Performance Expression in Commuted Waveguide Synthesis of Bowed Strings

DAVID A. JAFFE and JULIUS O. SMITH
Center for Computer Research in Music and Acoustics (CCRMA)
Department of Music, Stanford University, Stanford, CA 94305
Tel: (415) 723-4971; email: daj@ccrma.stanford.edu / jos@ccrma.stanford.edu

ABSTRACT: In [Smith 1993], an approach was described for implementing efficient real-time bowed string synthesis. Recent work has focused on differentiating the members of the violin family, as well as on the flexibility necessary to create expressive performance. This paper presents a technique for creating smooth transitions between notes, enabling a variety of bowing styles to be synthesized, such as legato, *marcato* and *martele.* A method for supporting such left-hand techniques as vibrato and glissando is also given, as is the efficient simulation of pitch-correlated bow noise. Examples from various periods of music history have been convincingly synthesized in real time using the Music Kit and DSP56001 under NEXTSTEP.

1. Motivation—Better Bowed and Plucked String Instruments

The current de facto standard in string synthesis is a MIDI keyboard controlling a sampler. There are two main problems with this approach. First, the keyboard model is inadequate to represent the continuity possible with stringed instruments. Second, sampling is notoriously inflexible when it comes to expression and nuance.

1.1 Efficient Physical Models

Physical modeling side-steps the problems of sampling by beginning, not with an acoustic waveform, but with the physics of the sound-producing mechanism itself. Physical modeling parameters are "just right", in that they correspond to real-world performance parameters and allow for a great deal of expressive variety in the context of a single perceived source [Jaffe, 1995].

Physical modeling of bowed strings in the past has suffered from great computational expense (as in the case of finite-element modeling) [Ruiz 1971]. In the mid-80s Smith developed a violin model based on non-linear bow dynamics [Smith 1986]. This model, while extremely effective, was still fairly expensive because it required a high-order filter to model the body of the violin. It also suffered from sensitivity to parameter values—if the parameters were not precisely right, it might not even oscillate at all. In 1993, Smith presented an alternative model in which the body filter is permuted with the string and combined with the excitation function, greatly reducing the computational expense of the model and removing the problem of the parameter sensitivity [Smith 1993].

Similarly, the plucked string synthesis technique originally described by the authors [Jaffe/Smith, 1982] can be improved through the use of this refinement. The new version is capable not only of representing a specific type of plucked string instrument (e.g. a viola), but even a particular instrument (e.g. an Amati with a 19th century top and an open crack in the back.)

A detailed discussion of the bowed and plucked string models is beyond the scope of this paper. Instead, we provide an overview, focusing on specific techniques to produce expressive effects. For clarity, we discuss only the bowed string, though most of these techniques apply equally to the plucked string.

1.2 Expressing the Connection Between Notes

Before synthesizing expressive musical phrasing, we need a language to depict it. Following the example of the NeXT Music Kit [Smith, Jaffe, and Boynton, 1989], we define a "phrase" in a manner slightly different from its conventional musical meaning. Our phrase consists of a series of connected notes on a given "voice", where each voice is capable of producing one simultaneous note at a time. A phrase begins with a noteOn event. If another noteOn is received, it is considered a "smooth rearticulation". In the case of the violin, this means that either the bow was changed or the left hand fingering was changed or both. If a noteOff is received, the phrase begins its concluding portion, which takes a certain amount of time. If no noteOn is received before the concluding portion has ended, the phrase is considered finished. If, on the other hand, a noteOn does arrive before the concluding portion is finished, the noteOn is again considered a "smooth rearticulation". Phrases are delineated by one or more samples of silence in a given voice. In MIDI parlance, a phrase corresponds to a series of overlapping notes on the same MIDI channel, with the synthesizer in MIDI Mono Mode.

2. Bowed String Model

The bowed string model consists of three computational blocks, the "bow", the "string", and a

"pitch/vibrato" module. The pitch/vibrato module, controls both the bow and the string. The bow feeds its output to the string, which in turn produces the sound.

