Nonlinear Commuted Synthesis of Bowed Strings

Julius O. Smith III
CCRMA, Music Department, Stanford University
jos@ccrma.stanford.edu, http://www-ccrma.stanford.edu/~jos

August 11, 2002

Abstract

A commuted-synthesis model for bowed strings is driven by a separate nonlinear model of bowed-string dynamics. This gives the desirable combination of a full range of complex bow-string interaction behavior together with an efficiently implemented body resonator. A "single-hair bow" may control a pulsed-noise version which provide the effects of multiple bow hairs. The pulsed noise may also include qualitatively the impulse responses of commuted high-frequency body modes.

Contents

---

[0] Expanded version for CCRMA affiliates. A shorter four-page version was submitted to the 1997 International Computer Music Conference.

1 Introduction

According to prevalent theories of bow-string interaction [1, 2], disturbances sent out by the stick-slip process along the string are fundamentally *impulsive* in nature. That is, the bow is normally either sticking or slipping against the string, and the main excitation events on the string occur when the slipping starts or ends, at which point there is a narrow acceleration pulse sent out in both directions along the string. (There is also sliding noise during slipping each period, but that can be dealt with separately.) Both the Helmholtz [1863] and Raman [1918] models of bowed string behavior consist only of sparse acceleration impulses on the string. Raman's theory, in fact, classifies the various motions according to how many impulses there are per period. Basic Helmholtz motion only consists of one impulse per period, while other modes, such as "surface sounds" generated by "multiple slips," or "multiple flybacks," consist of two or more acceleration impulses per period.

The implication of any "sparse impulse model" of bowed-string interaction is that it can be used to efficiently drive a commuted synthesis implementation for bowed strings [3, 4]. The advantage of commuted synthesis is that a potentially enormous recursive digital filter representing the resonating body is avoided. When an impulse reaches the bridge, a body impulse response (BIR) is "triggered" at the amplitude of the impulse. The commuted synthesis implementation thus "watches" impulses arriving at the bridge in the bowed-string model, and instantiates a BIR playback into a separate string model on the arrival of each impulse. (BIR playbacks which overlap in time are summed.) A BIR playback may be implemented, for example, using a wavetable oscillator in "one-shot" mode. The variable playback rate normally available in such an oscillator can be used to modulate apparent "body size" [5, Mandolin.cpp]. The impulse-triggered BIR playback scheme can be classified as an efficient "sparse-input FIR filter" implementation of the body resonator. For simple Helmholtz motion, this model reduces to the original bowed-string commuted-synthesis model, except that we may now generate automatically impulse amplitude and timing information from the bow-string interaction model, and we can use physical bow force, position, and velocity signals as the control inputs. In this way, we obtain the reduced computational cost of commuted synthesis, at least during smooth playing, while allowing for fully general interaction between the bow and string.

2 Nonlinear Commuted Model

The basic idea of commuted synthesis is to interchange the order of implementation of the string and the body resonator, as depicted in Fig. 1.

The bowed string synthesizer of the present paper is shown in Fig. 2. The bottom half is Fig. 1c, with an external trigger input, and some further details regarding pulsed noise generation. The top half of Fig. 2 provides an explicit model of bow-string dynamics. The "Impulse Prioritizer" measures the timing and amplitude of the largest impulses in the string waveform at the bridge and passes on the most important ones subject to complexity constraints. The second string which is driven by the BIR oscillators may be a digital waveguide model driven at the bowing point, or it may consist of an equivalent feedforward comb filter followed by a filtered delay loop. However, the advantage of a a full waveguide

---

"Nonlinear Commuted Synthesis of Bowed Strings," J.O. Smith, based on ICMC-97 paper of the same name.

