METHODS FOR MODELING REALISTIC PLAYING IN PLUCKED-STRING SYNTHESIS: ANALYSIS, CONTROL AND SYNTHESIS

Mikael Laurson

Center for Music and Technology
Sibelius Academy, Helsinki, Finland
laurson@siba.fi

Cumhur Erkut, Vesa Välimäki

Lab. of Acoustics and Audio Signal Processing
Helsinki Univ. of Technology, Espoo, Finland
cumhur.erkut@hut.fi
vesa.valimaki@hut.fi

ABSTRACT

This paper summarizes our recent achievements in developing signal processing methods for model-based synthesis of plucked-string instruments. The methods discussed are applicable to a broad family of musical instruments, but this paper concentrates on a case study where we focus on the classical guitar. The real-time guitar synthesis engine is based on digital waveguide modeling and commuted waveguide synthesis. The paper provides an overview of our Lisp-based real-time synthesizer and how it is controlled. Furthermore, extended procedures to synthesize transients, pizzicati, and improved dynamics are presented.

1. INTRODUCTION

The model-based sound synthesis of the stringed instruments has been an active research field within the last decade. The most efficient synthesis models are obtained using the theory of digital waveguides [1]. Commuted waveguide synthesis [2, 3] that is based on the linearity and time-invariance of the synthesis model is an important method for development of 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 [4].

Our model is realized in a real-time software synthesizer called PWSynth. PWSynth is a PatchWork [5] user-library that aims at a better integration of computer assisted composition and sound synthesis. PWSynth is a part of our project which investigates different control strategies for physical models of musical instruments. PatchWork is used also to generate control data from an extended score representation (Expressive Notation Package or ENP [6, 7]).

The calibration scheme for this model is based on pitch-synchronous short-time Fourier transform (PS-STFT) of the recorded tones [8, 9]. Given a single recorded tone and the model, it is usually straightforward to synthesize a very realistic replica of the tone. However, the realistic synthesis of individual tones does not guarantee that the model would sound natural in every context.

A recent paper [10] tackled the revision of the calibration process to provide a more robust calibration procedure. It also proposed extensions to capture information about performance characteristics, such as the damping regimes, repeated plucks of a string that is already sounding, vibrato characteristics, different pluck styles, and dynamic variations of a professional player.

This paper summarizes our achievements in model-based sound synthesis of plucked-string instruments with improved realism. The structure of the paper is as follows. A simplified physical model of a string instrument realized in our work is described in Section 2. In Section 3, we discuss the PWSynth implementation of methods proposed in [10], extending the analysis procedure for synthesis of realistic transients, improved dynamics, and pizzicati tones. Section 4 focuses on the control of our synthesizer and describes the ENP-PWSynth interface. Section 5 provides an overview of the real-time synthesizer PWSynth. Musical excerpts related to this paper and our presentation at the DAFX-00 conference are available online at
www.acoustics.hut.fi/demo/dafx2000-synth/.

2. STRUCTURE OF THE SYNTHESIZER

We have implemented a string instrument model that is based on the principle of commuted waveguide synthesis [2, 3]. A model for a vibrating string is the only part of the system that explicitly models a physical phenomenon (i.e., string vibration). Our string model implementation is illustrated in Fig. 1. It is a feedback loop that contains a

