MUS420 Lecture
Piano Synthesis

Julius O. Smith III (jos@ccrma.stanford.edu)
Center for Computer Research in Music and Acoustics (CCRMA)
Department of Music, Stanford University
Stanford, California 94305

June 27, 2020

Outline

- The Piano
- Commuted Piano Synthesis
- Hammer Modeling
- Piano String Modeling

The Piano

Acoustics of the Piano

- Five Lectures on the Acoustics of the Piano[1]
- The Piano Hammer as a Nonlinear Spring[2]

Piano Synthesis

We know how to

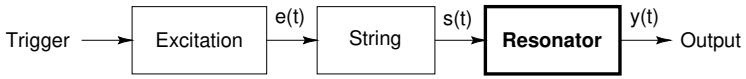
- simulate strings efficiently
- couple strings at the bridge

We will now discuss how to

- commute resonators and strings (for speed)
- convolve resonator impulse response with excitation (hammer)
- simulate a *piano hammer* in more detail
- identify the string-loop filter for stiff strings

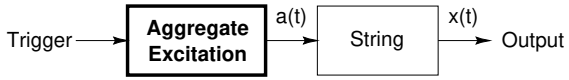
[1] http://www.speech.kth.se/music/5_lectures/
[2] http://www.acs.psu.edu/drussell/Piano/NonlinearHammer.html

Commuted Synthesis

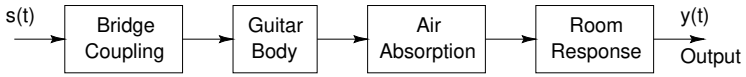


Schematic diagram of a stringed musical instrument.

Equivalent diagram in the linear, time-invariant case.



Use of an aggregate excitation given by the convolution of original excitation with the resonator impulse response.



Possible components of a guitar resonator.

Features of Commuted Synthesis

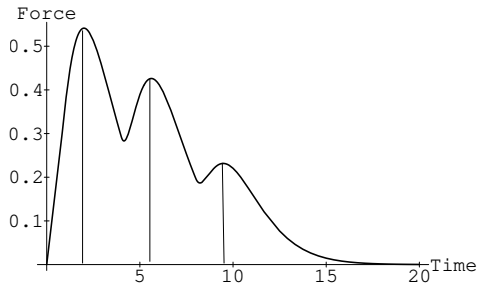
- Enormous resonators can be implemented inexpensively (three orders of magnitude less computation for typical stringed instruments)
- Good qualitative excitation signals are easy to measure (just tap on the bridge)
- Apparent "resonator size" can be modulated by changing the *playback rate* of the excitation table

Drawbacks:

- Requires *linearity* and *time invariance*

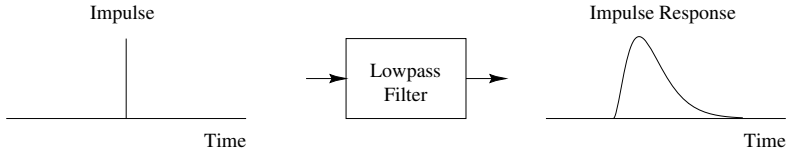
Commuted Piano Synthesis

Hammer-string interaction pulses (force):



- Vertical lines = locations and amplitudes of three *impulses*
- Hammer-string interaction signal synthesized using *one to three digital filters*
- Filters depend on *striking velocity*

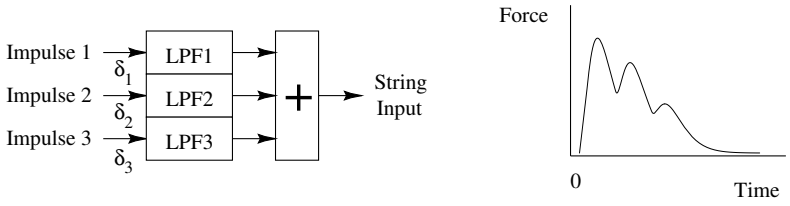
Synthesis of One Hammer-String Interaction Pulse