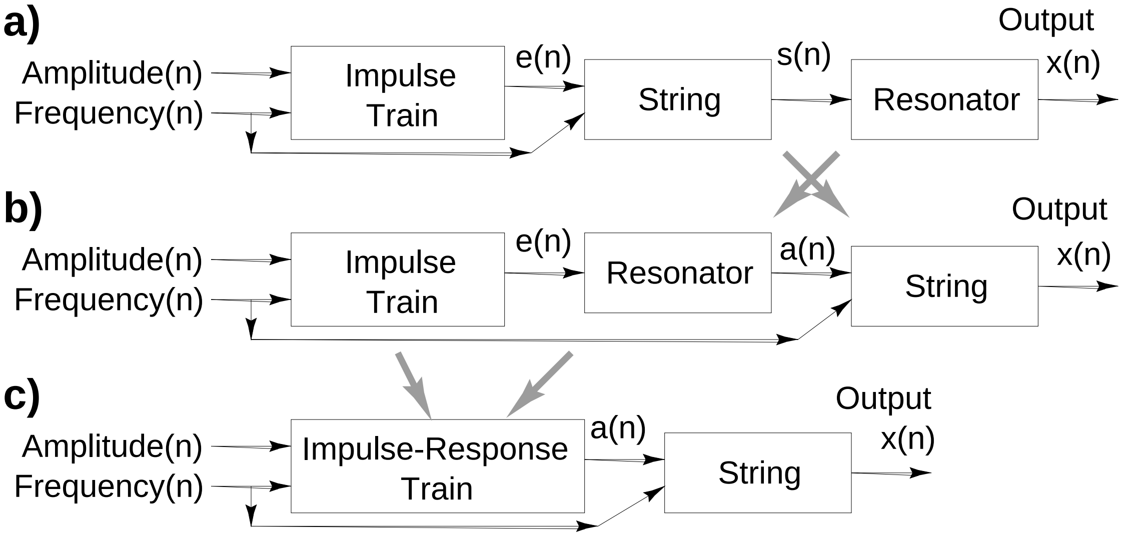

Figure 1: a) Simplified bowed string model, including only amplitude, pitch, and vibrato control capability. b) Equivalent diagram with resonator and string commuted. c) Equivalent diagram in which the resonator impulse response is played into the string each pitch period.

model of the string [6] is that the time-varying, nonlinear, partial termination of the string by the bow can be more conveniently implemented.

The Stick/Slip Bit can be used to switch between two models of partial string termination by the bow. For more accurate control of string damping by the bow, the contact force, relative velocity, position along the string, and bow angle can all be used to determine the frequency-dependent scattering junction created by the bow on the string [6]. It was found empirically that significant damping of the string by the bow is necessary for obtaining robust Helmholtz motion; otherwise, excessive ringing of the string segment between the bow and nut tends to cause slipping at times disruptive to the Helmholtz motion. Intuitively, one of the two "Helmholtz corners" sent in opposite directions along the string on each slip/stick impulse must be "filtered out" by the bow, while the other is "amplified" by the stick/slip process. Graphical animation of the bowed string motion was found to be very helpful for determining qualitative factors such as this.

3 Nonlinear Bow Friction

For this study, a simplified bow-string interaction model was implemented having the following characteristics:

- Static frictional "release force" is a multiple of the vertical bow force.
- Dynamic frictional force is small and fixed (independent of bow force).

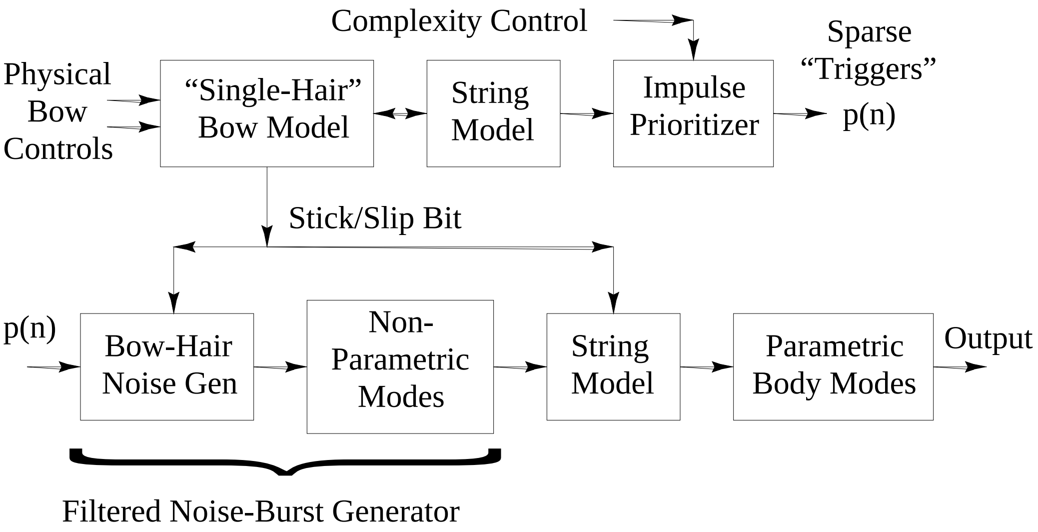

Figure 2: Commuted bowed string synthesis model driven by a separate bow-string model exhibiting full nonlinear dynamic behavior.

- The "capture force" is somewhat smaller than the "release force" but also a multiple of the vertical bow force.
- One bit of state is maintained ("sticking" vs. "slipping") in order to distinguish whether to use capture or release maximum forces.

The use of different maximum forces for release versus capture is motivated by the recent findings that there is evidence that the bow rosin *melts* during slipping and refreezes during sticking [7].

4 Friction Impulse Detection

The output of the bow-string simulation must be converted to discrete trigger events, with each trigger initiating playback of the body impulse response (BIR). Ideally, we would like a means of "thinning" the impulses coming from the bridge so as to keep the most important ones and neglect the least important ones to the degree necessary to meet computational resource restrictions.

