Mikael Laurson,* Cumhur Erkut,† Vesa Välimäki,† and Mika Kuuskankare*
*Center for Music and Technology
Sibelius Academy, Helsinki, Finland
http://cmt.siba.fi
†Laboratory of Acoustics and Audio Signal Processing
Helsinki University of Technology, Espoo, Finland
http://www.acoustics.hut.fi/

Methods for Modeling Realistic Playing in Acoustic Guitar Synthesis

Sound synthesis based on physical modeling of stringed instruments has been an active research field for the last decade. The most efficient synthesis models have been obtained using the theory of digital waveguides (Smith 1992). Commuted waveguide synthesis (Smith 1993; Karjalainen et al. 1993) is based on the linearity and time-invariance of the synthesis model and is an important method for developing a generic string instrument model. Recently, such a model has been presented including consolidated pluck and body wavetables, a pluck-shaping filter, a pluck-position comb filter, string models with loop filters and continuously variable delays, and sympathetic couplings between the strings (Karjalainen et al. 1998).

Our model is realized in a real-time software synthesizer called PWSynth. PWSynth is a user library for PatchWork (Laurson 1996) that attempts to effectively integrate computer-assisted composition and sound synthesis. PWSynth is a part of our project that investigates different control strategies for physical models of musical instruments. PatchWork is used also to generate control data from an extended score representation, the Expressive Notation Package (ENP) (Laurson et al. 1999; Kuuskankare and Laurson 2000; Laurson 2000).

Calibration of the synthesis model is based on the analysis of recorded guitar tones (Välimäki et al. 1996; Tolonen 1998). A recent article (Erkut et al. 2000) addressed the revision of the calibration process to improve efficiency and robustness. It also proposed extended methods to capture information about performance characteristics such as different pluck styles, vibrato, and dynamic variations of a professional player. In addition, the article presented basic techniques for simulation of the transients. Instead of using a detailed finger–string interaction model like that proposed by Cuzzucoli and Lombardo (1999), the simulation consolidates all the transient effects into the excitation signal and the update trajectories of the model parameters.

Computer Music Journal, 25:3, pp. 38–49, Fall 2001

The current article summarizes our achievements in model-based sound synthesis of the acoustic guitar with improved realism. First, a simplified physical model of a string instrument realized in our work is described. The next section discusses the calibration of the synthesis model. Then, we address controlling the synthesizer using ENP. After this, we provide an overview of the real-time synthesizer PWSynth. The final section discusses how we simulate various playing styles used in the classical guitar repertoire. Musical excerpts related to this article will be included on the forthcoming *Computer Music Journal 25:4* compact disc.

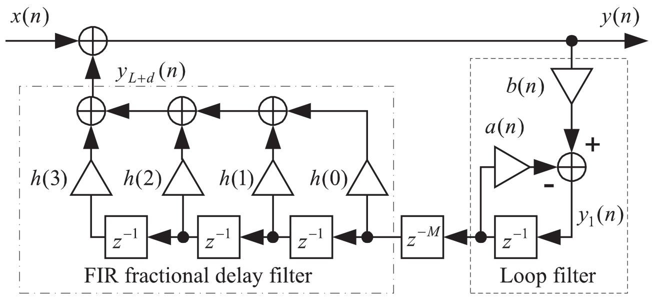
Structure of the Synthesizer

We have implemented a string instrument model that is based on the principle of commuted waveguide synthesis. We now present both the basic string model and a guitar string model that contains two basic models.

Basic String Model

A model for a vibrating string is the only part of the system that explicitly models a physical phenomenon. Our string model implementation is illustrated in Figure 1. It is a feedback loop that contains a delay line and two digital filters, as suggested previously in the literature (Jaffe and Smith 1983; Välimäki et al. 1996). The input signal $x(n)$ of the system is obtained from a recorded guitar tone, as described later. The digital filter seen in Figure 1

Figure 1. Block diagram of the basic string model.



inside a box drawn with a broken line is called the loop filter. Its output signal is computed as

$$y_1(n) = b(n)y(n) - a(n)y_1(n-1) \tag{1}$$

where n is the discrete time index, $a(n)$ is the feedback coefficient, and $b(n) = g(n)[1 + a(n)]$ is the gain coefficient of the loop filter. The magnitude of both g and a must be less than 1 to ensure that the feedback loop will be stable. Usually, g is slightly smaller than 1 and a is slightly smaller than 0, in which case the filter has a gentle lowpass characteristic. Note that the loop-filter coefficients $a(n)$ and $b(n)$ are allowed to be time-varying in Figure 1, since they must be changed, for example, during attenuation or re-plucking of the string.

In Figure 1, the delay line denoted by z^{-M} and the FIR filter with coefficients $h(k)$—which acts as a fractional delay filter—together constitute the loop delay which controls the pitch of the synthetic tone. The length of the delay line is $M = L - 2$ samples, where L is the integer part of the loop delay. The output of the four-tap FIR filter is computed as

$$\begin{aligned} y_{L+d}(n) = h(0)y_1(n-M-1) + h(1)y_1(n-M-2) \\ +h(2)y_1(n-M-3) + h(3)y_1(n-M-4) \end{aligned} \tag{2}$$

where $h(k)$ are the third-order Lagrange interpolation coefficients that are functions of the fractional delay parameter d:

$$\begin{aligned} h(0) &= -\frac{1}{6}d(d-1)(d-2) \\ h(1) &= -\frac{1}{2}d(d+1)(d-1)(d-2) \\ h(2) &= -\frac{1}{2}d(d+1)(d-2) \\ h(3) &= -\frac{1}{6}d(d+1)(d-1)\,. \end{aligned} \tag{3}$$

When $d = 0$, all coefficients $h(k)$ will be zero except $h(1) = 1.0$, thus the delay produced by the FIR filter will correspond to one sample, and the overall loop delay in Figure 1 will be $L = M + 2$ samples (assuming that a is small and the loop filter does not contribute much to the delay).