2.1 Bow

The heart of the bow is a periodically triggered excitation table ("PET"), where the excitation table is derived (off line) from measurements of real instruments. The PET module plays out the excitation table, jumping back to the beginning of the table once per period of sound, similar to the VOSIM and Chant vocal synthesis techniques [Roads 1989]. When the excitation table is longer than the pitch period, overlaps result. These can be managed by a "multi-PET controller" which triggers a "sub-PET" each period, cycling through a given maximum number of sub-PETs. The outputs of the overlapping sub-PETs are added together to form the overall output. For proper tuning, the multi-PET controller "counts down" an extended precision period duration (in samples) having a fractional part. The period length may in effect be rounded to the nearest sample when an excitation table is triggered to avoid the expense of interpolating the samples in the table.

The bow (or pick) position can be represented as an all-zero comb filter that simply subtracts a delayed version of the PET output, with the length of the comb filter corresponding to the distance from the bridge [Jaffe and Smith 1983]. The effect of bow position illusion is further strengthened by simulating the burst of noise that is produced when a string slips under a bow [Chafe 1990]. This noise occurs naturally when the string is slipping under the bow, and its duration per period is obtained by multiplying the period by the distance from the bow to the bridge divided by the string length. In principle, the string-slip noise must be convolved with the impulse response of the instrument body. In our implementation, when the PET controller triggers a new period, it also triggers a bow-noise pulse and provides it with a random number to be used as an amplitude scaling for the entire pulse. The envelope of this pulse consists of a rapid rise followed by a slower decay, mimicking the effect of the convolution that occurs in the real-world case. As a rule of thumb, we found that a decay value of approximately five times the bow position works well. We then pass that signal through a one-pole low-pass filter for control of musical dynamics.

For plucked strings, the PET simply operates in a one-shot mode, e.g., by giving it a desired pitch of 0**.**

2.2 String

The string portion of the model is a digital waveguide string [Smith 1992] in which a delay line holds approximately one period of sound at all times. The output of the delay line is lowpass filtered, summed with the bow output, and fed to the delay line input. There are three aspects of the string filter that effect the resulting timbre: frequency-independent attenuation, frequency-dependent attenuation and frequency-dependent dispersion [Jaffe/Smith 1983]. For bowed strings, frequency-independent attenuation is the most important of the three, and it controls the sustain quality of the string.

Our strategy for producing smooth glissandi without discontinuities proceeds as follows: We pre-allocate a delay memory block long enough to represent the lowest possible pitch. The delay length, a time-varying signal, controls the offset of the read pointer with respect to the write pointer. If this offset does not fall on a sample boundary, the delay module computes an interpolated value. The interpolation also has the effect of allowing arbitrarily fine tuning. We use simple linear interpolation. Due to the presence of vibrato, the differences in filtering by the linear interpolation for different notes is not significant and amounts to a slight low-pass filtering on average. This slight time-varying filtering effect in the interpolator can be compensated in the string loop filter, or allpass interpolation can be used.

For legato transitions, we cannot abruptly change the length of the delay, because this will cause a sharp discontinuity, which will then be recirculated, resulting in a "spurious pluck." Instead, we branch the string into two different length strings, representing the two pitches. Two delay pointers read the same delay memory, with the pitch period of each driven by its own time-varying offset signal. The output of these readers is then fed to a module that interpolates between the two delay outputs, driven by a cross-fade shaped like a half of a cosine cycle. Before the beginning of a legato transition, the first reader is at the old pitch period and the cosine ramp is at 0. At the start of the legato transition, the second reader is set to the new pitch period and the cosine ramp moves from 0 to 1. For the next legato transition, the process is repeated in reverse. Note that this method is applicable not only to string synthesis, but also to commuted synthesis of any quasi-periodic tone [Smith 1993]. Note also that the highly non-physical nature of this legato algorithm is due to the non-commutativity of the time varying string with the body resonator.