There are several alternative impulse thinning schemes. Perhaps the simplest is to set an impulse amplitude *threshold*, such as ten percent of the expected main impulse amplitude, such that any impulse over the threshold in magnitude is passed on as a trigger, and anything smaller is suppressed. When the threshold is crossed by the absolute value of the bridge acceleration waveform in an upward direction, the next local maximum is taken to determine the impulse amplitude and timing. No further impulses are accepted until the bridge acceleration falls below the threshold. As a further refinement, the samples on either side of the local maximum can be used to quadratically interpolate the peak, as is typically done for spectral peaks; alternatively, or in addition, the bow-string simulation can be run at a higher sampling rate than the commuted synthesis unit in order to further improve the impulse timing accuracy.

The simple threshold method does not introduce latency, which is important in the real-time case, but it does not enable optimal impulse detection methods and there is no direct control over complexity (it is not easily known in advance what threshold will thin the impulse stream to the necessary extent). An indirect control over complexity is obtained by setting the threshold dynamically as a function of the number of overlapping BIRs. In this way, the threshold can be lifted to increase the thinning when the complexity becomes too great. An advantage of this thinning algorithm is that it doesn't matter what the source of complexity is. For example, impulses may be thinned because the pitch went higher causing more BIR overlap, or because other voices came in reducing the available number of BIR oscillators, or because the end user changed a preference specifying an upper limit on computing resources to be devoted to sound synthesis on a general purpose computer.

A more direct impulse thinning scheme which introduces one period of latency delay is as follows: The most recent period of the bridge signal is kept in a circular buffer at all times. Let N_e denote the maximum number of stick-slip events allowed per period P. To restrict behavior to basic Helmholtz motion, N_e can be set to 1. To allow second-order Raman motion, $N_e = 2$ would be appropriate, and so forth. At each time step, the largest N_e peaks in the last period are defined as the impulses to send out. Since there is one period of latency, it is always the case that the emitted impulses are the most important ones within the past period. Having a period of "look ahead" enables use of more sophisticated peak detection schemes than the simple local-maximum-after-threshold-crossing method.

A variation on the threshold method which does not need threshold adaption for complexity control is analogous to *voice allocation* in polyphonic synthesizers: When an impulse crosses a nominal threshold level, the next local maximum triggers a BIR playback unless (1) all playback units are busy *and* (2) the desired playback amplitude is *smaller* than that of *all* of the playing BIRs. When all BIR units are busy but one of them is deemed less important than the desired new BIR, the least important BIR is *preempted*, interrupting its playback and restarting it at the desired amplitude for the new BIR playback.

5 Pulsed Noise

A stick-slip event never involves only one bow hair, and during the slipping interval, or string "flyback," there is a soft noise burst which is audible, especially at close range. It is well known that pulsed noise is an important feature of high quality bowed-string synthesis as well as other instruments [8]. The Stick/Slip Bit provided by the bow-string contact model (see Fig. 2) indicates when sliding noise is appropriate. As in the case of the time-varying string-damping discussed above, more refined noise-generation models can be devised based on the bow force, differential velocity, and position information available from the bow-string simulator, as well as an external "bow angle" control.

When the resonating body transfer function is *factored* [9] into slowly decaying modes (implemented parametrically using recursive filters and not necessarily commuted) and rapidly decaying modes (which are commuted and used in nonparametric form as impulse response

---

data), the commuted nonparametric impulse response is qualitatively a short, *high-frequency noise burst*, since it consists of the impulse responses of thousands of high-frequency, highly damped modes. In principle, this "damped-modes-noise-burst" should be convolved with the noise arising from the slipping bow. In other words, the string excitation for each stick-slip event can be modeled as a filtered noise burst which includes both the highly damped resonator modes and the bow noise.

6 Simulation Results

Figure 3 displays waveforms generated by the bow-string model given a constant bow force, velocity, and position. The frictional force applied to the string by the bow can be seen to diminish as the oscillation develops. The string displacement near the bridge clearly exhibits the single main impulse once per period associated with canonical Helmholtz bowed-string motion; there are also many secondary impulses associated with the ringing of the piece of the string between the bridge and the bow. The complexity control will determine whether these secondary impulses are included or suppressed.

Figure 4 illustrates the samples of bridge displacement waveform over a longer period of time. Note that each main Helmholtz impulse plots as two adjacent samples, indicating that a single-sample impulse is traveling on the string. (The observation point is 1/2 spatial sample from the bridge, so that a single impulse at the bridge appears twice, both before and after reflection at the bridge.) Note also that late in the stroke, a strong secondary impulse has developed, making the sound tend toward an octave higher. This "sul ponticello" sound is associated with insufficient bow force.

Figure 5 gives a close-up of the frictional force during the initial attack transient. As can be seen, even though the applied bow force and velocity are constant, a highly complex interaction occurs between the bow and string.

Figure 6 shows an overlay of the first 40 periods of oscillation of the bowed string, with each string snapshot taken slightly later than one period after the previous, and the first snapshot being taken at time zero. The bow is at the sharp upper corner on the left. Note that the vertical scale is highly magnified relative to the horizontal scale. There is also some distortion in the string shape resulting from the lumping of the string losses at the bridge and bowing point, as is typical in waveguide string modeling.