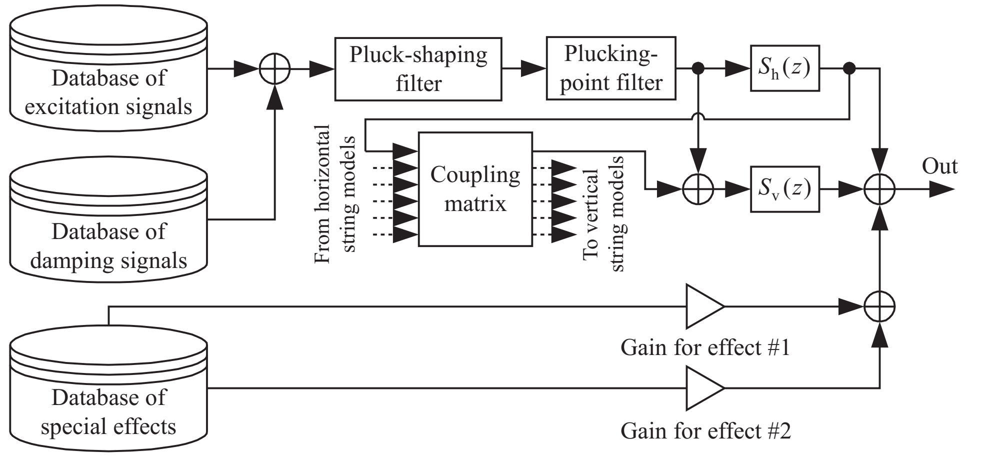
In some studies, an allpass filter is used to implement the fractional delay (Jaffe and Smith 1983), but we prefer an FIR filter, because we find it easier to generate clean vibrato and glissando tones with it. While the Lagrange interpolation filter produces a good approximation of ideal delay at low frequencies, it has a disadvantage: for non-zero values of d, it attenuates high frequencies. If this is harmful, a higher filter order can be used that effectively pushes the problem further towards the Nyquist frequency. According to our experience, a third-order filter is the minimum required for high-quality synthesis at the sampling rate of 44.1 kHz.

Guitar String Model

The structure of the guitar string model is illustrated in Figure 2 for one of the guitar's six strings. Two basic string models of Figure 1 are used for each guitar string, since the horizontal and vertical polarizations of vibration are modeled separately. When one of the string models is slightly detuned, a pleasant beating—characteristic of natural string tones—appears.

The excitation signals are collected in a database. For each tone, an excitation signal is selected according to string and fret numbers. The signal is first processed by the pluck-shaping filter (see Figure 2) and then by a feedforward comb filter that changes the plucking-point effect, if needed. This processed excitation signal is fed into both string models. Note that we could adjust the gain of the excitation signal for each polarization (Karjalainen et al. 1998), but we decided this was unnecessary in practice. The output of the horizontal string model is coupled with the vertical part of all guitar strings to account for sympathetic vibrations in an inherently stable manner (Karjalainen et al. 1998). Note that alternatively vertical models could be coupled to the horizontal ones; the key point,

Figure 2. Block diagram of the guitar string model.


Database of excitation signals
Database of damping signals
Database of special effects
Pluck-shaping filter
Plucking-point filter
$S_h(z)$
$S_v(z)$
From horizontal string models
Coupling matrix
To vertical string models
Out
Gain for effect #1
Gain for effect #2

however, is to ensure stability by avoiding feedback. As seen in Figure 2, the synthetic guitar tone is composed of the sum of the horizontal and vertical output signals. Again, we could use weights in the mixing of these signals, but have decided this is uneccessary in practice.

Furthermore, as indicated in Figure 2, we have created a database of "special effects"—such as rubbing and scraping of the string and various knockings on the guitar body—that are essential in synthesizing modern guitar repertoire. Such samples can be mixed with the guitar signal before playback. In our implementation, two sample players per string can read the effects database simultaneously, allowing another effect to be triggered before the previous one has ended.

To synthesize various playing effects, rules are needed that generate envelopes for parameter values. These are discussed briefly later in this article. Next, we consider how parameter values can be extracted from a recording.

