Using a Waveguide to Model the Pipa in Csound

Ningxin Zhang[⋆]

Berkleee College of Music
nzhang4@berklee.edu

Abstract. Csound offers a huge set of tools for composers and sound designers. The author is a sound designer, electroacoustic composer, and classically-trained *pipa* player who was especially inspired by the sound of a number of Csound's waveguide opcodes, and many of the physically modeled Csound instruments. Immediately, the author used them in an attempt to imitate the sound of the pipa. However, the result was quite different from the acoustic pipa tone and thus led to the research presented in this paper. What is shown here is how, in order to simulate a more convincing pipa, the author needed to follow physical modeling practice, and hand code the filters mathematically.

Keywords: Csound, Pipa, Waveguide, Karplus-Strong

1 Introducing the Pipa

The pipa is an ancient Chinese four-stringed plucked instrument made of pear-shaped wood.[1] The standard height of the pipa is 102cm, while the effective string length is 72.5cm. Its neck is made up of 6 deep triangle-shaped frets called "xiang", with the tuning peg head extends upwards. On the soundboard are 24 bamboo frets called "pin". All 30 frets are spaced according to well-tempered tuning. The open strings for modern pipa are typically A2-D3-E3-A3 (110.5 Hz, 147.33Hz, 165.75Hz, 221Hz). Except for the A3 steel core string, the other three are nylon strings. The diameters of the four strings are 120.67µm (A2), 101.0µm (D3), 94.67µm (E3) and 79.5µm (A3) respectively.

2 Modeling the Pipa

2.1 Description of the Model

In the initial attempt to model the pipa in Csound, the author discovered several waveguide-based physical modeling examples of plucked instruments using the *pluck*, *repluck* and *wgpluck* opcodes, and started by modifying their arguments to obtain a sound closer to that of the pipa. However, although arguments

[⋆] I'd like to thank my professor Dr. Richard Boulanger for guiding me on learning Csound and helping me patiently with the synthesis process in this paper.

[1] To play the pipa, the player holds the pipa vertically, presses on the frets with left hand and plucks the strings with fake fingernails attached to each finger of the right hand. A link of famous pipa master Liu Dehai performing pipa repertoire *Ambush from Ten Sides* can be found in [8].

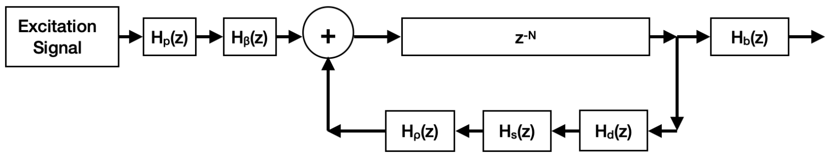

Fig. 1. Block diagram of the pipa model.

such as the excitation noise, and pluck position were changed, the resulting sound was still somewhat different from the sound of the acoustic instrument. This led to the following search to understand the underlying algorithms behind some of these opcodes.

The pipa model featured in this paper is shown in Fig. 1. This model is based on Yi-Huei Chen's physically modeled pipa, lecture notes by Victor Lazzarini[2], and the "Extended Karplus-Strong" algorithm.[3] To create a more realistic pipa sound, and to avoid problems such as: (1) depression on the non-harmonic elements, (2) extra decays on higher frequencies, (3) lack of a natural inharmonicity, the author added damping, stiffness, and tuning filters to the delay line to create a more realistic pipa sound. Further, to address the lack of dynamic variation in the output, instead of using random noise as an excitation, recordings of the pipa were used. This resulted in a richer sound closer to the actual pluck of the string. Then, a low-pass filter was used to adjust the amplitude of the excitation, and a comb filter was used to provide control over the pluck point. Finally, at the end of the synthesis process, a body model was added to finish the sound with the actual resonant body of the pipa.