7 Conclusions

The commuted bowed-string synthesis model was extended to incorporate driving information from a nonlinear model of bowed-string dynamics. The formulation allows a simplified "single-hair bow" to control a pulsed-noise driven commuted synthesis model, thereby simulating a full-width bow in the final sound quality. Commuting only the fastest decaying (high frequency) body modes results in a short, damped impulse response which can be regarded as a component of the pulsed noise. In summary, driving a commuted-synthesis model for bowed strings from a nonlinear model of bowed-string dynamics gives the desirable combination of a full range of complex bow-string interaction behavior together with a

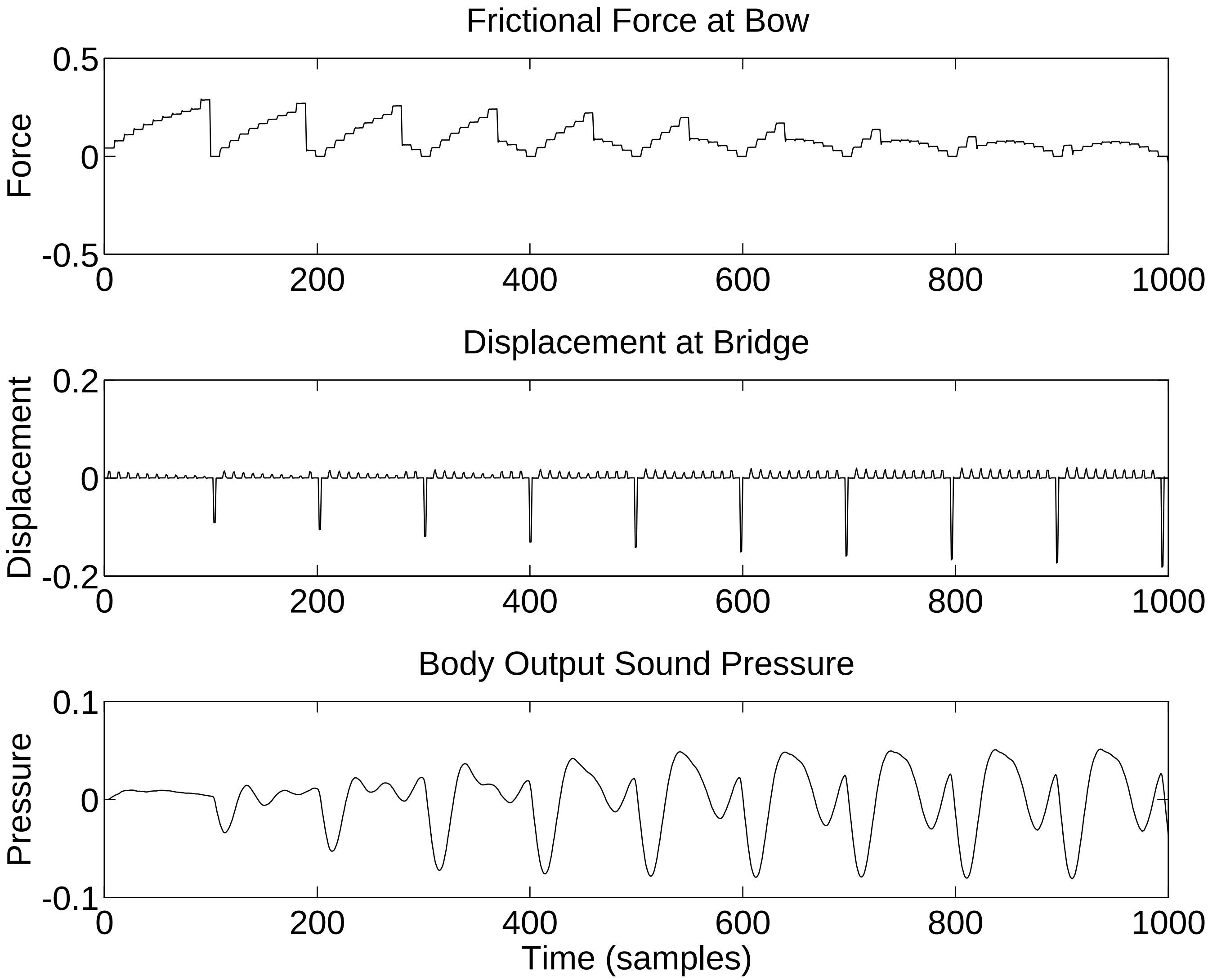

Figure 3: Output of the bow-string model before extracting bridge impulses. Top: Frictional force between bow and string. Middle: String displacement 1/2 sample from the bridge. Bottom: Sound pressure radiated from simulated body filter. Bowing parameters (fixed): speed 15 cm/sec, force 20 grams, position 3 cm from bridge. A two-pole, two-zero bridge-filter for a digital waveguide string model was calibrated to measurements of violin pizzicato waveforms. A torsional-wave loss coefficient of 0.9 was implemented at the bow at all times.