Glissandi and legato transitions done in this way have implications for the bow position filter. To keep a constant tone, violinists tend to keep the bow at a distance from the bridge that is a fixed percentage of the string length. This helps maintain a consistent timbre as the string shortens. In our implementation, this means the comb filter needs to be shortened as the string is shortened so that the zeros continue to fall in the

same place relative to the harmonics of the waveform. While this can be done by using a branched interpolating delay similar to that used in the string, we did not want to pay the extra computational expense. Instead, we simply accept the change in timbre. In fact, changing the bow position slightly between phrases is an effective way to add variety and color.

One last subtlety: When the phrase starts, it is essential to do something to make sure that the garbage left over in the delay memory does not get incorporated into the new note. An efficient approach is to use a switch to break the feedback path of the string. A new phrase begins with this path disconnected. Then, after one period, when the delay line is filled with valid samples, we enable the string feedback path.

2.3 Frequency Control

Since both the bow and string require frequency control, it is convenient to factor it out into its own computational block that produces a signal specifying the instantaneous pitch period of the sound to be synthesized. It combines periodic and random vibrato components with a fundamental frequency which may be enveloped to produce glissandi. A high-quality oscillator should be used to produce the vibrato or noticeable quantization in the frequency signal will result, causing a rough sound in the string. The random vibrato can be produced with a one-pole filtered noise generator.

For legato transitions, the pitch/vibrato controller produces two pitch periods, corresponding to the frequencies of the old note and the new note. Each of these is used to control one branch of the string, as described in section 2.2 on legato transitions. Such a split is not required for the PET. If the pitch period becomes shorter, the multi-PET controller simply starts triggering excitation tables at a faster rate. The question does arise, however, as to when in the course of the legato transition the multi-PET controller should start triggering at the new rate. We found that changing the PET frequency at the midpoint of the transition produced the best results.

Transition times between 15 and 30 milliseconds are most effective. Transition times less than about 10 milliseconds cause audible thumps which sound like the finger of the left hand hammering down on the string, causing an injection of some energy into the string. Transition times greater than 50 milliseconds cause a noticeable inharmonicity consisting of the least common multiple of the two frequencies. The real-word analog would be a forced-oscillation of a string, where the oscillator is at a different frequency from the string.

3. Learning to Play the Synthetic Violin

Making an expressive bowed string score using this model draws upon the same experience needed to coach a human violinist. Thus, a knowledge of string technique and style is invaluable. We discuss a number of examples taken from violin performance practice and show how they can be represented.

3.1 Shifting the Position—The One-Finger Case

When a violinist needs to go from a low position of the left hand to a higher position, or vica versa, he or she performs a shift. A simple glissando, coupled with a bow amplitude envelope that has a notch on rearticulation, simulates a one-finger shift. A typical amplitude envelope is shown here in Music Kit ScoreFile notation in which the breakpoints are specified as a list of (x,y) pairs, where x = breakpoint time and y = target amplitude:

```
[(0,0)(.15,1)|];                          // For the first noteOn
[(0,0)(.05,0.1)(.15,1)|];                 // For the rearticulation
```

The instrument receives a noteOn and begins playing the first envelope. After 0.15 seconds, the amplitude envelope is at 1.0. Some time later, another noteOn occurs. The amplitude envelope moves to the rearticulation point (an amplitude of 0.1) in 0.05 seconds. Then it moves to the stick point (1.0) in another 0.1 seconds. This creates a notch—the amplitude is reduced during the shift. The following scorefile excerpt performs the shift:

```
vln (noteOn,1) ampEnv:[(0,0)(.15,1)|] freq:a4;
t +1;                                     // Advance time
vln (noteOn,1) ampEnv:[(0,0)(.05,0.1)(.15,1)|]
               freqEnv:[(0,0)(.15,1)] freq0:a4 freq1:g4;
```

3.2 Shifting the Position—the Two-Fingered Case

There are two kinds of two-finger position shifts, the traditional type, which is by far the most standard, and the French impressionist type, used in the music of Debussy and Ravel. Both can be implemented as a combination of a glissando and a legato transition. The difference between the two

methods depends on the order of the glissando and legato transition. The traditional method proceeds as follows:

1. Lighten pressure on the currently held left-hand finger.
2. Slide the finger to the new position.
3. Press a new finger down on the finger-board.