- Filter input = impulse
- Filter output = desired hammer-string force pulse
- As input amplitude increases, output pulse *narrows* $\Rightarrow$ *nonlinear filter*
- For each specific impulse, filter is linear time-invariant
- Piano is "linearized" separately for each hammer velocity

Synthesis of Multiple Hammer-String Interaction Pulse

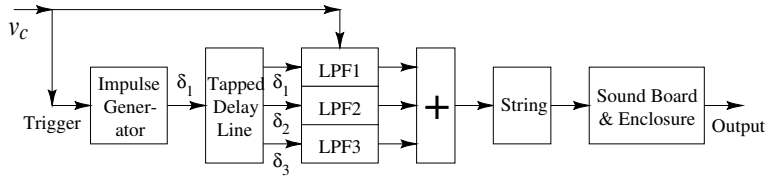
Multiple hammer-string interaction pulses = superposition of several individual pulses:



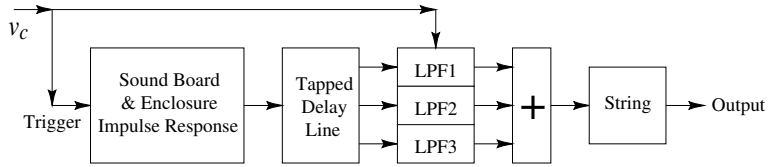
- Input to each filter is a single impulse
- Sum of outputs = superposition of hammer-string force pulses
- As impulse amplitude grows, output pulses become *taller and thinner*, showing less overlap.

Complete Piano Model

Natural Ordering:



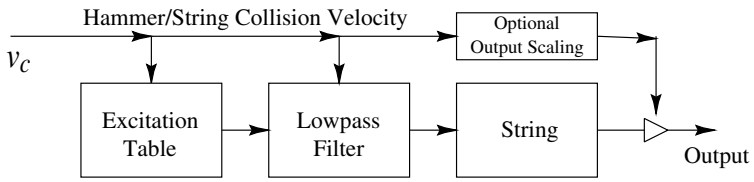
Commuted Ordering:



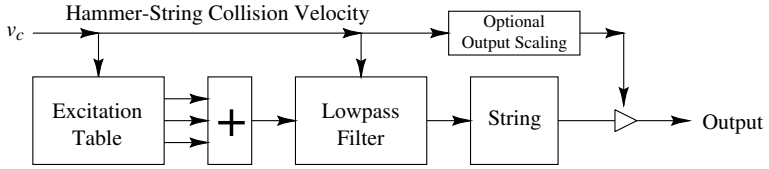
- Soundboard and enclosure are *commuted*
- Only need a stored recording of their *impulse response*
- An enormous digital filter is otherwise required

Simplified Variations of the Commuted Piano

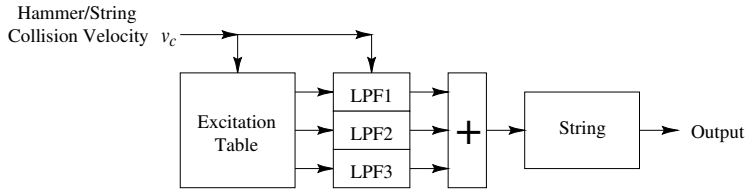
One interaction pulse:



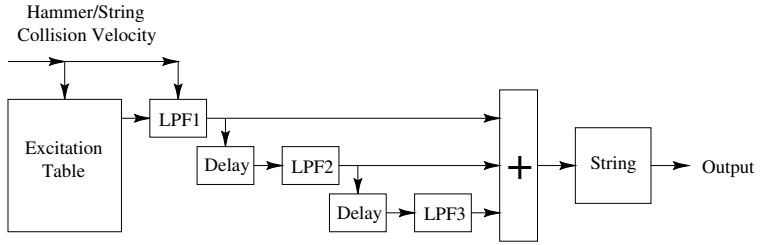
Up to three identical interaction pulses:



Three General Interaction Pulses



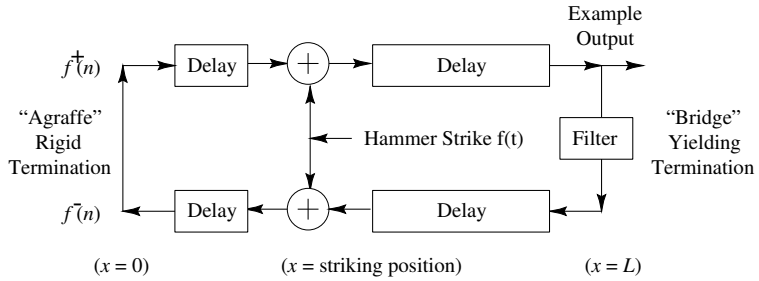
Differentially Filtered Interaction Pulses



Each new delay is equal to the travel from the hammer, to the agraffe, and back to the hammer

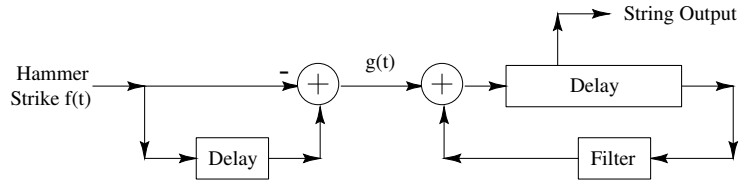
Striking a Digital Waveguide String

Waveguide string in "physical canonical form":



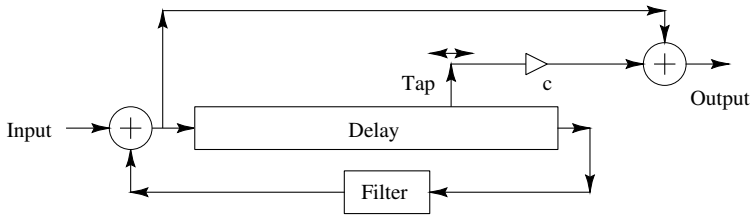
- Delay lines contain traveling *force waves*
- Hammer-string force pulses are summed into *both directions* (by physical symmetry)

Equivalent Computational Models



- Hammer position set by feedforward comb filter
- String model *independent of hammer position* (more valuable for plucked strings)
- Same factoring applies to *pickup locations*

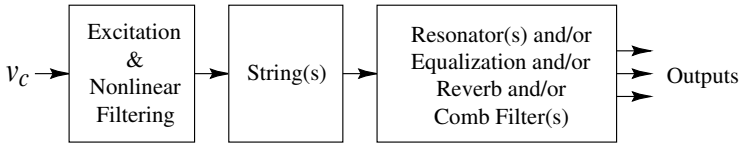
Use of a Tapped Delay Line



- Delay-line tap simulates hammer-strike echoes returning from the far end of the string
- Total delay line length equals twice the travel time along the entire length of the string
- Placement is not critical, so an interpolating tap is not necessary
- Tap motion gives a free built-in "flanger" effect
- Since force waves are most convenient, no sign inversion at the string endpoints

Excitation Factoring

Commuted Piano Synthesizer:



A few small filters are retained in parametric form for

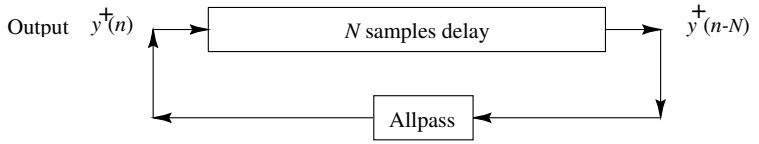
- Soundboard and enclosure
- Equalization for color variations
- Reverberation
- Flange
- Chorus
- Simulated hammer-strike echoes
- Multiple outputs

Filters are easily modified, even while a note is playing

Piano and Harpsichord Sound Examples

- `http://ccrma.stanford.edu/~jos/wav/pno-cs.wav`
- `http://ccrma.stanford.edu/~jos/wav/harpsi-cs.wav`
- `http://ccrma.stanford.edu/~jos/wav/Harpsichord.wav`
- Julien Bensa's synthetic piano

