum amplitude value of the input signal.

For instance: to fully wavehape a sinusoid oscillating between -4096 and + 4096 samples, the transfer function should span a table containing 8193 values (4096 X 2 plus 1 for zero crossing).

In Csound:

Use the table unit. The value of a sample is used as the index of the function table.

GEN13 - Stores a polynomial whose coefficients derive from the Chebyshev polynomials of the first kind.

Example: fischman1112.orc

This instrument demonstrates the difference between linear and nonlinear processing

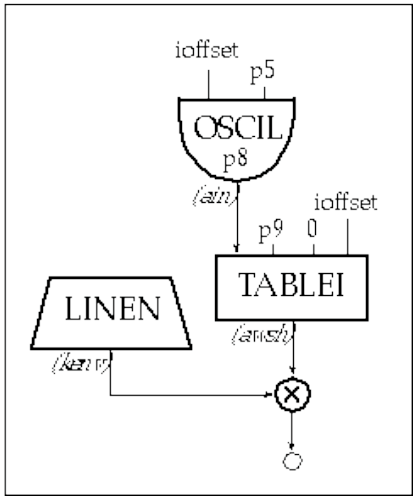
The waveshaping table (**tablei**) processes the signal **ain**

Since **ain** can be positive or negative, the upper half of the table processes the positive samples and the lower half the negative smaples

So, the offset needs to point to the middle of the table: **ftlen(p9)/2 -1** (-1, because the samples are numbered from 0)

The tables for the waveshaping:

f2 = linear
f3 = “nearly” linear
f4 = discontinous function



(Fischman)

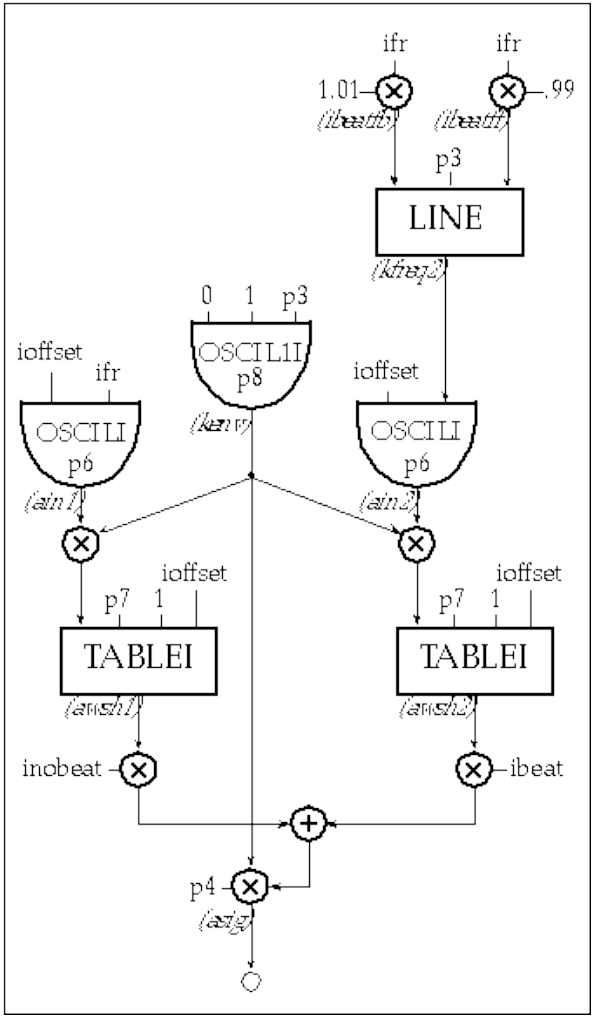
Example: fischman1113.orc

- Uses two waveshapers

a) Processes a sinewave of constant frequency
b) Processes a sinewave that varies its frequency during the duration of the note from 1.01*f* to 0.99*f*

- The outputs of the waveshapers are summed in a relative proportions of 0.8:0.2 (**inobeat:ibeat**), which produces a beating pattern (to make the sound more interesting)

- The offset (**ioffset**) is equal to 0.5 and not half of the size of the wavetable. This is because the table statements are used in normalized mode (the waveshaped signal varies between -0.5 and +0.5)



(Fischman)

Example: dodge_428.orc

(Dodge & Jerse)

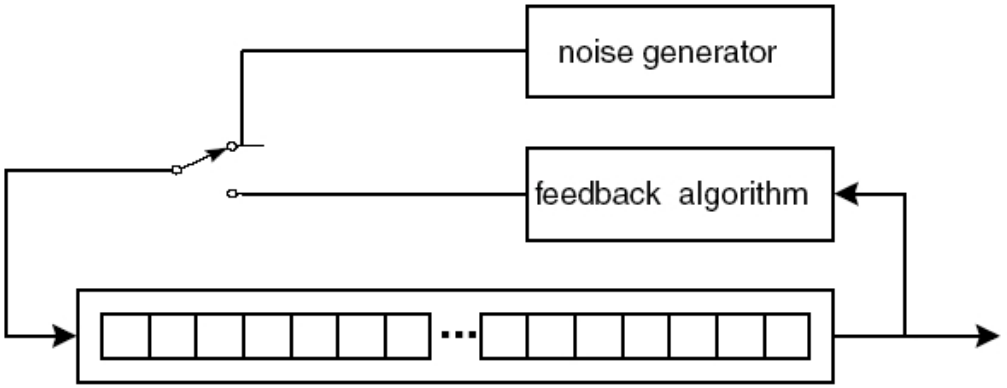
Karplus-Strong Synthesis

• Devised by Kevin Karplus and Alex Strong, as Stanford University

• Uses a time-varying lookup table to simulate the behaviour of a vibrating medium