2.2 Delay Line

The delay time, calculated in both samples and seconds, is used to set the loop length of the delay line, and to calculate the coefficient for the string tuning filter discussed in the section 2.4:

```
idlts = int(sr/ifreq-.5) ; delay time in samps
idlt = idlts/sr ; delay time in sec
kcount init idlts
if kcount < 0 goto continue
......
kcount -= 1
```

Opcodes *delayr* and *delayw* are used as a pair to read and write signal to the delay line:

[2] Professor Lazzarini's inspiring and enlightening lecture notes were shared, with his permission, by the author's thesis advisor.
[3] The Karplus-Strong algorithm is a simulation of an ideal string using the simplest frequency-dependent loss filter.

```
adel delayr idlt
......
    delayw awgout
```

2.3 String Damping Filter

The opcode *filter2* is used to implement all three filter models. The damping filter $H_d(z)$ is a first order low-pass filter designed for frequency dependent decay. As Lazzarini explained in his notes[3], the decay effect should be shortened or lengthened according to different fundamental frequencies. To do this, we divide the larger of the decay rate and amplitude response of the gain by the smaller of the two, and the quotient is the gain compensation to be multiplied by the filter coefficients. In fact, the coefficients used in this paper are following Chen's calculation results:

```
iwgdec = 7 ; attenuation in dB
ig = cos($M_PI*ifreq/sr)
igf pow 10,-iwgdec/(20*ifreq)

if (ifreq > 440) then ; coefficients at high/low frequency ranges
    igc = ig/igf
    adamp filter2 asig, 2, 1, 0.9015*igc, 0.1275*igc, 0.0331
elseif (ifreq <= 440) then
    igc = igf/ig

adamp filter2 asig, 1, 1, 0.6502*igc, -0.3412
endif
```

2.4 String Tuning Filter

An all-pass IIR filter $H_\rho(z)$ is used to fine-tune the instrument. According to Lazzarini's explanation of the all-pass tuning filter, because the sum of the integral delay size *int(sr/fun)* and the 1/2 sample[4] delay added by the low-pass filter is bigger than the desired decimal delay size *sr/fun*, this deviation should be subtracted from the desired fractional delay value D. The filter coefficient C is thus calculated according to the equation provided in [3]:

```
idltf = sr/ifreq-(idlts+.5) ; fractional delay D
icoe1 = (1-idltf)/(1+idltf) ; filter coefficient C

atuning filter2 asig, 2, 1, icoe1, 1, icoe1
```

[4] A more accurate phase response of the filter will be calculated to improve the tuning filter model.

2.5 String Stiffness Dispersion Filter

The transfer function $H_s(z)$ is designed to be a first order all-pass filter for high frequency range and a cascade of four first order all-pass filters for low frequency range. This is important for creating the inharmonic character for pipa that cannot be ignored, especially for the steel string. The inharmonicity is larger for lower strings than for higher strings. and a larger diameter of the string will also cause more obvious inharmonic effect. Besides, the inharmonicity has negative correlation with the length of the string. The equation of calculating the inharmonicity B can be found in [1] and also in the codes shown below. Next, by checking the information on [13] and [14], we get *Young's modulus*[5] for steel (29 Mpsi), and for nylon (0.425 Mpsi). Considering the difficulty of changing strings on this synthetic pipa, the instrument is divided into two parts: the notes below A3 is the nylon string part, while the notes higher than A3 will be played on the steel string. After that, we calculate the string tension $T_{(s)}$ by referring to the equation in [9]:

```
; inharmonicity coefficient equation
; B = pi^3*Q(Young's modulus)*d(string diameter)^4/
; 64*l(string length)^2*T(s)(string tension)
if (ifreq > 220) then ; A3
    iQ = 29 ; Young's modulus
    id = 0.00313 ; diameter
    igauge = 0.00001085
elseif (ifreq < 440) && (ifreq > 220) then ; E3, D3
    iQ = 0.425
    id = 0.00373
    igauge = 0.0002826
elseif (ifreq <= 220) then ; A2
    iQ = 0.425
    id = 0.00475
    igauge = 0.0002826
endif
    il = 28.5 ; length of string(inch)
    ig = 386.08858
iT = ((ifreq*2*28.5)^2*igauge)/ig ; T(s) = (f*2*L)²*µ/g
iB = ($M_PI^3*iQ*id^4)/(64*il^2*iT) ; coefficient B
```