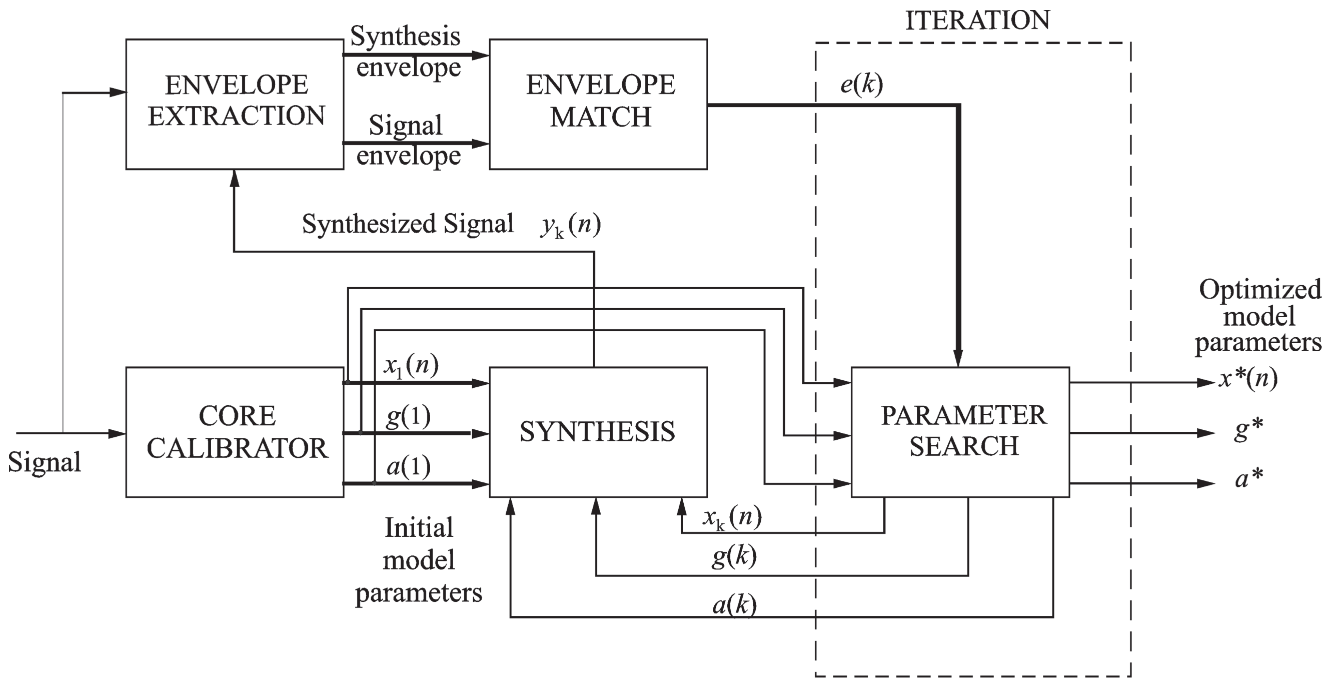
Calibration of the Synthesizer

The calibration of PWSynth can be divided into three subtasks. The first task is the estimation of the basic string model parameters (see Figure 1) for each string and fret, namely, the gain g and the coefficient a of the loop filter, the fractional part d and the integer part L of the loop delay, and the excitation signal $x(n)$. The second task is the calibration of the guitar string model (see Figure 2). Here, a subset of the calibration system is used to capture important characteristics of the analyzed recordings. Instead of providing the final parameters of the model, these tools render restricted initial estimates that can be used for further experimentation. The third task involves the analysis of special data that contains examples of different playing styles. In general, the analysis provides a template for parameter updates. The first and second tasks are presented below, whereas the third task is discussed with the methods for simulation of playing styles.

Calibration of the Basic String Model

The basic string model parameters are estimated with an iterative parameter extraction algorithm described in Erkut et al. (2000). The algorithm first extracts the initial model parameters from the anechoic recordings using a method based on the pitch-synchronous short-time Fourier transform as described in Tolonen (1998). We call this the *core calibrator*. Then, it updates the model parameters to provide a better match between the analyzed and synthesized tones. Figure 3 shows the details of the system.

Figure 3. An extended iterative parameter extraction scheme that matches the overall decay of the analyzed and synthesized sounds, after Erkut et al. (2000).



The core calibrator decomposes the analyzed tone into its deterministic and residual parts, just like the spectral modeling synthesis (SMS) method described by Serra and Smith (1990). However, in the core calibrator, the underlying model structure that accounts for the deterministic part is the basic string model, rather than the SMS sinusoidal model. This approach allows us to obtain the loop-filter parameters that provide the best match between the string model output and the deterministic part. Unlike SMS, the residual is not modeled as a stochastic process; it is directly used as a component of the excitation signal. The initial levels of the harmonics constitute the other component of the excitation. Tolonen (1998) provides further details.

The use of iteration is motivated by the core calibrator's high degree of sensitivity to measurement noise and the regularity of the analyzed samples. As a consequence, the loop-filter parameter estimates exhibit a large variance depending on the data. Estimation errors in the loop-filter parameters result in an unnatural decay in the synthetic tone. Moreover, the quality of the excitation signal is affected.

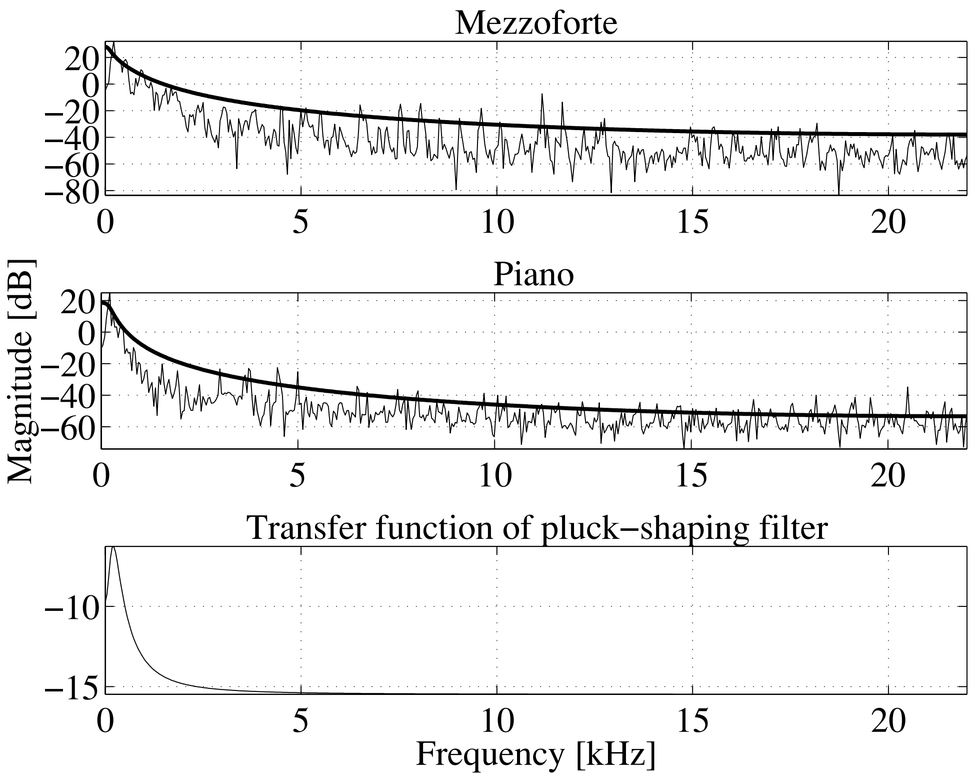
The iterative system shown in Figure 3 synthesizes a signal with the initial parameter estimates of the core calibrator. The amplitude envelope of the synthetic tone is compared to that of the original tone. If there is a discrepancy between the decay of envelopes of the original and synthetic tones, an iterative optimization algorithm is used to detect the optimal loop-filter parameters by minimizing the error, $e(k)$. This approach improves the perceived quality of the synthetic guitar tones.