• It starts with a lookup table of a fixed length, filled with random samples. In this case the table functions as a queue of sample values, rather than as a fixed array (as it would be in the case of a simple oscillator)

• As samples are output from the right side of the array, they are processed according to a certain algorithm, and the result is fed back to the left side



• The algorithm for processing the samples defines the nature of the output sound.

• For instance, the averaging of the current output sample with the one preceding it in the array functions as a type of low-pass filter.

• This technique resembles the way in which sounds produced by most acoustic instruments evolve: they converge from a highly disorganised distribution of partials (as in the attack of a note) to oscillatory patterns (as in the sustain part of a note).

• The Karplus & Strong's original algorithm averages the current output sample of a delay line with the preceding one, and feeds the result back to the end of the delay queue.

• Produces convincing simulations of the sound of a plucked string.

Implementation:

Some packages provide a unit, which normally require 5 parameters:

• Amplitude

• Frequency

• Size of the recirculating lookup table (specified in HZ, usually the same as the frequency

• The nature of the initalisation

• The type of feedback method

In Csound:

pluck

ar pluck kamp, kcps, icps, ifn, imeth [, iparm1, iparm2]

Implementation without Csound's pluck unit (by Russel Pinkston):

```
;============================
instr 1
;p4 = amplitude
;p5 = pitch
;
icps = cpspch(p5)
asig init 0
kcount init 1/icps*kr ;loop counter
;----------------------
adel delayr 1/icps ;length od delay line is 1/cps
asig tone adel,sr/2 ;filter output of delay
if (kcount < 0) kgoto continue
;----------------------
kloop: ;firstly fill delay line with noise
asig rand p4,-1
kcount = kcount-1 ;decrement loop counter
;----------------------
continue:
delayw asig
if (kcount >= 0) kgoto kloop
outs asig, asig
endin
;============================
```

It firstly fills a delay line with noise

```
kloop:
asig rand p4,-1
kcount = kcount-1
...
delayw asig
if (kcount >=0) kgoto kloop
```

Then in feed the output of the delay line through a 1st order LFP (tone) and feed the result back into the delay line. The pitch of the sound (**icps**) defines the length of the delay line (**adel delay 1/icps**)

```
adel delayr 1/icps
asig tone adel,sr/2
...
out asig
```

pinkston

Implementation with Csound's pluck unit:

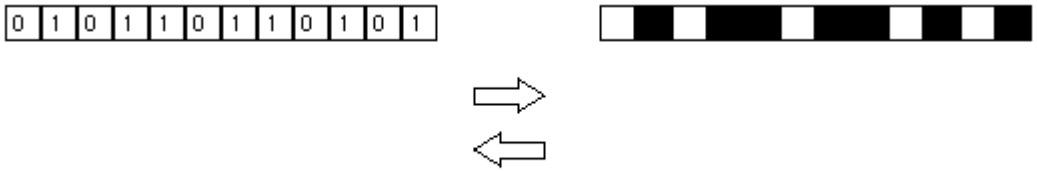
1. amplitude (*kamp*)
2. frequency (*kcps*)
3. size of the recirculating table-lookup (*icps*)
4. the nature of the initialization (*ifn*)
5. the type of feedback method (*imeth*)

- Usually ***kcps*** = ***icps***
- To initialise with random samples ***ifn*** = 0
- Csound offers 6 types of feedback. For the simple averaging method *imeth* = 1

Cellular automata lookup table

• Similar to Karplus-Strong, with the difference that the new samples of the delay line are computed using a cellular automaton, rather than the averaging method.

A cellular automata is implemented as a regular array of variables called cells. Each cell may assume values from an infinite set of integers and each value is normally associated with a colour, in this case 0 = white and 1 = black



1 if 111 then 0
2 if 110 then 1
3 if 101 then 0
4 if 100 then 1
5 if 011 then 1
6 if 010 then 0
7 if 001 then 1
8 if 000 then 0

Further reading:

“A Survey of Classic Synthesis Techniques in Csound” by Rajmil Fischman. In R. Boulanger (Ed.) The Csound Book. MIT Press. Pages 223-260.

“Efficient Implementation of Analog Waveshaping in Csound” by Michael Pocino. In R. Boulanger (Ed.) The Csound Book. MIT Press. Pages 565-573.

Karplus, K., and A. Strong 1983. "Digital Synthesis of Plucked-String and Drum Timbres." Computer Music Journal 7(2):43-55.