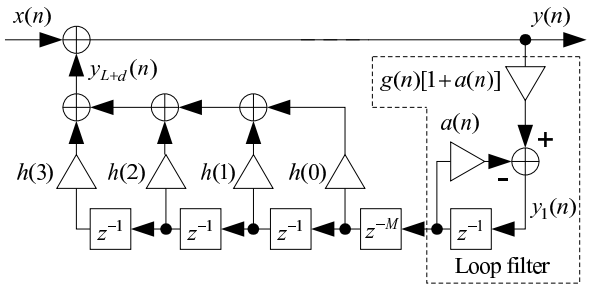


Figure 1. *Block diagram of the basic string model.*

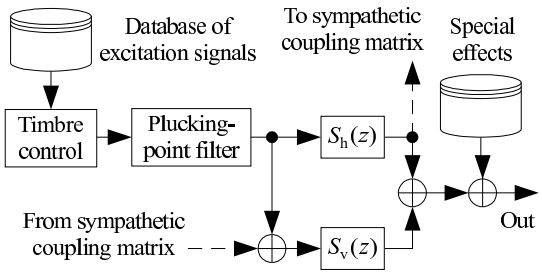


Figure 2. *Block diagram of the guitar string model.*

delay line and two digital filters, as suggested previously in the literature [11, 8]. The digital filter seen in Fig. 1 inside a box drawn with a broken line is called the loop filter. Its output signal is computed as

$$y_1(n) = g(1+a)y(n) - ay_1(n-1) \tag{1}$$

where $g(1 + a)$ and a are the gain coefficient and the feedback coefficient of the loop filter, respectively. 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 character. Note that the loop-filter coefficients are allowed to be time-varying in Fig. 1, since they must be changed, e.g., during attenuation or re-plucking of the string.

In Fig. 1, the delay line denoted by z^{-M} and the 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

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

where $h(k)$ are the third-order Lagrange interpolation coefficients that are functions of the fractional delay parameter d. When $d = 0$, all coefficients $h(k)$ will be zero except $h(1) = 1.0$, and thus the loop delay in Fig. 1 will be L samples (assuming that a is small and does not contribute much to the delay). In some studies, an allpass filter is used to realize the fractional delay [11], but we prefer an FIR filter because with it we find it easier to generate clean glissando tones. 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 which effectively pushes the problem further towards the Nyquist frequency. According to our experience, the lowest filter order that is sufficient for high-quality synthesis at the sampling rate of 44.1 kHz seems to be 3.

The structure of the guitar string model is illustrated in Fig. 2 for one of the six strings. Two basic string models of Fig. 1 are used for each guitar string, since the horizontal and vertical polarization of vibration are tackled separately. When one of the string models is slightly mistuned, a pleasant beating characteristic to 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 and playing style. The signal is first processed by the timbre filter (see Fig. 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 [4], but we felt that this was unnecessary in practice. The output of the horizontal string model is coupled to the vertical part of all guitar strings to account for sympathetic vibrations [4]. As seen in Fig. 2, the synthetic guitar tone is obtained as the sum of the horizontal and vertical output signals. Again, we could use weights in the mixing of these signals, but have decided not to do so.

Furthermore, as indicated in Fig. 2, we have created a database of special effects, such as fret noise sequences and various knockings of the guitar body, which have turned out to be essential in synthesizing modern guitar repertoire. Such samples can be mixed with the guitar signal before playback. In our implementation, two sample players can read the effects database simultaneously, allowing another effect to be triggered before the previous one has ended.

3. ANALYSIS OF RECORDED TONES

The current parameters of PWSynth are based on an iterative parameter extraction algorithm described in [10]. The algorithm first extracts the model parameters from the anechoic recordings of a professional player using the PS-STFT-based methods as described in [8, 9]. A synthetic tone is computed using the estimated parameters, and its amplitude envelope 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. This approach efficiently removes undesired inverse filtering artifacts from the excitation signals.

Two cases of note-to-note transitions are implemented in PWSynth. The first case is the damping of the string where the vibrating string is forced to rest by the player. As suggested in [10], the simultaneous update of the loop-filter coefficients according to the extracted loop-filter coefficient trajectories, and the use of specially obtained damping-excitation signals efficiently simulates the effect of damping. The second case is the repeated plucking of a vibrating string, which includes a sudden damping and re-excitation of the string. The short damping transients between the repeated plucks provide similar trajectories to that of the first case, and similar excitation signals.