Recent research has shown that the variation of the loop-filter parameters is not perceived below certain frequency-dependent thresholds (Tolonen and Järveläinen 2000). This property can be incorporated into the envelope-matching algorithm in Figure 3.

Calibration of the Guitar String Model

The excitation signals $x_{ij}(n)$ for each string i and each fret j are obtained from samples of a guitar played *mezzo-forte* using the iterative algorithm discussed above, and they are stored in the database shown in Figure 2. The special-effects data-

Figure 4. Calibration of the biquad pluck-shaping filter. The second-order LPC models of mezzo-forte *(top) and* piano *(middle) excitation signals are drawn with thick lines. The magnitude spectra of the excitation signals are also shown. The difference between the parametric curves is the desired magnitude response of the pluck-shaping filter (bottom).*



base consists of non-processed samples. A small number of damping signals is included in a separate database for simulation of natural-sounding transients. These signals are further elaborated later in this article when we discuss the simulation of playing styles.

The current pluck-shaping filter of PWSynth is a one-pole lowpass filter, similar to the loop filter in Equation 1. This filter converts the *mezzo-forte* excitations into softer (i.e., *piano*) ones. A method based on the deconvolution of two excitation signals differing in dynamics has been proposed for estimation of the pluck-shaping filter coefficients (Erkut et al. 2000). Some coefficients estimated by this method are plugged into PWSynth, but they are manually varied and extrapolated to ensure the synthesis quality over a broad frequency and dynamic range.

Recently, we developed another technique for designing a second-order (biquad) pluck-shaping filter. The design prevents high deviations of the filter coefficients from the noise introduced by deconvolution. In this technique, we represent the individual low-pass characteristics of the excitation signals parametrically and calibrate the filter according to the difference between the parametric curves. The following example (see Figure 4) demonstrates the method.

The excitation signals for *mezzo-forte* and *piano* plucks are parameterized using a second-order Linear Predictive Coding (LPC) fit. The ratio of the parametric curves (i.e., the difference on the dB scale) directly gives the magnitude response of the pluck-shaping filter to obtain the piano excitation signals from the *mezzo-forte* ones. The filtering is carried out with a biquad filter that has the following transfer function:

$$H(z) = \frac{g_p(1 + a_{1m}z^{-1} + a_{2m}z^{-2})}{g_m(1 + a_{1p}z^{-1} + a_{2p}z^{-2})}. \tag{4}$$

In this example, the *mezzo-forte* pluck parameters are $g_m = 0.0479$, $a_{1m} = -1.8746$, and $a_{2m} = 0.8765$, and the *piano* pluck parameters are $g_p = 0.0084$, $a_{1p} = -1.9531$, and $a_{2p} = 0.9541$. Note that the inherent nonlinearities in the energetic excitation signals prevent an exact perceptual match between different dynamics, and the described linear filtering procedure cannot handle these effects. Therefore, nonlinear methods such as frequency modulation may improve the realistic simulation of dynamics.

The excitation signals of the database contain a comb-filtering effect caused by the actual plucking point of the recorded samples. Ideally, this effect should be cancelled by preprocessing and then resimulated at the synthesis level, thereby providing precise timbral control by varying the plucking point. Although the guitar model includes a plucking-point filter, excitation equalization has not been performed in PWSynth, owing to the lack of a method that can reliably estimate the plucking point. Such a method has been recently reported, however (Traube and Smith 2000). Incorporation of this method into the calibration system is left as a future task.

Currently, a method for estimating the delay lengths of a dual-polarization string model is available (Välimäki et al. 1999), and techniques for estimating the other parameters are under development. However, these techniques have not yet been used in PWSynth. Instead, in order to introduce a slight beating, a detuning parameter value for each dual-polarization string model was found by ear.

The calibration of the coupling matrix (see Figure 2) requires a detailed analysis of the string terminations at the bridge and is an open research problem. The coupling matrix coefficients of the PWSynth have been adjusted by trial and error.

Synthesis Control Using ENP

In the following, we discuss a music notation software package called ENP that we currently use for controlling the synthesizer. The use of notation in expressive synthesis control is motivated by the lack of adequate real-time controllers, the familiarity of music notation, and the required precision of control. The use of music notation requires no special technical training, which in turn makes it possible to employ professional players to test and verify various physical models at a deeper level than before. This kind of collaboration is of course crucial, as it allows the combination of the both musical and technical expertise.

Expressive Notation Package (ENP)

The user enters musical material in common music notation into ENP. The system requires no textual input. Users can also add both standard and non-standard expressions that allow them to specify instrument-specific playing styles with great precision. Expressions can be applied to a single note (such as string number, pluck position, vibrato, or dynamics) or to a group of notes (e.g., left-hand slurs or finger-pedals). Groups can overlap, and they may contain other objects, such as breakpoint functions (BPFs). Macro expressions generate additional note events, such as tremolo, trills, portamento, and *rasgueado* (a strumming technique using the right-hand fingernails that is common in the flamenco playing style).

ENP allows fine-tuning of timing with the help of graphical tempo functions. To ensure synchronization of polyphonic scores, all tempo functions are merged and translated internally into a global time-map (Jaffe 1985). Tempo modifications are defined by first selecting a group of notes in the score. After this, a tempo function (group-BPF) is applied to the group. The tempo function can be opened and edited with the mouse. In addition to conventional accelerandi and ritardandi, the user can apply special "give and take" rubato effects to a group. As in the previous case, the user starts with a selection in the score and applies a tempo function to it. The difference, however, is that the duration of the selection is not affected by the time modification. Time modifications are only effective inside the selected group.

