A MODEL FOR REAL-TIME SOUND SYNTHESIS OF GUITAR ON A FLOATING-POINT SIGNAL PROCESSOR

Matti Karjalainen and *Unto K. Laine*

Helsinki University of Technology, Acoustics Laboratory,
Otakaari 5 A, SF-02150 Espoo, Finland

ABSTRACT

Computational modeling of musical instruments is an alternative to commonly used and more straightforward sound synthesis techniques like FM synthesis and waveform sampling. New powerful signal processors make it possible to run such models in real time. This paper deals with algorithms that can be used to synthesize guitar sounds on a floating-point signal processor. A FIR type Lagrange interpolator is introduced to implement efficient and precise fractional delay approximation that is needed to achieve arbitrary and varying length strings. This kind of interpolation is especially good in avoiding distortion and undesirable extra effects when the string length is changing continuously during the synthesis of a sound. The interpolator can also be used in other cases, e.g. in transmission-line modeling of acoustic tube resonators in wind instruments and also for vocal tract models in speech synthesis. In addition to the interpolation principle the paper describes the implementation of the guitar string model on the TMS320C30 floating-point signal processor.

INTRODUCTION

Real-time sound synthesis in computer music is usually realized by signal processing methods like FM synthesis, additive and subtractive synthesis, waveshaping, and waveform sampling [1]. These methods are popular due to their efficiency and ability to achieve good real-time performance on existing DSP hardware. Computational modeling of acoustic instruments is another approach to sound synthesis but the complexity of the models and the inherent difficulty in detailed modeling have prevented a widespread application of this approach in real-time synthesis. This paper describes a new method of modeling a variable-length string and its application to real-time synthesis of the acoustic guitar. Our implementation of the model runs on the TMS320C30 floating-point signal processor.

The traditional approach to efficient modeling of a vibrating string is to use proper digital filters or transmission lines, see e.g. Karplus and Strong [2] and its extensions by Jaffe and Smith [3] and Sullivan [4]. These show the direction of "semi-physical" modeling where only some of the most fundamental features of the string, especially the transmission line property, are retained to achieve efficient computation. More complete finite element models and other kinds of physical modeling may lead to very realistic sounds but tend to be computationally too expensive for real-time purposes. The transmission line model still bears a relatively close resemblance to the vibration of a real string although the physical parameters are not directly expressed in the synthesis algorithms.

When acceleration is the variable used to describe the vibration of a string then an ideal sharp excitation (plucking) corresponds to an impulse and the system can be considered as a digital filter of a special design. In most cases the effects of damping and dispersion of waves along the string can be lumped to the end points so that the string as such is a two-directional lossless delay line and the reflection of waves at the end points is controlled by simple digital filters including frequency dependent system losses.

Delay line lengths of real (non-integer) values are needed to generate sounds with pitch values on any useful musical scale. This introduces the requirement for fractional delay elements. If the length includes a fractional part but remains constant during sound generation the solution used traditionally is the addition of an all-pass filter (for the fractional part) to an integer-length transmission line [3]. If the length is dynamically varying, however, it is difficult to avoid extra sounds that are due to undesirable excitations of the transmission line when the integer part is incremented or decremented.

Our solution to the problem of a variable-length fractional delay string is to use a FIR-type Lagrange interpolator [5], a generalization of the linear interpolator used in [4]. A careful algorithm design removes all undesirable audible effects in gliding sounds (glissando). The rest of the paper describes the principles of string modeling, our algorithm and its properties, some implementation issues on the TMS320C30 signal processor, as well as aspects of modeling the whole guitar consisting of strings and the acoustic body.

TRANSMISSION LINE STRING MODELS

Fig. 1 shows a simplified diagram of the vibration of a plucked string. The triangle-shaped initial deviation pattern can be interpreted as a composition of two identical waves with half of the total amplitude and opposite propagation directions. Instead of vibration amplitude (deviation), a more common variable in acoustics is its derivative or velocity. The initial state pattern in terms of velocity is a unit step function. This, as well as deviation, is not localized to the excitation point so that it is somewhat inconvenient to control the initial state of the string.

If one more derivation is applied to the initial state function then it will be described in terms of acceleration which, in this idealized case, appears as an impulse. By operating in terms of acceleration the string can be considered as a linear filter, the excitation point being the input and any other point the output where the output signal is obtained by superposition of the left and right travelling wave components.

