Contents

List of Tables

List of Figures

Acknowledgements

This dissertation could not have been finished without the help and support of my supervisor, as well as of all the teaching staff of Acoustics at Salford University, Blend Noise Studio and my family.

First of all, I would like to thank Dr. Ian Drumm for his suggestions, encouragement and support during the entire project. Special thanks go to Dr. Olga Umnova, Dr. Mark Avis, Dr. Bill Davies, Prof. Yiu Lam, Dr. Francis Li and Dr. David Waddington for their support during the Audio Acoustics courses. I would like to express my sincere appreciation to Prof. Julius O. Smith, for both his long-life research and his books that furnished me with a wealth of knowledge otherwise impossible to find. My special appreciation goes also to Prof. Vesa Välimäki and to the Acoustics research staff at Aalto University, for their important suggestions and remarks. Thanks also to Prof. Balazs Bank, who provided me with technical support and help for this dissertation. I would like to thank my friends at Blend Noise Studio in Milan: Federico Bortoletto, Filippo Corbella, and Irene Pellecchia. Their help was crucial for the recording sessions and the testing procedures.

I also owe my gratitude to Prof. Francesco Braghin and Prof. Ferruccio Resta for sharing their knowledge during my last year at university in Milan. What I learned from them at that time was crucial for beginning this work. Thanks to Chiara who taught me that, in order to look for the truth, one must sacrifice something about himself. Finally, thanks to my friends Costanza Asnaghi for linguistics/lexicology consulting and Luciana Zelioli for printing this dissertation.

Non technical acknowledgements
Thanks to the extended study group for support, food, ospitality, coffee, sharing: Maria Lucia Costa, Monica Ciufoletti, Fabio Verduci, Anna Benelli, Irene Pellecchia, Costanza Asnaghi, Luciana Zelioli, Maria Barbieri, Carmen Soldo, Letizia Bizzozero, Giacomo Vianello, Nicoletta Barzaghi, Marzia Beacco, Carlo Coccia and all the others.
Special thanks to Moulinex and Nestle for their awesome products. I wish not to thank the social networks in general for their disturbance against any kind of productive activity.
Kudos to Max Mathews, Miller Puckette, Jean Baptiste Joseph Fourier, Jean Baptiste Le Rond d'Alembert and Philip Hansen Anselmo.

This dissertation is dedicated to my parents, my brother and my sister (Cristoforo, Maria B., Francesco and Elena). Their support, love and inspiration are the most important things to me. Without them, I could have accomplished nothing in my life.

Chapter 1

Introduction

1.1 Motivation

Physical Modelling is a particular kind of sound synthesis in which the physics of musical instruments is applied to sound generation. Instead of creating the sound directly with oscillators and filters, it aims to model physical processes that generate sound in the instruments[14].
The results of Physical Modelling are:

- Expressive and realistic sounds,
- No need for dedicated hardware or lots of memory to store samples,
- Possible real-time implementation of Physical Modelling on generic personal computers.

The elaboration of a physical model of a musical instrument is two-fold:

- To create interesting sounds that would not possibly be generated from an acoustic instrument;
- To recreate an imitation of real world instruments or variations upon the same instrument.

This project will develop the second motivation listen above, aiming to create a physical model for an existing acoustic guitar, in particular a flat top steel string guitar. The first element to be studied will be a single string. The string will be modeled with the aim of delay lines that represent a virtual waveguide. Two more elements will be implied in the building of the model, namely the guitar body and the interaction between the player and the instrument.

Beyond investigating physical modelling techniques applied to guitar, this dissertation aims to build the foundations for a guitar simulator that is real-time, MIDI compatible, parameter driven and easily expandable.

The calibration procedures will be performed with a flat top acoustic guitar as a reference to define a set of parameters inside the model. These parameters will also allow the user to change the instrument output, thus creating different versions of the synthesizer itself.

1.2 Outline

Chapter 2 presents a brief description of physics-based sound synthesis approaches to musical instruments and provides a general overview of the approach used to build the model.
Chapter 3 focuses on the design of the model and includes a detailed description of its different modules and parameters, from the string to the body filter.
Chapter 4 describes the calibration procedure for parameters estimation and the body impulse response measurement technique.
Chapter 5 presents the discussion of the realtime model, comparing the obtained results with the guitar samples used during the calibration procedure.
Finally, Chapter 6 summarizes the results of this thesis, outlining the limitations of the proposed model and its possible future developments.