ENP also supports user-definable performance rules (Laurson et al. 1999) that allow modification of score information. Performance rules are used to calculate timing information, dynamics, and other synthesis parameters in a way similar to the Swedish "Rulle" system (Friberg 1991). The ENP rules use a syntax that was originally designed for

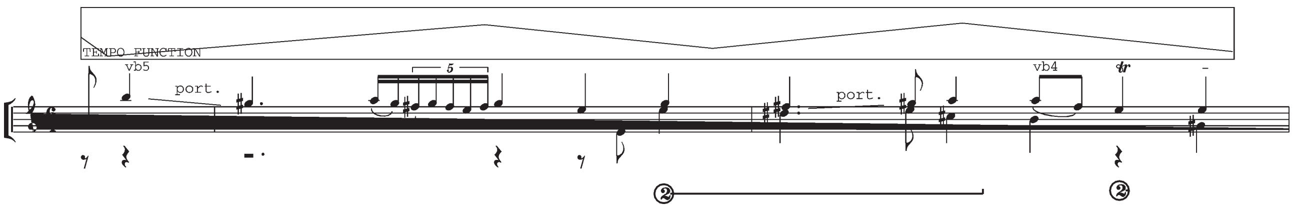


Figure 5. A musical excerpt from the standard classical guitar repertoire.

PWConstraints (for more detail, see Laurson 1996 and Laurson et al. 1999).

After all musical information has been entered, the score is translated into control information. This process is executed in two main steps. First, the note information provided by the input score is modified by the tempo functions and the ENP performance rules. In addition, some instrument-specific rules are applied that further modify the input score. Second, all notes of the input score are scheduled. While the scheduler is running, each note sends a special method to its instrument, which in turn starts other scheduled methods that typically produce the final control data. These methods are responsible for creating discrete control data (e.g., excitation information) or continuous data (e.g., gain of the loop filter, filter coefficients, or other low-level data).

Example from the Classical Guitar Repertoire

Figure 5 gives a fairly typical ENP example from the classical guitar repertoire (a transcription of *Loure* from the E major Partita for lute by J. S. Bach, BWV 1006a). In addition to conventional pitch and rhythm information, the score contains several standard expressions, such as left-hand slurs and the encircled "2," which indicates that the corresponding notes should be played on the second string. Furthermore, there are several portamento expressions (lines marked with "port"). that indicate a rapid glide of the left-hand finger. The non-standard expressions "vb5" and "vb4" denote that the notes in question should be played with a moderate vibrato. The example also illustrates a graphical tempo function that controls the amount of rubato applied to the passage. The slow-moving dance character is kept, however, by ensuring that the cadences are strictly in time. The timing is also modified within the dotted rhythmic motives characteristic of the piece with the help of performance rules.

A Contemporary Notation Example

Figure 6 presents a contemporary excerpt for the classical guitar (from *Lettera Amorosa* by Juan Antonio Muro). This example is interesting from both notational and synthesis perspectives. Note that ENP permits the notation of unmeasured music. ENP also allows the use of non-standard noteheads, which in turn permits the user to express novel instrumental playing techniques. For instance, the first non-standard notehead (the box with a triangularly shaped waveform right after the first run) indicates that the performer should rub the strings with the left hand. The second one (the small box containing the letter "T") stands for a "tambura" effect whereby the player hits the bridge of the instrument with the right-hand thumb. The last one (a note-head with an encircled "x") indicates a hit with the right-hand nail on the body (*golpe* in the Spanish terminology). These extended techniques are synthesized using the unprocessed samples contained in the special effects database shown in Figure 2.

Synthesis Engine

This section describes first the synthesis engine, PWSynth, used in our project. After this, we show how complex instrumental models can be parameterized using special parameter matrices.

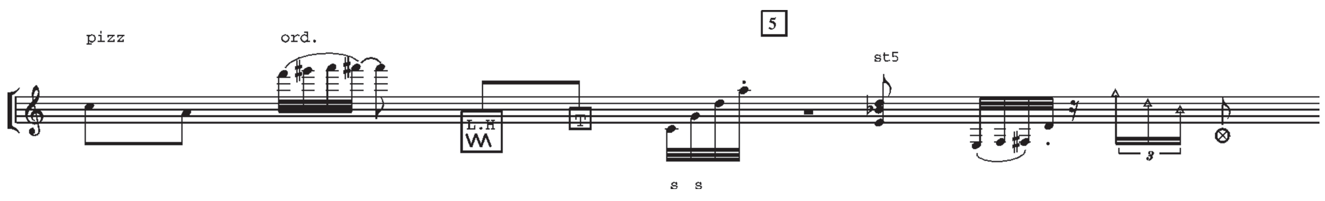


Figure 6. A musical texture with some modern notational conventions.

PWSynth

The starting point in PWSynth is a patch consisting of boxes and connections. This patch is just like any other PatchWork patch except that each PWSynth box contains a private C structure. The C structures inside PatchWork boxes are visible both to the Lisp environment and to a collection of subroutines written in C that are interfaced with the Lisp system. The Lisp environment is responsible for converting a PWSynth patch to a tree of C structures. Also, it fills and initializes all needed structure slots with appropriate data. Once the patch has been initialized, PWSynth calls the main synthesis C routine which in turn starts to evaluate the C structure tree. Real-time sliders and MIDI devices can be used inside the patch.

This scheme supports embedded definitions: a C structure can contain other C structures to any depth. This feature is of primary importance when designing complex instrument models. For example, the guitar model used in our system—consisting of a coupling matrix and six dual-polarization string models—contains a C structure defining the complete instrument. The instrument, in turn, contains seven substructures, one for the sympathetic couplings and six for the strings. The most complex entity used in our example is the dual-polarization guitar string model illustrated in Figure 2. It is built from ten substructures: two delay lines with third-order Lagrange interpolation, two loop filters, three sample players, a delay, a pluck-shaping filter, and a plucking-point filter.