In an ideal case the waves travel without any attenuation or change in their shape; only the end points reflect the wave by inverting the polarity. In practice the losses and the stiffness of the string material change the situation but still one pass of the

CH2977-7/91/0000-3653 $1.00 © 1991 IEEE

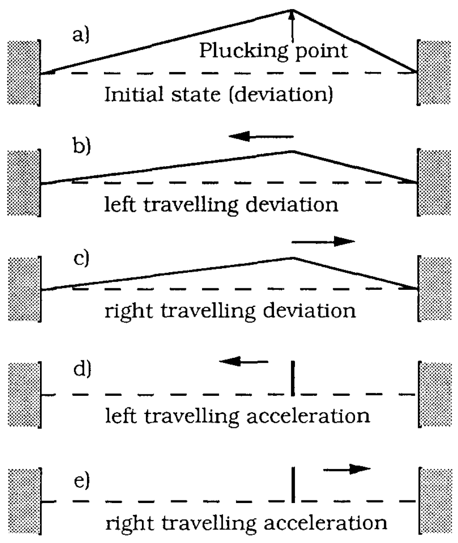
a)
Plucking point
Initial state (deviation)
b)
left travelling deviation
c)
right travelling deviation
d)
left travelling acceleration
e)
right travelling acceleration

Fig. 1. *The principle of wave propagation from the excitation point of string (a) at the moment of plucking in terms of deviation amplitude (b,c) and acceleration (d,e).*

wave over the string is so close to the ideal that the behavior of the string can be approximated well by lumping all deviations from the ideal case to linear filters at the reflecting end points.

In the analog world the string can be easily described by a differential equation (the wave equation). It is difficult, however, to implement the wave behavior by analog techniques. In discrete time domain the ideal one-directional transmission line is very simple: a sequence of delay units. The end points with reflection, attenuation and dispersion properties can be implemented as digital filters. (The Karplus-Strong algorithm is a further simplification of the principle by removing explicit excitation and pickup points.) Fig. 2 shows the principle of digital transmission-line modeling of a string.

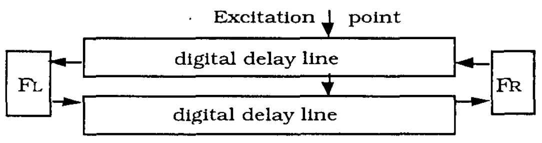

Fig. 2. *Modeling of a string by two one-directional digital delay lines (lossless transmission lines) and two digital filters* (FL, FR) *that implement the end point reflections as well as all losses and dispersive properties of the string.*

The digital delay-line implementation of the one-directional wave transmission is simple only if the desired length of the string corresponds to an integer number of delay elements. The musical scales are not possible to fit using only integer multiples of a fixed string length unit. The trivial solution is to use a signal processor for each string and to change the sampling frequency (clock frequency) of the processor continuously to match the desired vibration frequency of the string. A constant clocking frequency is normally needed, however, especially if several strings are to be computed on a single processor. In such a case fractional delay elements are needed.

FRACTIONAL DELAY APPROXIMATION BY LAGRANGE INTERPOLATION

Fractional delay approximation is a special case of the commonly known interpolation problem. In a discrete time and discrete point delay line the samples **s(n)** at a given moment are exactly specified only at integer valued delay points **n**. What is needed in fractional delay approximation is a good estimate of the signal value **s(n+x)** at a real-valued point **n+x**, where **x** is the fractional part ($0 \leq x < 1$), see Fig. 3.

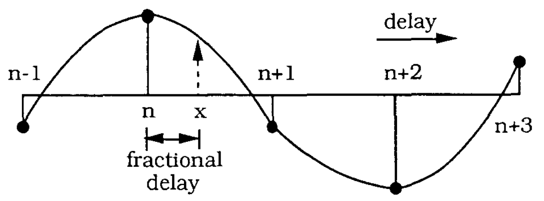

Fig. 3. *Fractional delay approximation means interpolation of delay line sample points by a proper interpolation function.*

As in time domain interpolation of band-limited signals sampled uniformly at the Nyquist rate, the sinc function **sinc(x)** = **sin(x)/x** can be shown to be the ideal interpolation function so that

$$s(n+x) = \sum_i s(n) \operatorname{sinc} \pi(x-i) \qquad \text{for } -\infty < i < +\infty .$$

Since the the summation interval is unlimited it cannot be used in practice as such. Proper windowing (shortening) of the sinc function leads to practical interpolators that approximate the ideal case. The requirements for a good interpolator vary depending on the purpose but in typical applications of fractional delays the following ones can be stated:

- computational efficiency of the interpolator,
- accuracy of the group delay (time domain),
- accuracy of the amplitude response (frequency domain),
- efficient computation of the interpolator coefficients (in cases when the coefficients must be updated often),
- smooth behaviour of the interpolator output when the integer part **n** must be changed in a variable length case.