PWSynth implements a parametric representation of the vibrato control with a transient time, vibrato frequency, and vibrato depth, extracted by the analysis of several vibrato regimes of the player in a musical context.

The approaches described above can clearly be expanded to simulate a broader set of effects. A consequence of such an expansion is a demand on the excitation database that is of a growing size, where the individual samples are highly redundant. Given the memory limitations and implementation complexity, it is desirable to minimize both the redundancy and the memory needs. In the following, we report two approaches that can efficiently simulate different playing styles using the available excitation signals.

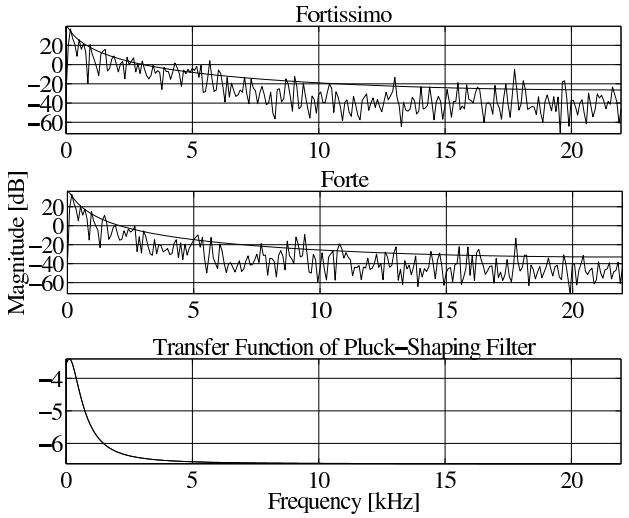


Figure 3: *The calibration of pluck-shaping filter. The frequency responses of fortissimo (top) and forte (middle) excitation signals are represented by smooth curves, obtained by 2nd order LPC fit (superposed thick curves). The difference of these curves is the desired transfer function of the pluck-shape filter (bottom).*

3.1. Modeling and parameter estimation of dynamics

In [10], a method was proposed to obtain the softer excitation signals by proper scaling and low-pass filtering of the most energetic (dynamically speaking) excitation signals with the aid of a pluck-shaping filter (Timbre control in Fig. 2). The coefficients of the pluck-shaping filter were determined by deconvolution and a weighted fit. However, the deconvolution process introduces a considerable amount of noise, which in turn results in high deviations on the pluck-shaping filter coefficients. Therefore, here we represent the individual low-pass characteristics of the excitation signals parametrically, and calibrate the low-pass pluck-shape filter according to the difference between the parametric curves.

The following example (see Fig. 3) demonstrates the method. The excitation signals for fortissimo and forte plucks are parameterized using a second order LPC fit. The difference between the parametric curves directly gives the transfer function of the pluck-shape filter in order to obtain the forte excitation signals from the fortissimo signals. The filtering is carried out with a biquad filter that has the following difference equation

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) \\ -a_1 y(n-1) - a_2 y(n-2) \tag{3}$$

The coefficients of the biquad filter for this example are b_0 = 0.4752, b_1 = - 0.8895, b_2 = 0.4152, a_1 = 1.9076, a_2 = - 0.9089. Note that the bottom part of Fig. 3 plots also the transfer function of the biquad filter, but since the fit is excellent, the transfer function completely overlaps with the difference curve. The method provides very good results for other dynamics as well.

3.2. Modeling and parameter estimation of pizzicato tones

In guitar playing, pizzicati are a special class of tones that are produced by plucking the string with the right hand and at the same time lightly touching (damping) the string with the palm of the same hand.

The analysis of pizzicati signals suggests that their main difference with normal plucks is at the decay characteristics of the harmonics. Fig. 4 shows the extracted amplitude envelope trajectories for the first three harmonics for normal and 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.

These constant and frequency-dependent decay rate alterations are simulated by a new set of loop-filter parameters. These parameters are extracted form the pizzicato sample set, using the iterative methods described above.