Chapter 2

Background and Literature Review

'The guitar is a small orchestra'
Andres Segovia

2.1 Sound Synthesis and Physical Modelling

Sound Synthesis is the art of generating sound using electronic hardware, software, or a combination of the two. The purposes of Sound Synthesis are:

- To create interesting sound impossible to generate from an acoustic instrument;
- To recreate an imitation of a real world instrument or variations of the same instrument.

In particular, sound synthesis of digital signals produces a stream of discrete values in time (samples) that needs to be fed into a DAC to become an acoustic event (passing through an amplifier and a loudspeaker/headphone system).

Different techniques to obtain digital sound synthesis have been studied over the years, including:

- Additive synthesis;

- Subtractive synthesis;
- FM synthesis;
- Granular synthesis;
- Sampling;
- Vocal / Formant synthesis;
- Physical modelling.

This dissertation will focus on the Physical Modelling technique. The Physical modelling synthesis is a method in which a waveform is generated through a mathematical model composed of a set of equations. The mathematical model aims at generating an acoustic event similar to an existing physical source, for example a music instrument.

The resulting set of equations is usually derived by simple physics laws that apply to the specific source to be emulated. Wave propagation laws are one example of such equations.

The model also has parameters that control:

- the description of the physical parts of the instrument (materials, dimensions, structure);
- the description of the player interaction (dynamics, plucking style, position of the pick along the string etc.).

This method has two goals. The first goal is to create a simulation of an existing instrument. The process of writing equations simulating the physics of the system leads to a better understanding of the system itself. The second goal is to generate sounds or build instruments that would be impossible to generate or build otherwise, and to simulate a variation of an existing instrument. For example, a manufacturer might want to simulate the output of his piano by changing a physical detail of its model without the needing to build a instrument prototype.

2.2 Excitation and Resonance

The two fundamental blocks of physically modeled instruments are the exciter and the resonator. The exciter is the action generating a vibration, and the resonator is the response of the instrument. Considered from a signal processing point of view, the exciter is a signal fed into a filter (the instrument). In this analysis, the exciter is the input of the system that in the physical world can be the pick plucking a string or a stick hitting the drum skin. The exciter is usually a non-linear element, while the resonator can be considered linear under some circumstances.

The relationship between the exciter and the resonator is also important defining the model. If the two are decoupled, the exciter is an input to the resonator only, and the resonator cannot interact backward with the exciter as it happens in a real instrument (interaction between a stick and the drum skin). In other modeled systems, a feedback route goes from the resonator to the exciter simulating the two-way exchange of energy.

2.3 Formulations

The classic newtonian laws are the starting point for the formulation of the problem of physical modelling, as seen, for example, in the second law of motion:

$$F = ma \tag{2.1}$$

One of the basic tools that will be used to describe the wave propagation will be partial difference equations, for example the classic d'Alembert wave equation. The PDE will be translated into the digital domain (with discrete variables for space and time), thus becoming a difference equation. A difference equation is a formula for computing an output of a digital filter based on past and present input samples and past output samples.

Other representations include:

- State Space models (difference equation with a vector of state variables);

- Transfer Functions (difference equation of a linear time invariant system or filter);
- Impedance networks;
- Wave digital filters;
- Digital waveguides (a bidirectional delay line).

The string model will be based on digital waveguides and transfer functions.

2.4 The Digital Waveguide

A digital waveguide is a tool used for spatial propagation of waves built upon digital delay lines and filters. A waveguide is used to propagate and filter a sampled solution to the wave equation. The digital waveguide can be applied to the propagation of waves through air, in a string and via other elements.

The following equation, known as the d'Alembert's formulation of vibrating string and derived from Newton law of motion (2.1), can be used to build a string physical model from scratch:[1]

$$Ky'' = \varepsilon\ddot{y}, \tag{2.2}$$

where K is the string tension (N) and ε is the linear mass density (Kg/m). Defining c as:

$$c = \sqrt{K/\varepsilon}, \tag{2.3}$$

we obtain the ideal mono dimensional wave equation, where $y = y(t, x)$ is the string displacement in the y direction at time t and position x along

[1]From a mathematical point of view, D'Alembert and Bernoulli started two important areas of audio synthesis: spectral modelling and physical modelling. While Bernoulli described physical vibration as a superposition of simple harmonic components, D'Alembert proposed a solution to wave equation in terms of traveling waves. These two different approaches describe the same phenomena. Bernoulli's solution appeared in the form of the specific integral $w(x,t) = f(x)g(t)$. Infinite solutions of this kind exist, and their infinite sum is equal to D'Alembert's solution.

the string. c acquires the physical meaning of wave propagation speed. The classic wave equation can also be written in the following form:

$$\frac{\partial^2 y(t,x)}{\partial x^2} = \frac{1}{c^2}\frac{\partial^2 y(t,x)}{\partial t^2} \tag{2.4}$$

The D'Alembert's solution to this equation is a linear superposition of two functions:

$$y(t,x) = y_r(t - x/c) + y_l(t + x/c). \tag{2.5}$$

y_l and y_r represent traveling waves propagating in opposite direction with the same speed c. y_l is traveling in the direction of increasing x variable and y_r in the opposite direction.

