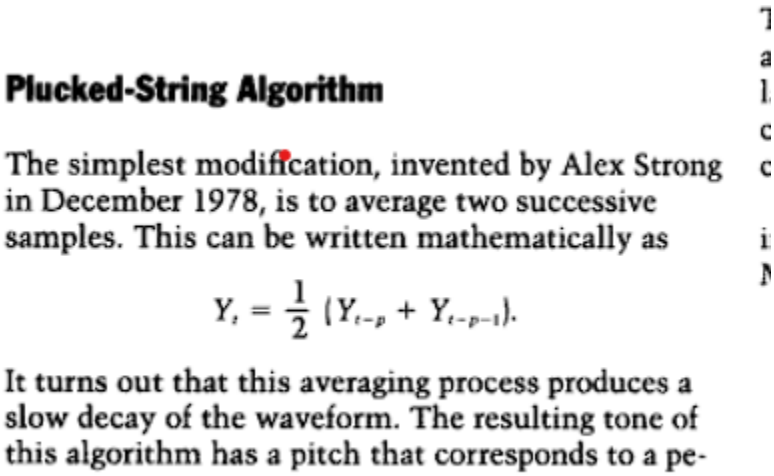
FDST Lab Final
Michael Xie

Project:Basic Implementation of the Karplus-Strong Algorithm

After the presentation, I realize that I am so bad at doing presentations, I learn a lot from doing presentations this semester.
I will put my notebook into sections.
Code is all construct by myself after reading these references:
1. Sound and Music - The Karplus-Strong Algortihm (stanford.edu)
2. Digital Synthesis of Plucked-String and Drum Timbres on JSTOR
3. Extensions of the Karplus-Strong Plucked-String Algorithm on JSTOR

Section in notebook:

1. Waveguide synthesis, repeating a picture/pattern/group of samples for x periods, we will have a signal at x Hz.
   In the first section, it is simple demonstration with this idea:
   A. We will generate a sample group, the values are random within the range of (-0.5,0.5)
   B. Then, we will connect 500 groups, but, we will do a calculation that every sample in each group will be 0.99 of the value of the corresponding sample in the previous group:
   Group[1][x] = Group[0][x] * 0.99
   …
   Group[i] [x] = Group[i-1][x] * 0.99 like this
   C. Then we get a signal already sounded like some sort of a plucked string
2. Now we will introduce the KS algorithm, instead of doing 0.99, it propose a formula with make the output signal more like sound of a plucked string.