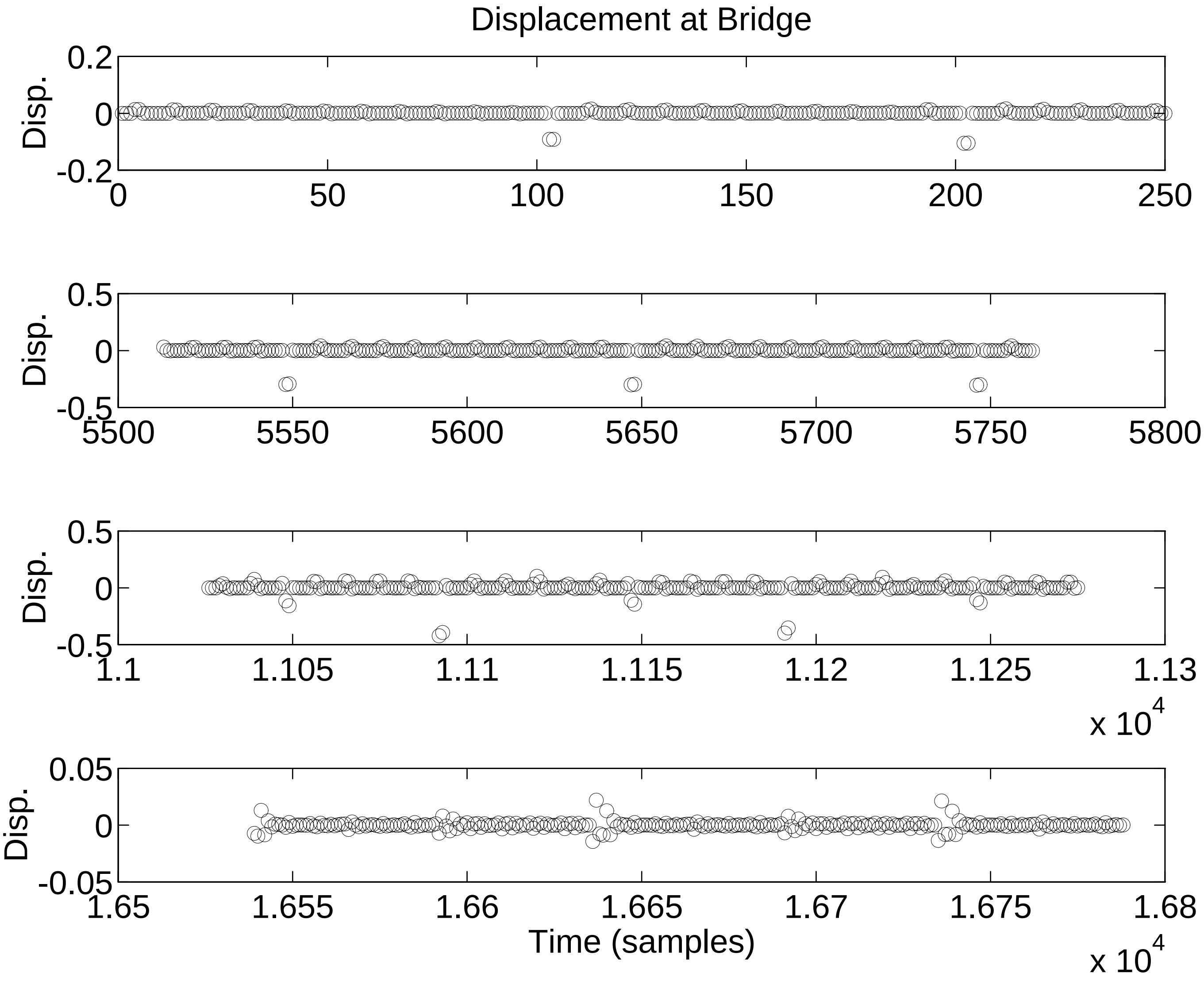

Figure 4: String displacement 1/2 sample from the bridge over four short time intervals spanning 1.7 seconds.