The extremities of a finite string of length L are fixed, and the conditions at the two ends are that

$$\begin{aligned} y(t,0) &= 0, \\ y(t,L) &= 0. \end{aligned} \tag{2.6}$$

When the perturbation (y_l or y_r) reaches an extremities of the string, it is reflected back in the opposite direction. At this point, energy losses are not considered yet.

The equation is translated into the digital domain both in time and in space. This means that both time and space are sampled and they are no longer continuous functions but rather discrete sequences.

$$\begin{aligned} t_n &= nT, \\ x_m &= mX, \end{aligned} \tag{2.7}$$

where n and m are integer values used as index in the time and spatial vectors. T is the sampling period in seconds, and X is the distance travelled in a sample period:

$$X = cT. \tag{2.8}$$

The two vectors are in fact two arrays of discrete values of lengths N and

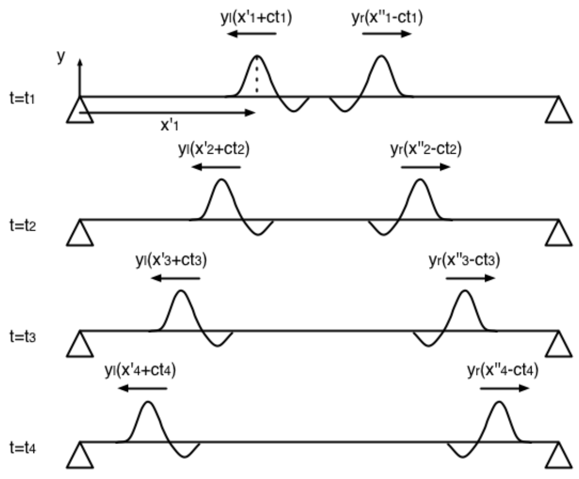


Figure 2.1: *Travelling waves: two solutions travelling at same speed but in opposite directions through the medium.*

M. The solution to wave equation in the digital domain is the following:

$$\begin{aligned} y &= y_r(t_n - x_m/c) + y_l(t_n + x_m/c), \\ y &= y_r(cnT - mX) + y_l(cnT + mX), \\ y &= y_r(cnT - mcT) + y_l(cnT + mcT), \\ y &= y_r(cT(n-m)) + y_l(cT(n+m)), \\ y &= y_r(n-m) + y_l(n+m). \end{aligned} \tag{2.9}$$

Substituting the digital vectors lead to see y as a function of the two indexes n and m. The wave propagation through the digital waveguide is then simulated by shifting values along the spatial array x as the digital time t is flowing.

The ideal string will then be modeled through a bidirectional digital waveguide, which in turn appears to be made up of two delay lines of length N/2 (N is defined as 2L/X). The upper delay line propagates the perturbation from left to right, the lower one from right to left. There is no loss of energy in this model so at both the end of the string the value of y, which is a displacement is reflected with a change of sign. This will be modeled with a multiplication of the signal for a gain of -1.

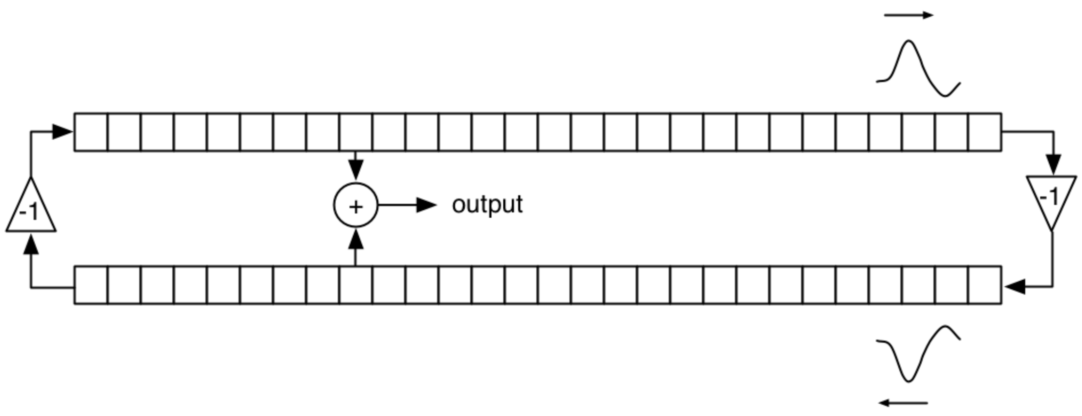


Figure 2.2: *Digital waveguide: the monodimensional digital wave as a sum of the values of two separate guides.*