Instead of a FIR type interpolation the fractional delay can be approximated also by IIR type filters. All-pole filters are efficient but not very attractive otherwise because of problems in achieving good group delay and amplitude response at the same time. All-pass filters have perfect amplitude properties and efficient implementation but for variable length strings the run-time computation of coefficients is slow and a smooth behaviour cannot be guaranteed when the integer length jumps. The following section shows how Lagrange interpolation can solve these problems.

Lagrange interpolator

Intuitively, the most natural way to achieve the interpolation (i.e. fractional delay element $\mathbf{D = xT}$, where $\mathbf{x} \in \mathbf{R}$, $\mathbf{0 \leq x < 1}$) is to construct a continuous function (polynomial), which goes through the known sample points, and then by using this function, to approximate the signal value at **x**. This method is known as **Lagrange** interpolation. The simplest form of it is the well-known linear interpolation between two points.

The Lagrange interpolator can be designed as a N-tap FIR-filter [5]. In the following we restrict our scope to filters where $\mathbf{N = 2k}$, $\mathbf{k} = \{\mathbf{1, 2}, \ldots\}$. The transfer function of the N-tap Lagrange interpolator is given by the formula:

$$(1)\quad L_{N,x}(z) = \sum_{i=-N/2+1}^{N/2} \lambda_i(x)\, z^{-i}, \quad \text{where } 0 \le x < 1$$

The tap coefficients $\lambda_i(x)$ are given by:

$$(2)\quad \lambda_i(x) = \prod_{j=-N/2+1,\, j\neq i}^{N/2} \frac{x-j}{i-j}$$

Here the Lagrange interpolator consists of all index points **i** used in equation (1) so that **i** = 0 corresponds to point **n** in Fig. 3. To make a causal system the delay line should include so many elements that all interpolator index points are realizable. When **x** = 0.0 the interpolator is inactive with $\mathbf{L(z)} = \mathbf{z}^{-0} = 1$ and when **x** = 1.0 it represents an exact delay element with $\mathbf{L(z)} = \mathbf{z}^{-1}$. Active interpolation occurs between these values. When the lengthening of the delay continues over one element the whole interpolator is shifted one step further (i.e. one element added to the delay line). The choice of **N** to be even helps to continue the interpolation in a new shifted position without any additional transients caused by the shifting itself.

In real time applications the number of taps (**N**) of the interpolator have to be reduced to about 2 - 6. This makes the modeled delay to differ from the ideal fractional delay **D** and also the amplitude response of the FIR to differ from the ideal all-pass characteristics.

As earlier analyzed by Laine [5], the amplitude response of the Lagrange interpolator introduces towards the folding frequency a monotonically increasing attenuation. The largest attenuation occurs when **x** = 0.5. In this case a zero appears on the unit circle at the folding frequency. This corresponds to the case when the fractional delay is exactly modeled.

The fractional delay will be modeled with high accuracy approximately up to half of the folding frequency. The largest error appears at the folding frequency, when **x**≈0.4 (or 0.6) [5]. By increasing the order of the FIR (or the samping frequency) the useful bandwidth of the interpolator can be increased.

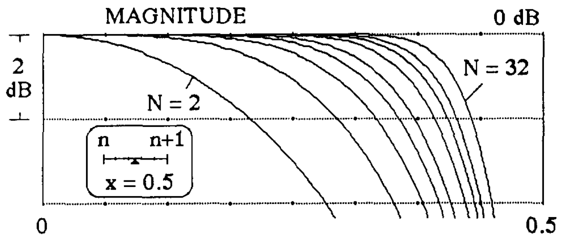

Fig. 4. *Magnitude of N-tap Lagrange interpolator when x=0.5 and N=2,4,8,10,14,18,22, and 32.*

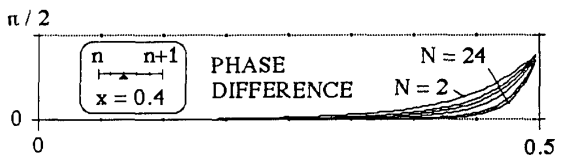

Fig. 5. *Lagrange interpolator phase difference to ideal delay (linear phase) when x=0.4 and N=2,4,6,10, 22 and 24.*

Fig. 4 visualizes the interpolation at **x**=0.5 with different orders of Lagrange interpolators (number of taps, **N**, varies from 2 to 32). The -1 dB point moves towards the folding frequency when **N** increases being about 0.413 in relative frequency when **N** = 32. The corresponding phase difference curves are shown in Fig. 5. Here the error is analyzed when **x**=0.4.