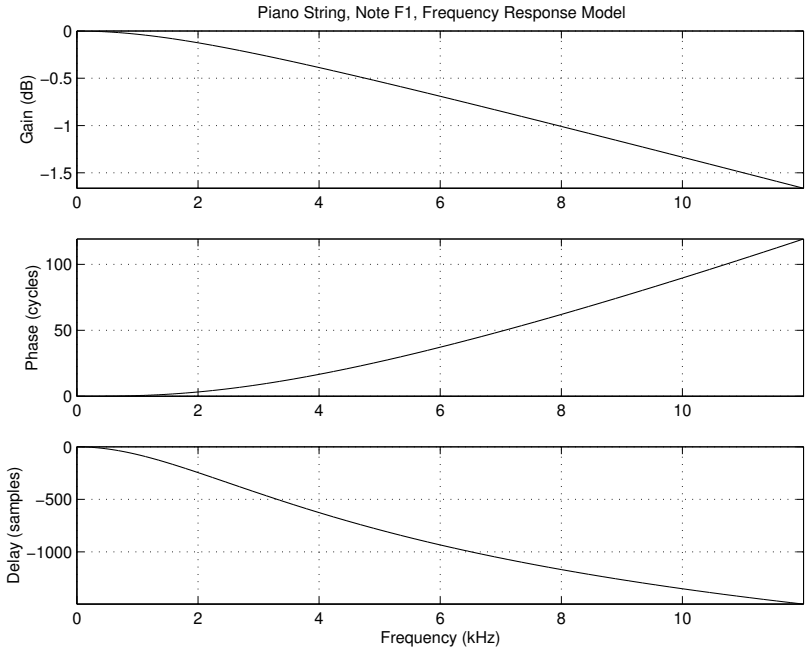
The Ideal (Lossless) Stiff String



Simulation of a rigidly terminated, stiff string.

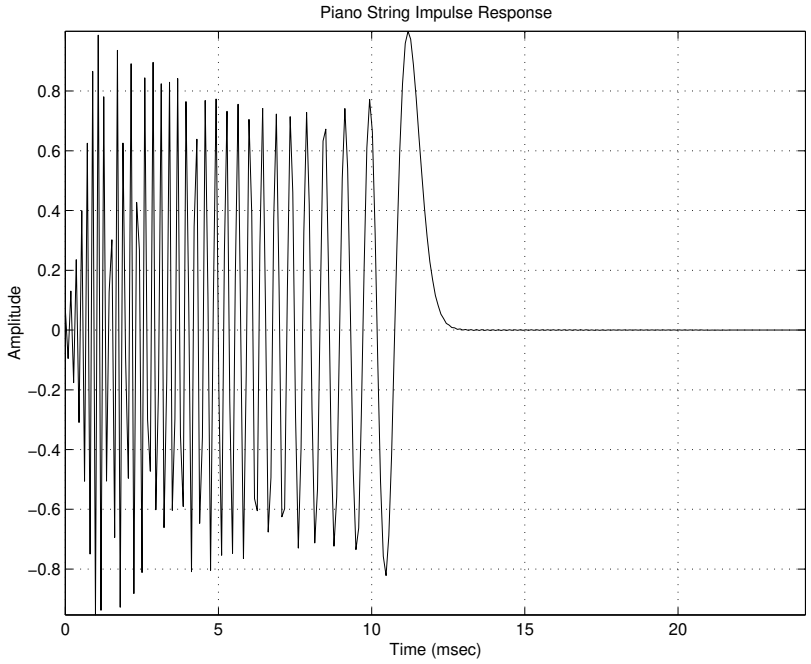
- All N allpass filters $H_a(z)$ have had a minimum sample of delay extracted, corresponding to the delay at $f_s/2$
- All have been combined at a *single* point
- The delay-line length N is the number of samples in K periods of f_K, where K is the number of the highest partial (usually the last one before $f_s/2$)
- In stiff strings, high frequencies travel faster $\Rightarrow$ allpass delay *decreases* with frequency
- An order 10 allpass gives high quality piano synthesis
- FFT convolution most efficient for maximum accuracy (FIR filter length $\approx N/2$ for note F1)

Nonparametric Model of Measured Piano-String Frequency Response (from Julien Bensa's thesis)



High-quality frequency-response model based on measurements from a grand-piano string, note F1.