The optimal design parameters and equations for calculating coefficients for the dispersion filters refer to table 1 and equations 4-7 in [2]. The coefficients k_d and C_d are calculated using logarithmic scale, with a second-order polynomial and a straight line as follows:

```
ik1 = -0.0026580 ; polynomial coefficients
ik2 = -0.014811
```

[5] Young's modulus, (the modulus of elasticity in tension or compression), shows how easily the material can stretch or deform when lengthwise force is applied.

```
ik3 = -2.9018
iC1 = 0.071089 ; straight line coefficients
iC2 = 2.1074
......

ikey = log(ifreq)/log(1.059)-56.813
ikdB = $M_E^(ik1*(log10(iB)^2)+ik2*(log10(iB))+ik3)
icdB = $M_E^(iC1*(log10(iB))+iC2)
iD   = $M_E^(icdB-ikey*ikdB)
icoe2 = (1-iD)/(1+iD)

astiff filter2 asig, 2, 1, icoe2, 1, icoe2
......
```

2.6 Excitation Signal

Many waveguide synthesis examples used simple random noise as an excitation signal, yet these do not accurately represent the sonic character of the pipa. Thus, to further model the timbre of the pipa, the excitation signal of this model is a combination of recordings of body impulse response (IR) and single plucking tone on the A3 string. The two recordings use the *ftgen* opcode and the *GEN01* routing to transfer the sound files into f-tables :

```
giPipa ftgen 0, 0, 262144, 1, "pipa-G4.wav", 0, 0, 0
giIr   ftgen 1, 0, 16384, 1, "pipa-Body-IR.wav", 0, 0, 0
```

The pick position low-pass filter $H_p(z)$, and comb filter $H_\beta(z)$, are implemented using *butterlp* and *delay* opcodes[6]:

```
knoise1 oscil1i 0, 1, ftlen(giPipa)/sr, giPipa
anoise1 upsamp knoise1
knoise2 oscil1i 0, 1, ftlen(giIr)/sr, giIr
anoise2 upsamp knoise2

anoise1 butterlp anoise1, iamp*iamp*sr/2 ; lowpass filter
anoise2 butterlp anoise2, iamp*iamp*sr/2
acomb1 delay anoise1, ipluck*ifreq ; comb filter
acomb2 delay anoise2, ipluck/ifreq
anoize1 = anoise1-acomb1
anoize2 = anoise2-acomb2
```

2.7 Body Model

The extra spectrum-based body model $H_b(z)$ (suggested in [1]) is added to our model by using four cascading biquad filters:

[6] Oscillators will replace the oscil1i opcode later to avoid up-sampling the signals.

```
; low-shelf, peak-notch, peak-notch, high-shelf
afilt1 biquad asig, 0.2485, -0.4937, 0.2453, 1, -1.9747, 0.9755
afilt2 biquad afilt1, 0.5888, -0.4687, -0.1190, 1, -1.8746, 0.8794
afilt3 biquad afilt2, 0.7701, -0.1817, -0.4351, 1, -0.7269, 0.3400
afilt4 biquad afilt3, 0.1753, -0.1708, 0.0743, 1, -1.1763, 0.4913
```