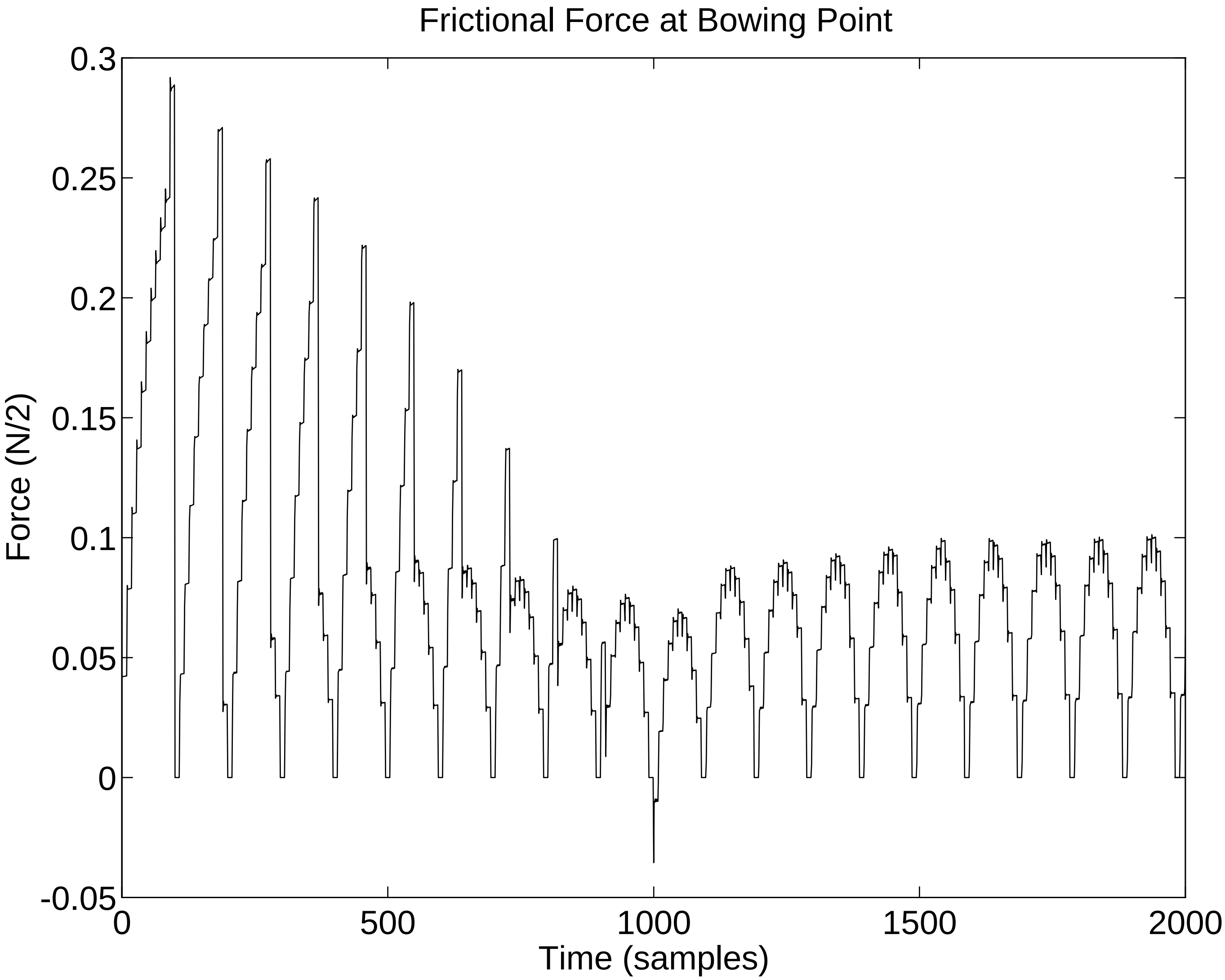

Figure 5: Close up of the frictional force waveform during the initial attack.