This can be implemented as a glissando transition with bow amplitude notch, followed by a legato transition.

In contrast, the French type proceeds as follows:

1. Lighten pressure on the currently held left-hand finger.
2. Press the new finger down on the finger-board.
3. Slide the finger to the new position.

This can be implemented as a legato transition, followed by a glissando transition with bow amplitude notch.

3.3 Bowing Styles

Bowing styles are conveyed largely by the bow amplitude envelope, which tends to be somewhat differently shaped from conventional computer music usage. This is because the actual amplitude envelope of the bowed string model is the that of the string output which resonates at the bow signal frequency. As an example, a simple rectangular envelope for the bow signal produces an exponential rise, steady state, and exponential decay in the final output, and in fact sounds quite natural for short envelope durations. As a result of this indirectness of amplitude control, accents and bow bite must be apparently exaggerated in the bow envelope. For example, a *marcato* bow stroke might appear to have three times the ordinary amplitude during the attack portion: `[(0,2)(.1,3)(.15,.8)|(.18,0)];`

For bouncing bow strokes, the string is allowed to free-ring as soon as the bow leaves the string. Therefore, an envelope that sustains at a non-zero level until the noteOff occurs is inappropriate.

Here is an example *spiccato* envelope: `[(0,0)(.01,1)(.025,1)(.035,.5)(.1,.5)(.12,0)|(.13,0)];`

This envelope is also suitable for rapid *sautille* bowing, a style that works especially well with the bowed string model, since each bow stroke injects energy into the string, interacting with the energy from the previous bow stroke.

A standard broad bow stroke is the *martele*: `[(0,0)(.0175,.6)(.025,1.2)(.2,1.2)(.25,.8)|(.3,0)];`

Another important bowing style parameter is bow position, which may be adjusted to give *sul ponticello* or *sul tasto* effects.

3.4 Legato String Crossing

The most accurate approach to legato string crossings is to allocate a separate bow module and string module for each virtual string. These can share the same pitch/vibrato unit. However, if chords and double stops are also to be supported, a complete bowed string model is needed for each virtual string.

4 References

Chafe, C. 1984. “Simulating Performance on a Bowed Instrument.” CCRMA Tech. Rep. STAN-M-48, Stanford University.

Chafe, C. 1990. “Pulsed Noise and Microtransients in Physical Models of Musical Instruments.” CCRMA Technical Report STAN-M-65, Stanford University.

Jaffe, D. 1995. *Ten Criteria for Evaluating Synthesis and Processing Techniques*. Computer Music Journal, MIT Press, 19(1):76-87.

Jaffe, D. and J. O. Smith. 1983. “Extensions of the Karplus-Strong Plucked String Algorithm.” *Computer Music Journal* 7(2):56-69. Reprinted in *The Music Machine*, Roads, C., ed., MIT Press, 1989.

Hiller, L., and P. Ruiz, “Synthesizing Musical Sounds by Solving the Wave Equation for Vibrating Objects.” *J. Audio Eng. Soc.*, Part I: vol. 19, no. 6, June 1971; Part II: vol. 19, no. 7, July/Aug. 1971.

Roads, C., ed. 1989. *The Music Machine*, MIT Press.

Smith, J. O. 1986, “Efficient Simulation of the Reed-Bore and Bow-String Mechanisms,” *Proc., International Computer Music Conference (ICMC)*, The Hague, Computer Music Association.

Smith, J. O. 1992. “Physical Modeling using Digital Waveguides.” *Computer Music Journal* 16(4):74-87.

Smith, J. O. 1993. “Efficient Synthesis of Stringed Musical Instruments.” *Proc., ICMC*, Tokyo.

Smith, J. O., and S. A. Van Duyne 1995. “Commuted Piano Synthesis.” *Proc., ICMC*, Banff.

Smith, J. O., D. A. Jaffe, and L. Boynton. 1989. “Music System Architecture on the NeXT Computer.” *Proceedings of the Audio Engineering Society Conference*, Los Angeles.