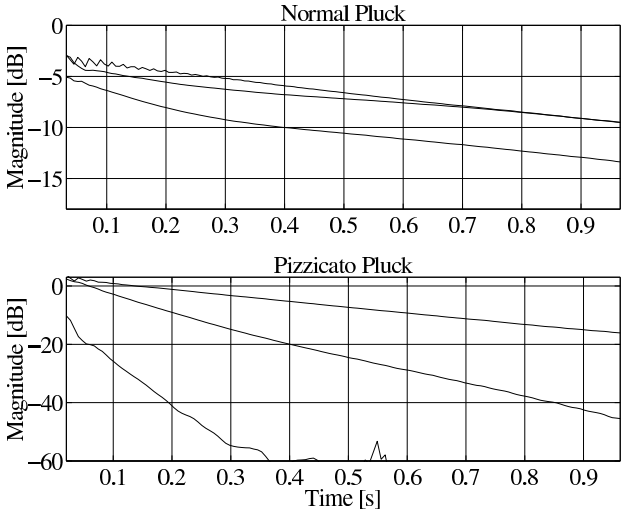


Figure 4: *Amplitude envelopes of the first, (solid line), the second (dash-dot line), and the third harmonic (dashed line) for normal (top) and pizzicato (bottom) plucks. The solid, dash-dot, and dashed curves are the trajectories of the first, second, and third harmonics, respectively.*

Another difference between the normal and pizzicato plucks is found in the excitation signals. Fig. 5 shows the Fourier transforms of the normal and pizzicato excitation signals. The pizzicato excitation signals contain considerably less high-frequency energy, compared to that of normal plucks. This effect is simulated using the dynamics filtering methods described in the previous subsection.

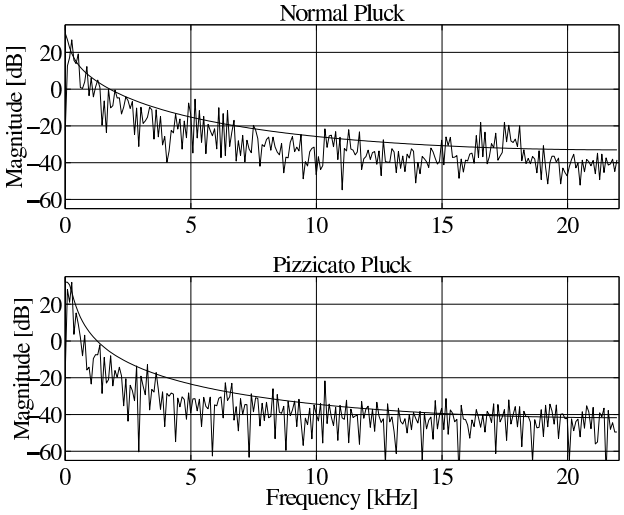


Figure 5. *Fourier transforms of the normal (top) and pizzicato (bottom) excitation signals. The signals are represented by smooth curves, obtained by 2nd order LPC fit (superposed thick curves).*

4. CONTROL

In order to be able to synthesize the various playing effects the extracted analysis data is organized as a large database. The database consists of preprocessed excitation signals, samples, parameter values and rules that generate envelopes.

The user enters in ENP the musical phrase or piece in standard notation. The system requires no textual input and includes facilities for fine tuning of the timing information. The user can also add both standard and non-standard expressions that allow 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 (left-hand slurs, finger-pedals). Groups can overlap and they may contain other objects, such as breakpoint functions. The latter case is called a group-BPF. Macro expressions generate additional note events, such as tremolo, trills, portamento, and rasgueado.