Corresponding Impulse Response



Loop-filter impulse response, note F1.

Stiff-String Loop-Filter Identification

A basic problem in waveguide synthesis of string is identifying the *loop filter* $H_l(z)$ which controls

- Partial overtone decay time (due to $\left|H_l(e^{j\omega})\right|$)
- Partial overtone tuning perturbations (due to $\angle H_l(e^{j\omega})$)

Loop-Filter Design Methods

One of the earliest methods was "periodic linear prediction" — a linear combination of a small group of samples predicts a sample one period away from the midpoint of the group.

More recently, methods have been developed based on measurements of the *time-constant of decay* for each partial overtone, since this is what matters the most.

- Record a plucked (or struck) string.
- Measure frequency f_k and decay time τ_k for each overtone.
  - Energy Decay Relief (EDR) useful for this purpose

  - Sinusoidal modeling has been used as well
  - In either case:
    * Fit a straight line to the *log* of the amplitude envelope (e.g., using `polyfit` in Matlab or Octave).
    * Decay time τ_k is simply related to the *slope* of this line.
- Translate each decay-time τ_k into a desired loop-gain at each frequency f_k, thus determining $\left|H_l(e^{j\omega_k})\right|$ for each partial.
- If desired, use the inharmonic spacing of the f_k to compute samples of the desired *phase* of the loop filter, $\angle H_l(e^{j\omega_k})$.
- Use a general purpose digital filter design routine to design the loop filter which best approximates the desired frequency-response samples $H_l(e^{j\omega_k})$.
- Phase-matching filter design utilities:
  - Steiglitz-McBride algorithm (IIR or FIR)
    See `stmcb` in Matlab Signal Processing Tool Box
  - Equation-Error Minimization (IIR or FIR)
    See `invfreqz` in matlab
  - Hankel-Norm Minimization (IIR)

Also called a "balanced truncation model" (state space)

  - Linear-phase FIR (`fir1`, `firls`, `remez`)

- Phase-insenstive filter design:
  - Linear Prediction
    See `lpc` or `yulewalk` in the Matlab SPTB
  - Create a desired linear or minimum phase and use a phase-sensitive design method.
- The fit at low frequencies should be *weighted* higher.
- A simple $1/f$ weighting usually works well.
- Bark spectral warping is commonly used. It implicitly weights low frequencies more by "stretching out" the low-freqency frequency axis and compressing the high-frequency frequency axis.
- Piano strings are highly *dispersive*:
  - Recall that every LTI filter can be factored into a linear phase filter in series with an allpass filter
  - The allpass part for a low-pitched string requires on the order of 100 poles (and 100 zeros) to provide a closely matched impulse response
  - Ten poles is a typical number used in practice (can be very good perceptually)

  - Most methods for allpass filter design fail numerically at this high order
  - The linear-phase part, in contrast, is well approximated using only an 8-tap FIR filter, and can be designed by any good method for FIR filter design (*e.g.*, `invfreqz` in Matlab/Octave)

Order Estimation

Given a desired frequency response, a rough digital filter *order estimate* can be obtained as follows:

- FIR: Look at the IFFT duration (*e.g.*, $20\log(|h(n)|)$)
- IIR: Start with *half* of the FIR order (equal degrees of freedom)
- In either case, convert desired frequency response to *minimum phase* whenever possible: Matlab code:
  `http://ccrma.stanford.edu/~jos/filters/Matlab_listing_mps_m.html`
- For piano strings, a low-order minimum-phase damping filter is fine because we're going to need a high-order allpass dispersion filter anyway