Parameter Matrix

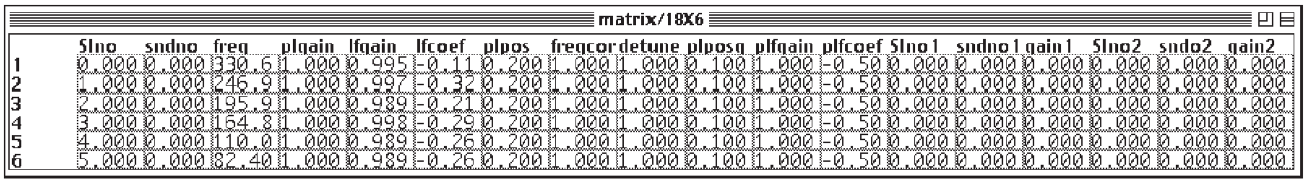
The interaction between ENP and PWSynth is realized in complex cases with the help of graphical matrices (see Figure 7). Each row and column are named. By taking the intersection of the row and column names, a large symbolic parameter space is easily generated. Each resulting parameter name (pointing to an address in a C structure) is a pathname similar to the ones found in other synthesis communication protocols, such as Open Sound Control (Wright and Freed 1997). Thus, if we assume that our current guitar synthesizer is named "guitar1," we can refer to the loop-filter coefficient ("lfcoef") of the second string ("2") of our guitar model with the pathname "guitar1/2/lfcoef." This naming scheme is powerful because it allows the simultaneous use of many instrument instances with separate parameter paths. For instance, we can easily add to a patch new instrument boxes (such as "guitar2," "guitar3," etc.), each having a unique parameter name space.

The rows in Figure 7 indicate the string numbers (1–6), and the topmost row gives the parameter names. The numerical values in the 6 × 18 matrix are used as initial values when starting the synthesis. During synthesis, the control data list, which is generated by an ENP input score, updates the matrix items continuously.

The first two parameter names, "SIno" and "sndno," refer to the excitation sample used by the current string. The system uses a double indexing scheme where the first index refers to a collection of samples or "Sample-Instrument" (SI). The second index points to the actual excitation sample within the current SI. Assuming that the excitation samples for the first string are found in the SI with index 0, we would use an index pair (0, 7) to refer to the excitation sample of the seventh fret of the first string.

The next parameter in Figure 7, "freq," gives the desired frequency of the current string, and "plgain" in turn gives the pluck gain. The next

Figure 7. Graphical parameter name matrix.



matrix/18X6

	Slno	sndno	freq	plgain	lfgain	lfcoef	plpos	freqcor	detune	plposg	plfgain	plfcoef	Slno1	sndno1	gain1	Slno2	sndo2	gain2
1	0.000	0.000	330.6	1.000	0.995	-0.11	0.200	1.000	1.000	0.100	1.000	-0.50	0.000	0.000	0.000	0.000	0.000	0.000
2	1.000	0.000	246.9	1.000	0.997	-0.32	0.200	1.000	1.000	0.100	1.000	-0.50	0.000	0.000	0.000	0.000	0.000	0.000
3	2.000	0.000	195.9	1.000	0.989	-0.21	0.200	1.000	1.000	0.100	1.000	-0.50	0.000	0.000	0.000	0.000	0.000	0.000
4	3.000	0.000	164.8	1.000	0.998	-0.29	0.200	1.000	1.000	0.100	1.000	-0.50	0.000	0.000	0.000	0.000	0.000	0.000
5	4.000	0.000	110.0	1.000	0.989	-0.26	0.200	1.000	1.000	0.100	1.000	-0.50	0.000	0.000	0.000	0.000	0.000	0.000
6	5.000	0.000	82.40	1.000	0.989	-0.26	0.200	1.000	1.000	0.100	1.000	-0.50	0.000	0.000	0.000	0.000	0.000	0.000

pair, "lfgain" and "lfcoef," controls the behavior of the loop filter. The parameter "plpos" gives the current pluck position, "freqcor" is a frequency correction factor used to correct the playback speed of the current excitation sample, "detune" defines the amount of detuning used by the dual-polarization strings of the guitar string model, and "plposg" determines the coefficient of the plucking-point filter (see Figure 2). The pair "plfgain" and "plfcoef" control the pluck-shaping filter. The final six parameters define the parameters of the two extra sample players used to trigger special playing effects.

Simulation of Playing Styles

We now present some ways various playing styles used by a classical guitarist can be simulated with ENP. We describe basic techniques, such as simple plucks, fast "re-plucks" (i.e., repeated plucks), staccato, pizzicato, left-hand slurs, portamento, vibrato, and *forte* playing. General implementation of some of these techniques has been mentioned in Erkut et al. (2000); here, we represent them as implemented in our present system.

Simple pluck events are simulated as follows. The system selects from the database the appropriate excitation sample, the nominal fundamental frequency, and some other controller parameters according to the current string and fret numbers. Amplitude and pluck position values are read from the current note. After the system has written the new parameter values to their appropriate addresses, the sample player module triggers the current excitation signal to produce a tone.

