Physical Modelling

Physical modelling synthesis

The synthesis of sound by using a mathematical model: sets of equations and algorithms to simulate a physical source of sound.

- Sound is generated using model parameters that describe the physical materials used in the instrument and the user's interaction with it,
- For example, by plucking/bowing a string, or covering toneholes on a flute, clarinet etc.
- For example, to model the sound of a drum, there would be a formula for how striking the drumhead injects energy into a two dimensional membrane.

Physical Modelling: Examples

Hardware: Yamaha VL1 (1994), Roland COSM, Many since.

Software: Arturia Moog, PianoTeq

Examples of physical modelling algorithms:

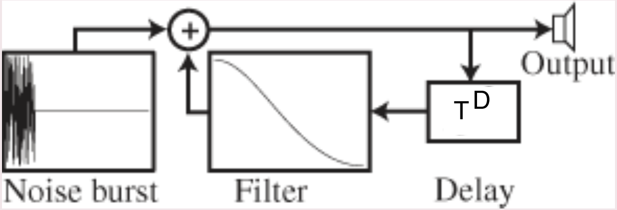
- Karplus-Strong strong synthesis (1971)
- Digital waveguide synthesis (1980s)
- Formant synthesis (1950s)

Physical Modelling Exampe

Karplus-Strong Algorithm

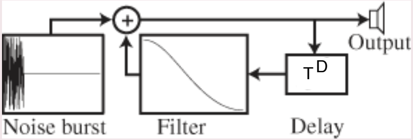
Simple Algorithm: *Makes a musical sound from noise*

- Loops a short noise burst through a filtered delay line to simulate the sound of a hammered or plucked string or some types of percussion.



- Feedback, Filtering and delay.
- Essentially **subtractive synthesis** technique based on a **feedback** loop similar to that of a **comb filter**.

Karplus-Strong Algorithm More Details:



- Input: A burst of white noise, L samples long, (can use other signal).
- Output signal and feedback into a delay line.
- Output of the delay line is fed through a filter -gain of the filter must be less than 1 at all frequencies, usually a first order lowpass filter
- Filtered output is simultaneously mixed back into the output and fed back into the delay line.

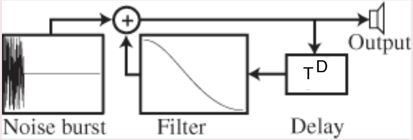
Physical Modelling Example

Karplus-Strong Algorithm Tuning

- Period of the resulting signal is the period of the delay line plus the average group delay of the filter;
- Fundamental frequency is the reciprocal of the period.
- Required delay D for a given fundamental frequency $F_1$is therefore calculated as:

$$D = \frac{F_s}{F_1}$$

where F_s is the sampling frequency.



Physical Modelling MATLAB Example

MATLAB Karplus-Strong Algorithm: karplus.m:

```
% ******* Constants and Other Parameters ******* %
fs = 44100;    % sampling rate
N = 80000;     % length of vector to compute
D = 200;       % delay line (or wavetable) length

% ******* Simple String Attenuation Filter ******* %
b = -0.99*[0.5 0.5];
z = 0;

% ******* Initialize delay lines ******* %
y = zeros(1,N);                    % initialize output vector
dline = 2 * rand(1, D) - 1.0;
ptr = 1;

figure(1); subplot(3,1,1);plot(dline);set(gca,'fontsize',18);
title('Original delayline');

subplot(3,1,2);plot(dline);set(gca,'fontsize',18);
title('Filter delayline step n');
```

Physical Modelling MATLAB Example (Cont.)

karplus.m

```
loopsound(dline,fs,fs/D);

subplot(3,1,3); plot(y); title('Waveform Step n');set(gca,'fontsize',18);

figure(1);
% ******* Run Loop Start ******* %
for n = 1:N,
  y(n) = dline(ptr);
  [dline(ptr), z] = filter(b, 1, y(n), z);
  % Increment Pointers & Check Limits
  ptr = ptr + 1;
  if ptr > D
    ptr = 1;
  end
```

Physical Modelling MATLAB Example (Cont.)

karplus.m

```
  if  mod(n,2000) == 0
      subplot(3,1,2);plot(dline)
      str = sprintf('Filter delayline step %d',n);
      title(str);
      subplot(3,1,3); plot(y);
      str = sprintf('Waveform Step %d',n);
      title(str);
      figure(1);
  end
end

% Scale soundfile if necessary
max(abs(y))
if max(abs(y)) > 0.95
  y = y./(max(abs(y))+0.1);
  disp('Scaled waveform');
end

figure(2);clf;plot(y); title('Final Step');set(gca,'fontsize',18);
sound(y',fs);
```

See also Ch5_7_Physical_Modelling_Synthesis.mlx for results

Physical Modelling MATLAB Example: Drum Sound

The basic algorithm is as follows:

- Start with wavetable X, of length p,
  - such that
    $$X(t) = +1/2(X(t-p) + X(t-p+1))$$
    with probability b, and
  - $$X(t) = -1/2(X(t-p) + X(t-p+1))$$
    with probability $1 - b$ for $t > p$.
- Since b introduces randomness into the sound, the initial wavetable can be anything from a completely random signal to a sine wave to a constant.
- The wavetable length p affects the decay rate of the sound (big = long decay) as well as the pitch somewhat (big = low pitch).
  - p should be in a range from about $150 - -500$.
- The probability b is called the **blend factor** and can range from 0 to 1.
  - $b = 1/2$ introduces the most randomness and produces the best snare sounds.
  - b near 0 simply averages the samples, and produces string-like sounds where p controls the pitch. Note: doesn't work for constant or sine wavetables.
  - b near 1 produces wierd electric crash cymbal-like sounds where most of the pitches die out quickly. Note: doesn't work for constant or sine wavetables.

See Ch5_7_Physical_Modelling_Synthesis.mlx for code and to hear results.

Crash cymbal sounds

Choose values:

- $b > 0.98$,
- $p = 200 - -800$,
- **random wavetable**
- **decaying envelope**

See Ch5_7_Physical_Modelling_Synthesis.mlx for code and to hear results

Metallic plink sounds

Choose values:

- $b > 0.98$,
- $p = 5 - -50$,
- **random wavetable**
- **Envelope decay usually needed for** $b = 1$

See Ch5_7_Physical_Modelling_Synthesis.mlx for code and to hear results

And back to string sounds

Choose values:

- $b < 0.05$,
- $p = 20 - -400$,
- **random wavetable**
- **Envelope decay usually needed for $b = 1$**

See Ch5_7_Physical_Modelling_Synthesis.mlx for code and to hear results

Some more examples

See Ch5_7_Physical_Modelling_Synthesis.mlx for

- FULL MATLAB (Demo) EXAMPLE: Generating Guitar Chords Using the Karplus-Strong Algorithm
  - Playing a Note on an Open String
  - Playing a Note on a Fretted String
  - Playing Guitar Chords
  - Guitar Strumming