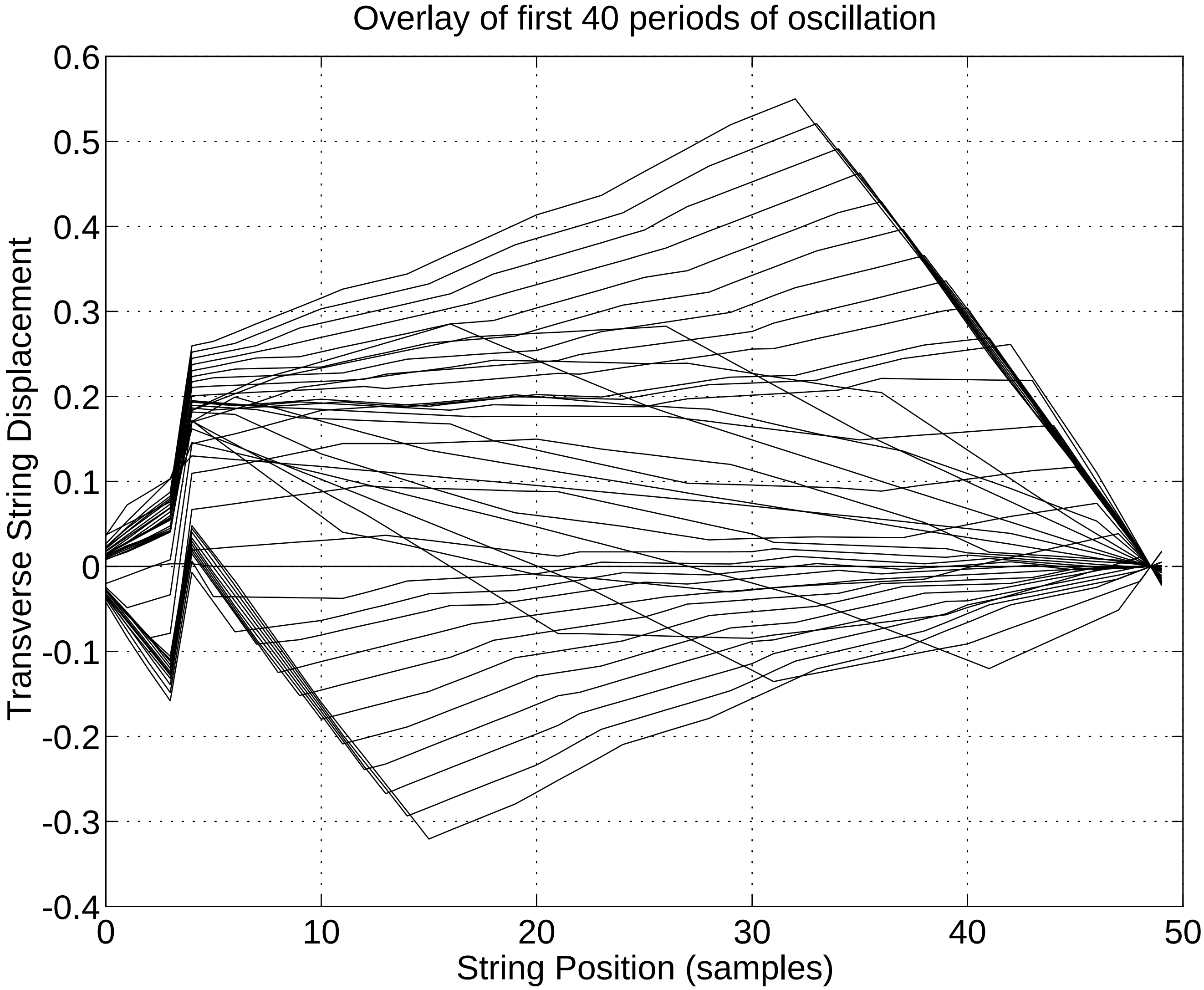

Figure 6: Snapshots of string state for first 40 periods of oscillation.

reduced-complexity body resonator.

8 Appendix: Selected Software Items

All simulations for this paper were carried out using Perry Cook's Synthesis ToolKit (STK) in C++ [5]. The method `stringVelocityAtPosition(int position)` below can be added to bowed.cpp in the STK to facilitate extracting the string state for display as shown in Fig. 6. (Animations of bowed-string motion using this added method were found to be especially valuable for obtaining insight into bowed string dynamics.)

```
MY_FLOAT BowedStr :: stringVelocityAtPosition(int p) /* p from 0 to nsamples-1 */
{
    int bdelrt = (int)bridgeDelay->delay()+1; /* bow-to-bridge-to-bow + p.l. delay */
    int ndelrt = (int)neckDelay->delay()+1; /* bow-to-nut-to-bow + p.l. delay */
    int bdel = bdelrt >> 1; /* number of spatial samples from bridge to bow */
    int ndx = bdel - p; /* convert (0:N-1) position to delay (1:N) on left */
    MY_FLOAT leftGoingAtP;
    MY_FLOAT rightGoingAtP;
    if (p < bdel) {
        /* "Now" is always where the InPoint points which is not yet written */
        /* OutPoint points to "now - delay" */
        /* Bridge is on the left, nut on the right */
        /* "Now" is at the bow */
        /* Position zero is at far left = half way along delay line */
        leftGoingAtP = bridgeDelay->contentsAtNowMinus(ndx);
        int rndx = bdelrt-ndx+1;
        if (rndx < bdelrt) { /* last sample delay resides in lastOutput variable */
            rightGoingAtP = -bridgeDelay->contentsAtNowMinus(rndx);
        } else {
            rightGoingAtP = -bridgeDelay->lastOut();
        }
    } else { /* nut side */
        ndx = p - bdel + 1; /* convert (0:N-1) position to delay (1:N) */
        rightGoingAtP = neckDelay->contentsAtNowMinus(ndx);
        int lndx = ndelrt-ndx+1;
        if (lndx < ndelrt) {
            leftGoingAtP = -neckDelay->contentsAtNowMinus(lndx);
        } else {
            leftGoingAtP = -neckDelay->lastOut();
        }
    }
    return rightGoingAtP + leftGoingAtP;
}
```

Usage of the above method is illustrated in the following code fragment:

```
MY_FLOAT stringState[MAXPERIOD];
for (i=0;i<period/2;i++)
    stringState[i] = 0;
for (i=0;i<samples;i++) { /* main sample loop */
    ...
    if (i<20*period) {
        MY_FLOAT v = 0.0;
        long len = period/2;
        for (int j=0; j<len; j++) {
            v = vscale*vln->stringVelocityAtPosition(j); /* m/s */
            stringState[j] += v*ONE_OVER_SRATE; /* m */
            stringOut->tick(STRINGSCALING*stringState[j]); /* to soundfile */
        }
    }
```