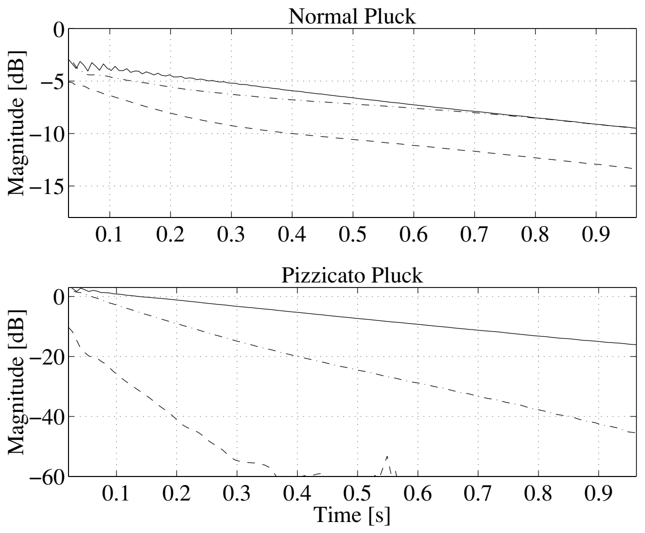
A human player touches the vibrating string momentarily for performing certain basic techniques, such as damping of a vibrating string or re-plucking it. At this instant, the touch splits the string into two portions and constitutes a common lossy boundary condition for each part. The vibration of the string dies out very rapidly due to the losses. Such transient regimes have been presented in Erkut et al. (2000), and it has been shown that their inclusion improves the quality of synthetic tones. The touching point is usually close to the plucking point; hence the portion from this point to the bridge remains the same regardless of the actual fret position. Special excitation signals contained in the database of damping signals corresponding to various touching points are used for simulation of this effect.

The harmonic tones resulting from the string portion between the touching point and the nut have been removed from these short damping signals, so that they can be injected to any note. The update of the loop-filter parameters roughly simulates the rapid decay of the string. The update times of the parameters are based on our previous observations from analyzed examples.

Just before the string is re-plucked, the gain of the loop filter of the current string is lowered to zero in a very short time (typically around 10 msec). For fast re-plucks, that is, when the current string to be plucked is still ringing, the system sends one of the special samples just before the actual pluck excitation. This technique is also used when playing in a staccato style where the player dampens the strings with the right-hand fingers.

In guitar playing, pizzicati are a special class of tones that are produced by plucking the string with the nail of the right hand thumb and then damping the string with the palm of the same hand. The analysis of pizzicati signals suggests that their main difference from normal plucks is the decay characteristics of the harmonics. Figure 8 shows the extracted amplitude envelope trajectories for the first three harmonics of normal and

Figure 8. *Amplitude envelopes of the first (solid line), the second (dash-dot line), and the third harmonic (dashed line) for normal (top) and pizzicato (bottom) plucks.*



pizzicato plucks, respectively. The overall decay rates increase significantly for the pizzicato case, and the figure suggests an additional frequency-dependent decay rate increase for higher harmonics. In our system, the pizzicato effect is accomplished by slightly lowering the gain and the cut-off frequency of the loop filter of the current string. The parameters extracted form the pizzicato sample set using the iterative methods described in the calibration section which provide a template for the range of the alteration.

Although this technique produces reasonable results, the quality of the pizzicato effect may be improved by filtering the excitation signal as well. This prediction is based on several recent observations. Figure 9 shows the Fourier transforms of the normal and pizzicato excitation signals together with the magnitude responses of the corresponding second-order LPC models. The pizzicato excitation signals contain considerably less high-frequency energy compared to those of normal plucks, because the string is damped with the right hand so quickly. The LPC models can be used for composing a biquad pluck-shaping filter (see Equation 4, where the same idea was applied) that converts a normal excitation into a pizzicato excitation.

The left-hand slurring technique is implemented simply by sending a pluck event with a small amplitude value. In this case, the gain of the loop filter is not modified. Portamento is realized with an extremely fast chromatic left-hand slur passage between two notes situated on the same string. (This simulates the movement of the left-hand finger on the fingerboard.)

Additional pitch information is sent from ENP as a scaling factor. If there is no glissando, the factor has a constant value of unity. If the score has a glissando sign between two notes (having the initial and target frequencies *freq1* and *freq2*, respectively), the factor is an envelope starting at 1.0 and ending at the ratio of *freq2* to *freq1*.

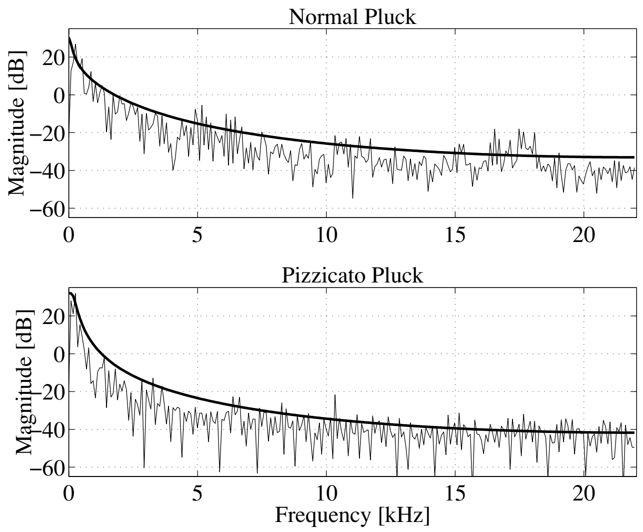
Our system implements a parametric representation of the vibrato control with a rate, maximum depth, and temporal envelope for depth. This implementation is consistent with our observations live player's use of a vibrato in a musical context.

The maximum depth (max-depth) of vibrato is calculated as follows. If the score does not contain any specific vibrato expressions (i.e., we want only to play a "straight" tone), the max-depth value depends on whether the current fret is zero (i.e., an open string) or not. If it is zero, the max-depth is equal to zero. For higher fret values, the max-depth is calculated by adding increasing amounts of vibrato (the amount is always moderate) as the fret value increases. This addresses the fact that, for higher frets, the string is looser, which in turn makes it more difficult for the player to keep the pitch stable. If, however, the score contains vibrato expressions, the vibrato max-depth is calculated depending on the name of the vibrato expression. Vibrato expressions are named using "vb" with a suffix (an integer from one to nine) indicating the max-depth of the vibrato. For example, a slight vibrato is simulated with the expression "vb1," a moderate vibrato is produced with "vb5," and an extreme vibrato is achieved with "vb9."

The rate of the vibrato is normally kept constant (typically around 5–6 Hz). The overall depth, however, is controlled by an envelope scaled to the current max-depth value, with an ascending–descending function with two "humps" to avoid a mechanical-sounding effect when applying a vibrato.