IMPLEMENTATION MODEL

Two transmission lines with an integer number of unit delays plus a fractional delay are needed for the string model. Fig. 6 shows the basic principle where the end-point reflections are carried out at the other end of the delay lines (arrow up) and at a fractional point **x** (arrow down). Dashed lines show the direction of the data transfer.

Notice that the signal at point **x** is interpolated *from* the upper line and after reflection filtering it is interpolated *to* the lower line by superposition to the proper integer index points on the zero-valued samples flowing from the left. By this alignment of the integer index points of both lines the excitation input as well as pick-up outputs may be positioned in any integer index point. (There is no reason in practice to use computationally expensive interpolation for input and output points.)

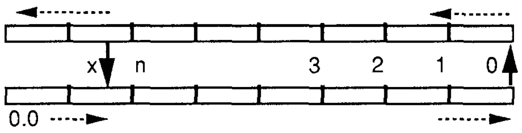

Fig. 6. *The basic implementation model for the string by using interpolation from and to a fractional delay point at the other (left) end of the string model.*

It is desirable to make a minor modification to the model and to use only one common fractional delay for two reasons; (a) to make the computation more efficient and (b) to avoid transients when the integer part of the delay jumps in a continuously variable length string. Fig. 7 shows this principle where the length of one line is kept integer and the other contains the sum of the fractional lengths of Fig. 6.

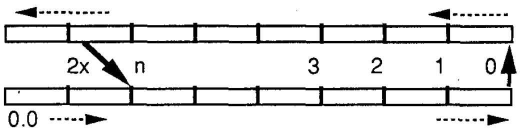

Fig. 7. *A modified implementation model where only one interpolation is needed.*

With this principle when the fractional part **x** grows from 0.0 to 0.5 the corresponding interpolation point *from* the upper line moves from 0.0 to 1.0 and the reflected value is returned *to* integer point **n**. When **x** grows from 0.5 to 1.0 the reflection is taken *from* integer point **n**+1 and the result is interpolated *to* **n**+2(**x**-0.5). Notice that, contrary to the case in Fig. 6, no actual discontinuity jump in interpolation occurs and thus no transients or distortions are generated in the transmission lines. Otherwise the model in Fig. 7 behaves like the basic model in Fig. 6.

Algorithm for the computation of the string model

The algorithm to compute a single sample of the string output on the TMS320C30 is the following (refer to Fig. 7):

1) Low-pass filter the input impulse (plucking) by a proper smoothing filter (unless sharp ideal plucking is desired).
2) Feed one half of the filtered input to each transmission line to the (integer) position of the plucking.
3) Move address pointers to **n**. If $0.0 \leq \mathbf{x} < 0.5$ interpolate *from* the upper line corresponding to point n+2x, filter it by the end reflection filter and feed back *to* point **n** of the lower line. If $0.5 \leq \mathbf{x} < 1.0$ read *from* point **n**+1 of the upper line, filter by the end reflection filter and interpolate *to* the lower line corresponding to point **n**+2(**x**-0.5). If some left-hand excitation to the string (by hammering etc.) is generated this should be superimposed to the end reflection point.
4) Read output from pick-up point(s) of both lines and add them. This output corresponds to the acceleration of the string point. It should be filtered further by a filter system, corresponding to the effect of the guitar body, to get the final acoustic output. In a simple case without acoustic body it is enough to integrate the acceleration signal by a (leaky) integrator to get the velocity signal. A part of the string output should also be fed to other strings so that, if free to vibrate (so called sympathetic strings), they too begin to vibrate.

In addition to these operations the interpolation coefficients must be updated if the length of the string is varying. This can be done at a frequency slower than the sampling rate to speed up the total computation.