Fig. 6 gives a fairly typical ENP example from the classical guitar repertoire (from "Madroños" by Federico Moreno Torroba). Besides the conventional pitch and rhythm information, the score contains several standard expressions such as left-hand slurs and the encircled '5's which indicate that the corresponding notes should be played on the fifth string. The non-standard expression 'vb9' denotes that the note in question should be played with a strong vibrato, the groups 'pl0' and 'pl1' indicate the pluck position of the right hand, and the 'st5' group gives the amount of staccato to be applied for the notes belonging to the group.

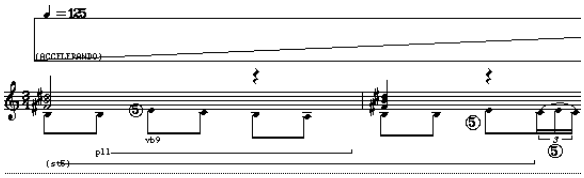


Figure 6. *A musical excerpt from the standard classical guitar repertoire.*

Fig. 7, in turn, gives a more modern excerpt for the classical guitar (from "Lettra Amorosa" by Juan Antonio Muro). This example is interesting both from the notational and synthesis point of view. ENP allows to use non-standard note-heads which in turn permits to express novel instrumental playing techniques. For instance, the first non-standard note-head—the box with a triangle shaped waveform right after the first fast 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 where the player hits with the right hand thumb the bridge of the instrument. The next one indicates a hit or 'golpe' with the right-hand nail on the body, and so on. It is also interesting to note that ENP permits to notate 'non-measured' music (i.e. without measure lines).

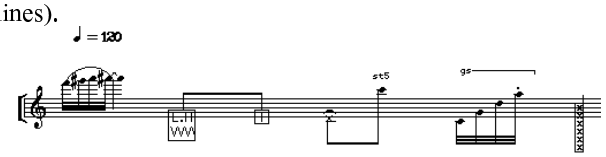


Figure 7. *A musical texture with some modern notational conventions.*

ENP allows fine-tuning of timing with the help of graphical tempo functions. In order to assure synchronization of polyphonic scores, all tempo functions are merged and translated internally into a global time-map. For instance the break-point function in the upper part of Fig. 6 is a tempo function and indicates an accelerando. Besides tempo functions, ENP supports user definable performance rules [6] which allow to modify score information. Performance rules are used to calculate timing information, dynamics and other synthesis parameters.

After the input is finished the score is translated into control information. During this process the expressions trigger instrument specific rules that generate appropriate control data.

The control system is open-ended and can adapt easily to new analysis results in the modeling of musical instruments.

5. SYNTHESIS USING PWSYNTH

The starting point in PWSynth is a patch consisting of boxes and connections. This patch is as any other PatchWork patch except that each PWSynth box contains a private C-structure. Fig. 8 gives a typical example of the system. On the left we have a window containing two patches: the first one, called '*SYNTH*', defines the synthesis part whereas the second one—'*ENP*'—is responsible for the calculation of the control data. The input ENP score is shown in the background behind the main window.

The synthesis part defines—using a box called 'guitar1'—a guitar synthesizer. The 'guitar1' box has three inputs. The first input defines the overall amplitude of the output. The second input is connected to a matrix-object, which contains a 6×6 matrix defining the sympathetic coupling of the strings. The third input is connected to a 6×18 parameter matrix where each row gives the parameters required for a single string. The contents of the parameter matrix is shown below in Fig. 9. Thus our guitar synthesizer consists of a coupling matrix and 6 strings each having 18 parameters. The output of the 'guitar1' box is connected to a 'synth' box, which in turn writes the final result either to the DAC or to a soundfile.

The control information is calculated with a box called 'enp→synth' having three inputs. The first one is an ENP input score (discussed in the previous section). The second input is used for ENP performance rules. The third input defines the output mode for the calculated control information. The current system supports two output-types: 'pwsynth' and 'midifile'. The 'pwsynth' mode generates inside Lisp a control data list that is used to control a PWSynth patch. The 'midifile' mode, in turn, produces a MIDI file that can be used to control an external synthesizer (the 'midifile' mode has been utilized for instance to control a guitar synthesizer in SuperCollider [12]).