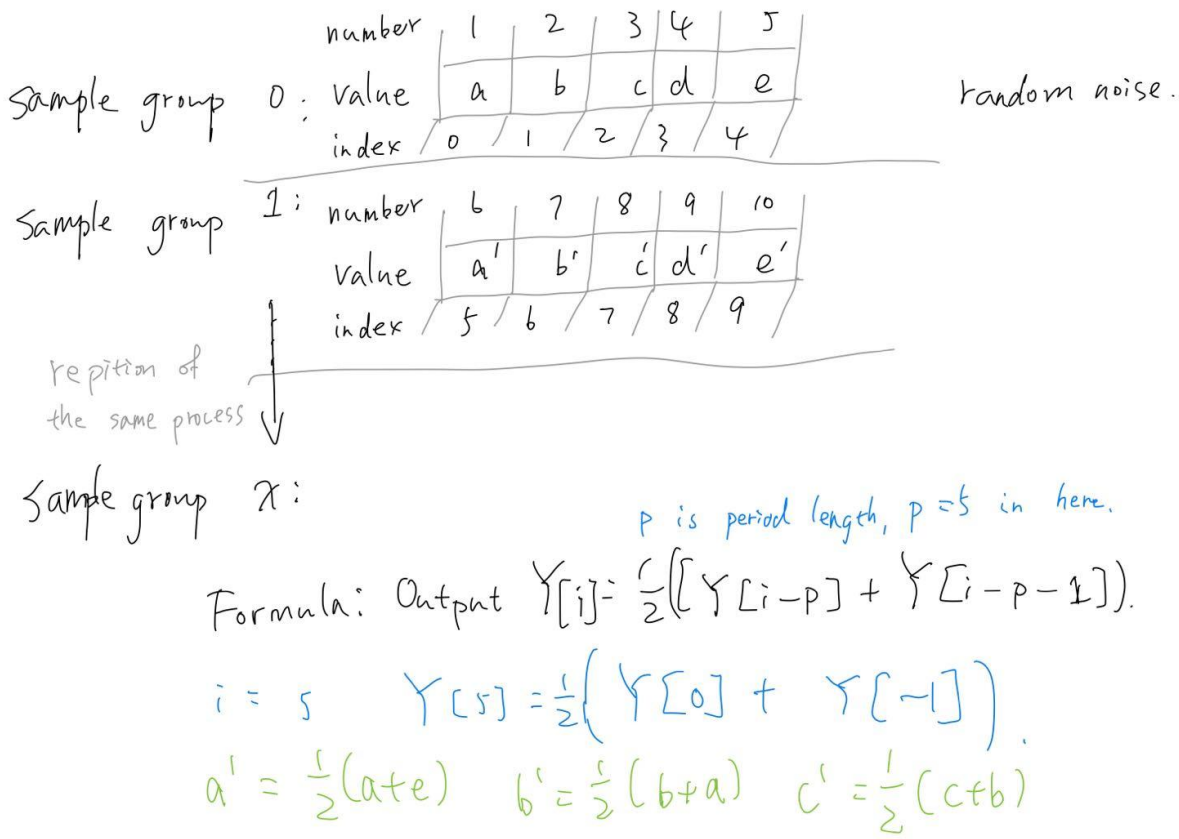
Plucked-String Algorithm

The simplest modification, invented by Alex Strong in December 1978, is to average two successive samples. This can be written mathematically as

$$Y_t = \frac{1}{2}\,(Y_{t-p} + Y_{t-p-1}).$$

It turns out that this averaging process produces a slow decay of the waveform. The resulting tone of this algorithm has a pitch that corresponds to a pe-

Instead of multiplying 0.99 for each sample, the KS will do:
average two successive samples in current group to get the samples in the next group

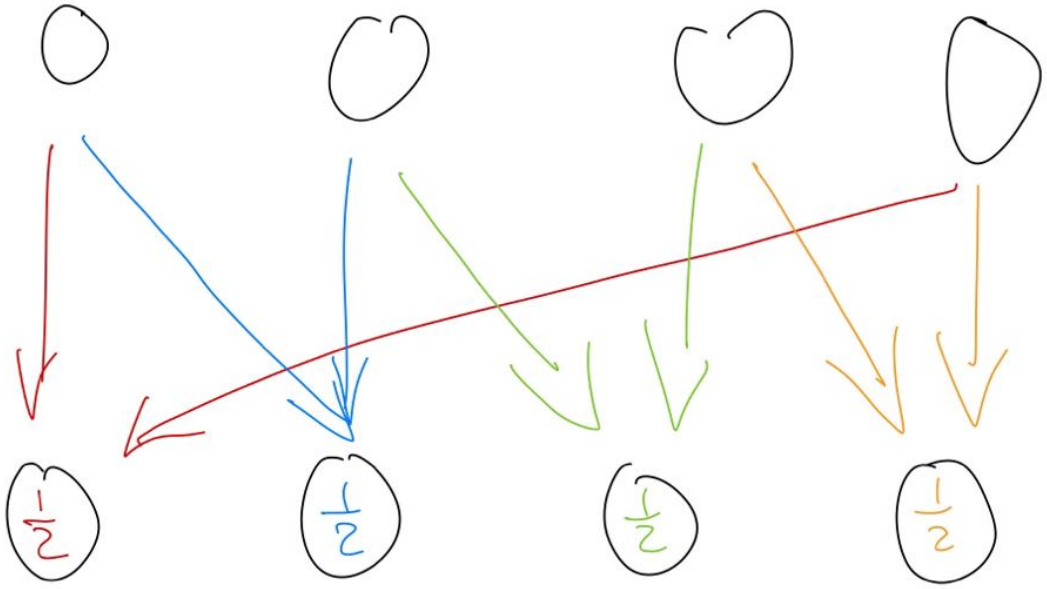


In my code:

```
# Apply the Karplus-Strong algorithm
for i in range(period_length, N):
    y[i] = 0.5 * (y[i - period_length] + y[i - period_length - 1])
```