The control parameters for the string model in our experiment were:

* The maximum length of the string (integer)
* The variable length of the string (float)
* Filter parameters (in floating-point format) for the end-point reflections (float) at both ends of the string, including DC gain (close to -1.0) and filter coefficients of a first or second order low-pass filter
* Excitation point (integer) and filter coefficients (float)
* Pick-up point (integer) and filter coefficients (float)

The control parameters may all be variable according to the sound that is to be produced. Many of the parameters are dependent on the physical properties of the string and the way it is played so that the parameter values may be estimated from physical properties or simply be adjusted to generate sounds that closely resemble those from a good acoustic guitar.

EXPERIMENTS ON THE TMS320C30

The TMS320C30 floating-point signal processor was used for real-time implementation of the string model. Although efficient computation by fixed-point processors is possible, floating-point processors like the C30 have many advantages such as easy and fast programming, wider dynamic range, avoidance of signal level scalings, etc. The C30 has a very efficient *circular addressing mode* that means indirect addressing with post-increment or decrement so that the address register is automatically updated to step around in a circular buffer. It was used to implement the delay line of the string model.

A sampling frequency of 44.1 kHz was used to achieve full audio bandwidth and to minimize the amplitude response problems of Lagrange interpolators. This sampling frequency guarantees a bandwidth of about 5 kHz where the magnitude response is essentially independent of the fractional delay value. More strings can be synthesized on a single processor if the sampling frequency is lowered but the bandwidth problems become more audible. By optimizing the program code for the C30, including memory allocations and bus loadings, some 2 - 6 strings can be computed in real time on a single processor.

A further optimization can be achieved by modifying the "maximally flat" character of Lagrange interpolation to FIR-type interpolation with controlled ripple in the passband. In this case the interpolator coefficients may be computed by proper windowing of the sinc function [6]. The price for the widening of the passband is the much more complex computation of the coefficients. One way to compensate for this drawback is to use table lookup and possible interpolation of the coefficients.

For guitar synthesis experiments we have implemented both a simple sequencer program and a MIDI interpreter so that both presequenced and real-time control of synthesis is possible. It was found that even if isolated sounds of the string model sound very good a longer sequence or piece of music controlled from a MIDI keyboard does not necessarily sound as good. Many details and nuances of control should be added, including e.g. various "extra" excitations to the string and the loading of the touch point of the string during normal playing.

FROM STRING TO GUITAR MODEL

Successful modeling and synthesis of a string is only a part of synthesizing the whole guitar. The coupling of strings to the body through the bridge and to other strings should be modeled as well as the resonating and radiating properties of the body. This constitutes a very complex transfer function from the string to the listener, the acoustic body being the most difficult part for real time synthesis. LPC analysis of the response of an impulse-like artificial excitation to the bridge shows that an IIR filter with 200 to 500 poles is needed for relatively detailed modeling of this transfer function. A careful selection of a smaller number of low and middle frequency resonances is enough for fairly natural sounding synthesis. (Even a proper spectral envelope without resonances does not sound too bad, which shows that the primary role of the guitar body is amplification.) The results of these studies will be published later.

Real-time simulation of the guitar body with high fidelity requires one processor with C30-level performance. Six strings take 1 to 6 processors depending on the level of sophistication. Multiprocessor realizations are thus needed for high-quality full-scale synthesis when using present signal processors. Next generation DSP's may run the task on a single chip and multiprocessor configurations will allow for essentially higher level of details and improved naturalness. Modeling of acoustical instruments is a feasible and attractive approach to high-quality real-time sound synthesis.

ACKNOWLEDGEMENTS

This study is part of a research project that is financed by the Academy of Finland.

REFERENCES

[1] C. Roads, S. Strawn (ed.), *Foundations of Computer Music*. MIT Press, 1985.
[2] K. Karplus, A. Strong, "Digital Synthesis of Plucked String Algorithms," *Computer Music Journal* (*CMJ*), 7(2), 1983.
[3] D.A. Jaffe, J.O. Smith, "Extensions of the Karplus-Strong Plucked String Algorithm," *CMJ* 7(2), 1983.
[4] C.R. Sullivan, "Extending the Karplus-Strong Algorithm to Synthesize Electric Guitar Timbres with Distortion and Feedback," *CMJ*, **14**(3), 1990.
[5] U.K. Laine, "Digital Modelling of a Variable Length Acoustic Tube," in *Proc. of Nordic Acoustical Meeting*, Tampere. The Acoustical Society of Finland, 1988, pp. 165-168.
[6] T. Laakso, U.K. Laine, and M. Karjalainen, "Approximation of Fractional Delay by FIR Filters". To be published.