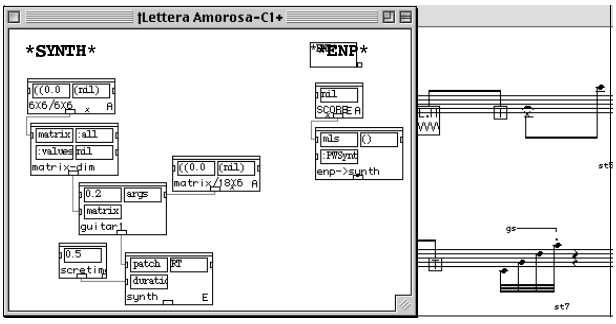


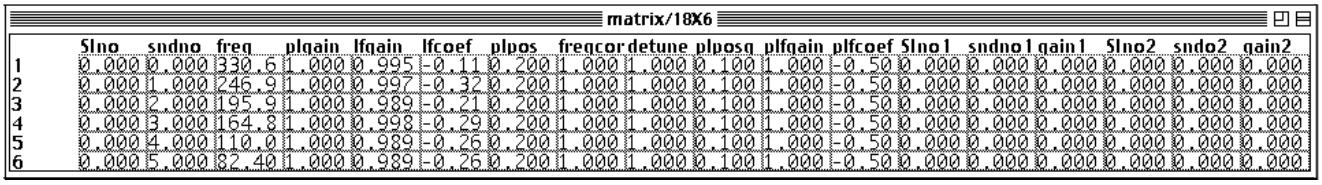
Figure 8. *Overview of the control and synthesis tools.*

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 to 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 simple scheme supports also imbedded definitions of C-structures (a C-structure can contain other C-structures to any depth). This feature is of primary importance when designing complex instrument models. For instance, the 'guitar1' box in Fig. 8—consisting of a coupling matrix and six dual-polarization string models—contains a C-structure defining the complete instrument. This top-level C-structure contains seven sub-C-structures, one for the sympathetic couplings and six for the strings. The most complex entity used in our example is the dual-polarization string model. It is a C-structure built out of nine sub-C-structures: two delay lines with 3rd order Lagrange interpolation, two loop-filters, three sample players, a delay and a timbre filter.

The interaction between ENP and PWSynth is realized in complex cases with the help of graphical matrixes (see Fig. 9). 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 protocol schemes (OSC [13]). Thus, if we assume that our current guitar synthesizer has the name 'guitar1', we can for instance 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 add easily to Fig. 8 new instrument boxes (such as 'guitar2', 'guitar3', etc.) each having a unique parameter name space.

The column to the left in Fig. 9 contains the string numbers (1-6) and the upper 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 (generated by an ENP input score) updates continuously the matrix items.

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'. Thus to refer to the excitation sample of the seventh fret of the first string we would use an index-pair '0 - 7' (we assume here that the excitation samples for the first string are found in the 'SI' having the index 0). The next parameter 'freq' gives the desired frequency of the current string. 'plgain' in turn gives the pluck gain. The next pair, 'lfgain' and 'lfcoef', control the behaviour of the loop-filter. 'plpos' gives the current pluck position, 'freqcor' is a frequency correction scaler 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' gives the gain of the Plucking-point filter (see Fig. 2). The pair 'plfgain' and


matrix/18X6

	slno	sndno	freq	plgain	lfgain	lfcoef	plpos	freqcor	detune	plposq	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	0.000	1.000	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	0.000	2.000	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	0.000	3.000	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	0.000	4.000	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	0.000	5.000	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

Figure 9. *Graphical parameter name matrix.*

'plfcoef' control the Timbre control filter. The final six parameters define the parameters of two extra sample-players (shown as 'Special effects' in Fig. 2) used to trigger some special playing effects.