Note that for our first group repetition: Y[-1], will return the last element of our signal,
But the influence is minor, we can ignore it for now.
In KS, they assume Y[-1] = 0, but we can leave it for now.

In other situation, Y[i-p-1] will return the last sample of the previous group, graphical illustration will look like this:



There we have the very basic KS algorithm.
There is the test sound in the next cell, as we can see it is very long
It will goes onto the last element of our place holder.
We can control the length in the next section.

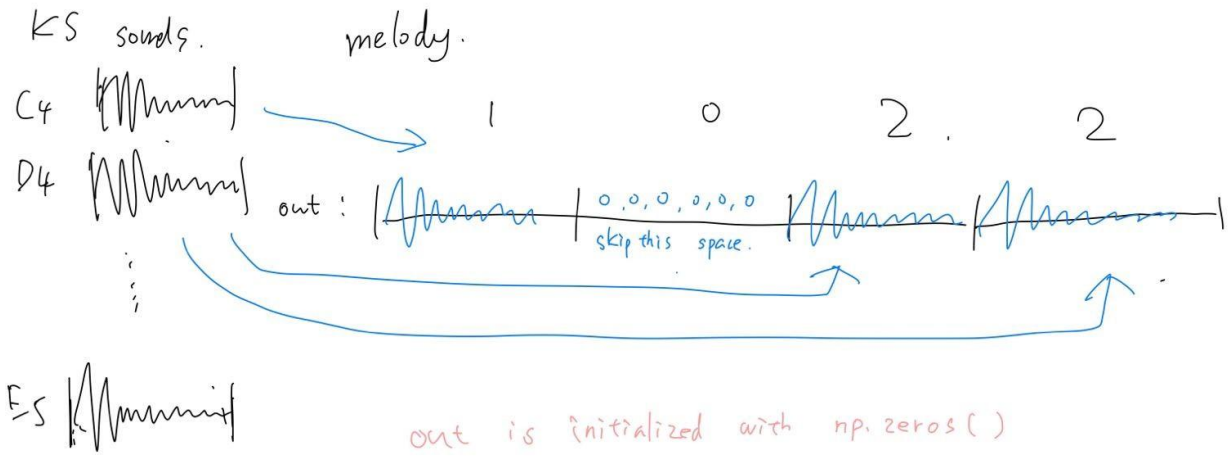
3. Next part is one of the extensions, we can add a decay factor outside of this averaging process, a similar process like in 1 part, the factor can range from 0 to 1
   Having a decay factor of 0.99 will control the signal to a good length.

   In the cell, there is different examples of different decay factor values:
   as the decay factor approaching to 0, the signal will decay quicker

4. making a group of KS guitar sound from C4 to E5

   note: the timbre of the sound is dependent on the random noise generated for each time but the timbre difference is so little, it is hardly notice.
   It made every KS note slightly different from each other, instead of 100% replication
   we will generate a random sample group for C4
   we will generate another random sample group for D4, different than that of C4
   ... etc
   there will be more ways to control the timbre in those extension algorithm

5. Then we will concate groups of KS sound as a scale or as a melody.
   We will generate an empty place holder, and we will stack the KS note to the correspond location
   It is like putting bricks on an empty playground



6. running through fast_conv() from previous lab
   convole with 2 IR: a fender Twin Amp
   a concert hall

IR credit:
[Room Impulse Response Data Set | Isophonics](#) for hall IR
[Fender Twin 73 IR Pack (shift-line.com)](#) for fender amp IR