The following matlab function was used to generate highly helpful animations of the string state. Each string snapshot was written successively into one long sound file, and this routine was called with the sound data along with M set to the snapshot length in samples:

```
function out=datamovie(in,M,sleep,ax);
%DATAMOVIE   datamovie(in,M,sleep,ax);
%           Display sequence of data frames of length M.
%           If sleep>0, that many cycles are waited between plots.
%           If sleep == -1, RETURN is needed to advance to the next plot.
%           If ax is a quoted string, "axis(ax)" is called.
clf;
if (nargin<2), M=length(in); end
if (nargin<3), sleep=0; end
if (nargin<4), ax = [1 M min(in) 1.1*max(in)]; end
skip = M; h=plot(in(1:M),'erasemode','background'); axis(ax); drawnow;
if sleep == -1, disp '*** PAUSING *** RETURN to continue'; pause; end
for i=1:(length(in)-M)/skip
    set(h,'ydata',in(skip*i+1:skip*i+M));
    if sleep == -1, disp '*** PAUSING *** RETURN to continue'; pause;
    elseif sleep>0, for j=1:sleep, y = tan(j); end; end
end
```

Usage of the `datamovie` function is illustrated by the matlab script below:

```
% seestr.m - matlab script for viewing bowed string waveshape evolution
ilen = 50;              % Number of spatial samples along string
sleep = 0;              % pause/speed control to datamovie
name1 = 'string'; [strdata fs len header] = loadsig(name1); % for NeXT .snd files
strdata = strdata/32768.0;
datamovie(strdata,ilen,sleep);
```

References

[1] M. E. McIntyre and J. Woodhouse, “On the fundamentals of bowed string dynamics”, *Acustica*, vol. 43, no. 2, pp. 93–108, Sept. 1979.

[2] K. Guettler, “The bowed string computer simulated — some characteristic features of the attack”, *Catgut Acoustical Society Journal*, vol. 2, no. 2, pp. 22–26, Nov 1992, Series II.

[3] J. O. Smith, “Efficient synthesis of stringed musical instruments”, in *Proceedings of the 1993 International Computer Music Conference, Tokyo*. 1993, pp. 64–71, Computer Music Association, available online at http://www-ccrma.stanford.edu/~jos/cs/.

[4] D. A. Jaffe and J. O. Smith, “Performance expression in commuted waveguide synthesis of bowed strings”, in *Proceedings of the 1995 International Computer Music Conference, Banff*. 1995, pp. 343–346, Computer Music Association.

[5] P. R. Cook, "Synthesis toolkit in C++, version 1.0", in *SIGGRAPH Proceedings*. May 1996, Assoc. Comp. Mach., See http://www.cs.princeton.edu/~prc/NewWork.html for a copy of this paper. The Synthesis Tool Kit (STK) software itself is distributed by CCRMA: http://www-ccrma.stanford.edu/CCRMA/Software/STK/.

[6] J. O. Smith, "Efficient simulation of the reed-bore and bow-string mechanisms", in *Proceedings of the 1986 International Computer Music Conference, The Hague*. 1986, pp. 275–280, Computer Music Association, Also available in [**?**].

[7] J. H. Smith, *Stick-Slip Vibration and its Constitutive Laws*, PhD thesis, Cambridge University, 1990.

[8] C. Chafe, "Pulsed noise in self-sustained oscillations of musical instruments", in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, Albuquerque*, New York, 1990, IEEE Press, available as CCRMA Technical Report STAN-M-65, Music Dept., Stanford University.

[9] M. Karjalainen and J. O. Smith, "Body modeling techniques for string instrument synthesis", in *Proceedings of the 1996 International Computer Music Conference, Hong Kong*. Aug. 1996, pp. 232–239, Computer Music Association, available online at http://www-ccrma.stanford.edu/~jos/cs.html.

[10] H. L. F. von Helmholtz, *On the Sensations of Tone as a Physiological Basis for the Theory of Music*, Dover, New York, 1954, English translation of 1863 (German) edition by A. J. Ellis.

[11] C. V. Raman, "On the mechanical theory of vibrations of bowed strings, etc", *Indian Assoc. Cult. Sci. Bull.*, vol. 15, pp. 1–158, 1918.

[12] H. L. F. von Helmholtz, *Die Lehre von den Tonempfindungen als Physiologische Grundlage für die Theorie der Musik*, F. Vieweg und Sohn, Braunschweig, 1863.

[13] R. Pitteroff, "Modelling of the bowed string taking into account the width of the bow", in *Proceedings of the Stockholm Musical Acoustics Conference (SMAC-93)*, Stockholm, July 1993, pp. 407–410, Royal Swedish Academy of Music.

[14] N. C. Pickering, *The Bowed String*, Amereon, Ltd., Mattituck NY, 1991, Also available from Bowed Instruments, 23 Culver Hill, Southampton, NY 11968.

[15] J. O. Smith, "Physical modeling synthesis update", *Computer Music Journal*, vol. 20, no. 2, pp. 44–56, Summer 1996, available online at http://www-ccrma.stanford.edu/~jos/pmupd/.