6. CONCLUSIONS AND FUTURE PLANS

Newest developments in the model-based synthesis of the classical guitar were described, including signal processing methods for synthesizing realistic transients between notes, and pizzicato, and forte tones.

Examples of short phrases and excerpts from musical pieces that demonstrate the novel capabilities of our classical guitar synthesizer are available online at `www.acoustics.hut.fi/demo/dafx2000-synth/`.

Our future plans include further reduction of the excitation database size by removing the redundancies between the excitation signals. The parametrization of the low-frequency modes as suggested in [8] and [9] is also an attractive method to reduce the excitation database size.

7. ACKNOWLEDGMENTS

This research has been conducted within the projects "Sounding Score—Modelling 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 V. Välimäki has been financed by a postdoctoral research grant from the Academy of Finland. The authors are grateful to Mr. Mika Kuuskankare who helped in the programming related to the ENP system.

8. REFERENCES

[1] J. O. Smith, "Physical modeling using digital waveguides," *Computer Music J.*, vol. 16, mo. 4, pp. 74–91, 1992.

[2] J. O. Smith, "Efficient synthesis of stringed musical instruments," in *Proc. ICMC'93*, pp. 64–71, Tokyo, Japan, 1993.

[3] M. Karjalainen, V. Välimäki, and Z. Jánosy, "Towards high-quality sound synthesis of the guitar and string instruments," in *Proc. ICMC'93*, pp. 56–63, Tokyo, Japan, 1993.

[4] M. Karjalainen, V. Välimäki, and T. Tolonen, "Plucked-string models: from the Karplus–Strong algorithm to digital waveguides and beyond," *Computer Music J.*, vol. 22, no. 3, pp. 17–32, 1998. Reprint available online at the URL: `www.acoustics.hut.fi/publications/`.

[5] M. Laurson, *PATCHWORK: A Visual Programming Language and Some Musical Applications*. Doctoral dissertation, Sibelius Academy, Helsinki, Finland, 1996.

[6] M. Laurson, J. Hiipakka, C. Erkut, M. Karjalainen, V. Välimäki, and M. Kuuskankare, "From expressive notation to model-based sound synthesis: a case study of the acoustic guitar," in *Proc. ICMC'99*, pp. 1–4, Beijing, China, 1999.

[7] M. Kuuskankare and M. Laurson, "Expressive Notation Package (ENP), a tool for creating complex musical output," in *Proc. Les Journées d'Informatique Musicale*, pp. 49–56, 2000.

[8] V. Välimäki, J. Huopaniemi, M. Karjalainen, and Z. Jánosy, "Physical modeling of plucked string instruments with application to real-time sound synthesis," *J. Audio Eng. Soc.*, vol. 44, no. 5, pp. 331–353, 1996.

[9] T. Tolonen, *Model-based Analysis and Resynthesis of Acoustic Guitar Tones*. Report 46, Lab. Acoustics and Audio Signal Process., Helsinki Univ. Tech., Espoo, Finland, 1998. Available online at the following URL: `http://www.acoustics.hut.fi/publications/`.

[10] C. Erkut, V. Välimäki, M. Karjalainen, and M. Laurson, "Extraction of physical and expressive parameters for model-based sound synthesis of the classical guitar," presented at the *AES 108th Int. Convention*, preprint no. 5114, Paris, France, 2000.

[11] D. A. Jaffe and J. O. Smith, "Extensions of the Karplus–Strong plucked-string algorithm," *Computer Music J.,* vol. 7, no. 2, pp. 76–87, 1983.

[12] M. Laurson, "Real-time implementation and control of a classical guitar synthesizer in SuperCollider," in *Proc. ICMC'2000*, pp. 74–77, Berlin, Germany, 2000.

[13] M. Wright and A. Freed, "Open Sound Control: a new protocol for communicating with sound synthesizers," in *Proc. ICMC'97*, pp. 101–104, Thessaloniki, Greece, 1997.