In addition to the pluck-shaping operations mentioned previously, the initial pitch of a note played

Figure 9. The second-order LPC models of normal (top) and pizzicato (middle) excitation signals are drawn with thick lines. The magnitude spectra of the excitation signals are also shown.



forte is somewhat sharp, quickly and smoothly returning to the nominal, fingered pitch value. A more elaborate way to simulate this effect would be to use a tension modulation algorithm (Tolonen et al. 2000).

Conclusions and Future Work

Recent developments in the model-based synthesis of the classical guitar were described, including signal processing and control methods for simulating several playing styles. Examples of short phrases and excerpts from musical pieces that demonstrate the novel capabilities of our classical guitar synthesizer will be available on the forthcoming *Computer Music Journal 25:4* compact disc.

Our future plans include further reduction of the excitation database's size by removing the redundancies between the excitation signals. The parameterization of the low-frequency modes, as suggested previously (Välimäki et al. 1996; Tolonen 1998; Välimäki and Tolonen 1998) is also an attractive method for shortening the excitation signals. We are currently working on the synthesis of different plucked string instruments, such as those from the lute family. The excitation signals and parameter values of the synthesizer must be extracted, and some of the control rules must be customized for each instrument.

Acknowledgments

This research was conducted within the projects "Sounding Score—Modeling of Musical Instruments, Virtual Musical Instruments, and Their Control" and "Sound Source Models—Analysis, Synthesis, and Coding" financed by the Academy of Finland. The work of C. Erkut has been supported by the Jenny and Antti Wihuri Foundation, and the work of V. Välimäki has been supported by a postdoctoral research grant from the Academy of Finland. The authors are grateful to Mr. Klaus Helminen who played the samples used for calibrating the guitar synthesizer.

References

Cuzzucoli, G., and V. Lombardo. 1999. "A Physical Model of the Classical Guitar, Including the Player's Touch." *Computer Music Journal* 23(2):52–69.

Erkut, C., V. Välimäki, M. Karjalainen, and M. Laurson. 2000. "Extraction of Physical and Expressive Parameters for Model-Based Sound Synthesis of the Classical Guitar." Paper presented at the *108th AES Convention*. New York: Audio Engineering Society.

Friberg, A. 1991. "Generative Rules for Music Performance: A Formal Description of a Rule System." *Computer Music Journal* 15(2):49–55.

Jaffe, D. A. 1985. "Ensemble Timing in Computer Music." *Computer Music Journal* 9(4):38–48.

Jaffe, D. A., and J. O. Smith. 1983. "Extensions of the Karplus–Strong Plucked-String Algorithm." *Computer Music Journal* 7(2):76–87.

Karjalainen, M., V. Välimäki, and Z. Jánosy. 1993. "Towards High-Quality Sound Synthesis of the Guitar and String Instruments." *Proceedings of the 1993 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 56–63.

Karjalainen, M., V. Välimäki, and T. Tolonen. 1998. "Plucked-String Models: From the Karplus–Strong Algorithm to Digital Waveguides and Beyond." *Computer Music Journal* 22(3):17–32.

Kuuskankare, M., and M. Laurson. 2000. "Expressive Notation Package (ENP), a Tool for Creating Complex Musical Output." *Proceedings of Les Journées d'Informatique Musicale*. Bordeaux, France: SCRIME, pp. 49–56.

Laurson, M. 1996. "PATCHWORK: A Visual Programming Language and Some Musical Applications." Doctoral dissertation, Sibelius Academy.

Laurson, M., et al. 1999. "From Expressive Notation to Model-Based Sound Synthesis: A Case Study of the Acoustic Guitar." *Proceedings of the 1999 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 1–4.

Laurson, M. 2000. "Real-Time Implementation and Control of a Classical Guitar Synthesizer in SuperCollider." *Proceedings of the 2000 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 74–77.

Serra, X., and J. O. Smith. 1990. "Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic Plus Stochastic Decomposition." *Computer Music Journal* 14(4):12–24.

Smith, J. O. 1992. "Physical Modeling Using Digital Waveguides." *Computer Music Journal* 16(4):74–91.

Smith, J. O. 1993. "Efficient Synthesis of Stringed Musical Instruments." *Proceedings of the 1993 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 64–71.

Tolonen, T. 1998. "Model-Based Analysis and Resynthesis of Acoustic Guitar Tones." Report 46. Espoo, Finland: Helsinki University of Technology, Laboratory of Acoustics and Audio Signal Processing.

Tolonen, T., and H. Järveläinen. 2000. "Perceptual Study of Decay Parameters in Plucked String Synthesis." Paper presented at the *109th AES Convention*. New York: Audio Engineering Society.

Tolonen, T., V. Välimäki, and M. Karjalainen. 2000. "Modeling of Tension Modulation Nonlinearity in Plucked Strings." *IEEE Transactions on Speech and Audio Processing* 8(3):300–310.

Traube, C., and J. O. Smith, 2000. "Estimating the Plucking Point on a Guitar String." *Proceedings of the COST-G6 Conference on Digital Audio Effects*. Verona, Italy: Università degli Studi di Verona, pp. 153–158.

Välimäki, V., et al. 1996. "Physical Modeling of Plucked String Instruments with Application to Real-Time Sound Synthesis." *Journal of the Audio Engineering Society* 44(5):331–353.

Välimäki, V., and T. Tolonen. 1998. "Development and Calibration of a Guitar Synthesizer." *Journal of the Audio Engineering Society* 46(9):766–778.

Välimäki, V., et al. 1999. "Nonlinear Modeling and Synthesis of the Kantele—a Traditional Finnish String Instrument." *Proceedings of the 1999 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 220–223.

Wright, M., and A. Freed. 1997. "Open Sound Control: A New Protocol for Communicating with Sound Synthesizers." *Proceedings of the 1997 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 101–104.