3 Synthetic Results

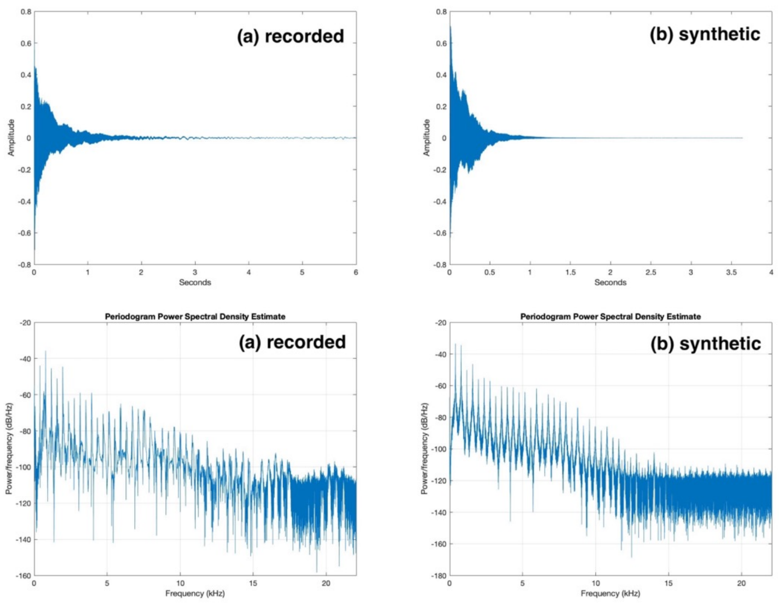

Fig. 2. Synthetic reseults.

The recorded and synthetic audio signals in the time and spectral domain are visualized in Fig. 2. The decay time of the synthetic result is basically consistent with that of the pipa recording. To further fine-tune the design, the plan is to make better recordings of the excitation and address some noise issues at high frequency range. Aside from this, the resulting tone is much closer to the pipa than the author was ever able to get using the existing opcodes.

4 Next Steps

The goals now are to further improve the overall sound, and begin to address the wide range of sound variations associated with all the playing techniques. Future plans include porting the model to the *Raspberry Pi* and creating a physically modeled physical instrument with a *Bela* trill flex sensor.

References

1. Chen, Y. and Huang, C., 2011. Sound Synthesis of the Pipa Based on Computed Timbre Analysis and Physical Modeling. *IEEE Journal of Selected Topics in Signal Processing*, 5(6), pp.1170-1179.
2. Rauhala, J. and Valimaki, V., 2006. Tunable dispersion filter design for piano synthesis. *IEEE Signal Processing Letters*, 13(5), pp.253-256.
3. Lazzarini, V., n.d. Physical Modeling in Csound.
4. Lazzarini, V. et al.: *Csound: A Sound and Music Computing System.* Springer (2016)
5. Lazzarini, V., 2021. *Spectral Music Design.* Oxford University Press.
6. Boulanger, R. and Lazzarini, V., 2011. *The Audio Programming Book.* Cambridge, Mass.: MIT Press.
7. Boulanger, R., 2000. *The Csound book.* Cambridge, Mass.: MIT Press.
8. Youtube.com. 2022. *Liu Dehai Performing Ambush from Ten Sides.* `https://www.youtube.com/watch?v=1Rx5uqX4mBA`
9. Omnicalculator.com. 2022. *Guitar String Tension Calculator.* `https://www.omnicalculator.com/other/guitar-string-tension#guitar-string-tension-formula-the-physics-behind-guitar-string-tension`
10. Smith, J. O., 2022. *Physical Audio Signal Processing for Virtual Musical Instruments and Audio Effects.* `https://ccrma.stanford.edu/~jos/pasp/pasp.html`
11. Smith, J. O., 2022. *MUS420 Lecture Elementary Digital Waveguide Models for Vibrating Strings.* `https://ccrma.stanford.edu/~jos/SimpleStrings/SimpleStrings.html`
12. University of Birmingham. 2022. *Physics - Young's modulus.* `https://www.birmingham.ac.uk/teachers/study-resources/stem/Physics/youngs-modulus.aspx#:~:text=The%20Young's%20modulus%20(E)%20is,%CE%B5%20%3D%20dl%2Fl`
13. Polytechindustrial.com. 2022. *Nylon® 6/6* (Polyamide) | Poly-Tech Industrial. `https://www.polytechindustrial.com/products/plastic-stock-shapes/nylon-66`
14. AZoM.com. 2022. *ASTM A36 Mild/Low Carbon Steel.* `https://www.azom.com/article.aspx?ArticleID=6117`