Figure 2.3 represents the lossless digital waveguide described above:

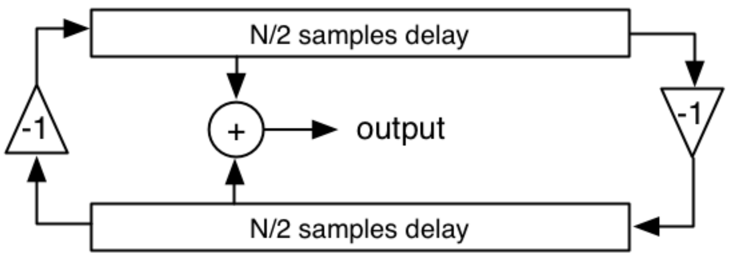


Figure 2.3: *Digital waveguide: digital delays as building blocks.*

In the following code, the dual waveguide is implemented in MATLAB with a displacement initial condition. Loss due to internal energy dissipation and loss at the bridge and nut are included in a lumped element, a filter on one side of the string. At every cycle, one side is now reflecting the displacement (the nut) while the other (the bridge) is dissipating energy from the wave (Figure 2.4):

```
fs=44100;                    % sampling frequency
```

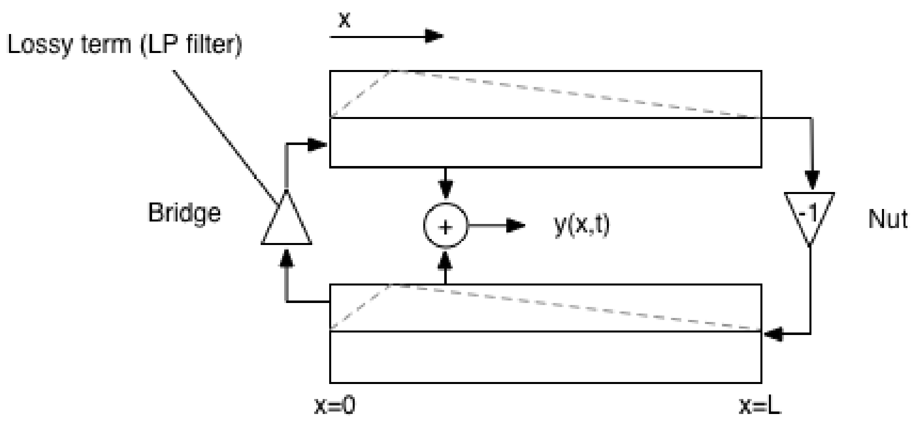


Figure 2.4: *Digital lossy waveguide with initial displacement: the two delay are initialized with a displacement at t=0 and transfer functions are added at the bridge and nut. The nut is simply reflecting the incoming function.*

```
T=1/fs;                  % sampling time
f1=329.6;                % string fundamental frequency
L=floor(fs/f1);          % approximated delay line length
b0=0.2;                  % coefficient b0 for LP 1 pole filter
a1=-0.8;                 % coefficient a1 for LP 1 pole filter

duration=5;              % duration in sec of synthetised sound

t=zeros(1,fs*duration); % pickup output vector init
x=linspace(1,100,L);     % spatial vector

y1=zeros(1,L);           % transverse motion vector y1
y2=zeros(1,L);           % transverse motion vector y2
pickup=20;               % pickup location along x axis
pluck=30;                % pluck location along x axis
A=0.5;                   % amplitude of pluck
filterstate1=0;          % one pole LP filter initial state

% this section produces the plucked shape for the string ...
    (initial
% conditions)
for i=1:pluck
```

```
    y1(i)=(A/pluck)*i;
    y2(i)=(A/pluck)*i;
end

for i=1:(L-pluck)
    y1(i+pluck)=(-A/(L-pluck))*i+A;
    y2(i+pluck)=(-A/(L-pluck))*i+A;
end

% buffers holding values at the bridge and at the nut
tmp=0;
tmp2=0;

% for loop computing the motion of the string
% lowpass filter 1 pole for the feedback
for n=1:(fs*duration)

    tmp=y2(1);
    tmp2=y1(L);
    x_n=filterstate1;                                          ...
           % input to allpass filter at previous step x(n-1) ...
       is the output of LP filter)

    filterstate1=b0*tmp2-(a1)*filterstate1;              % ...
       y(n)=b0*x(n)-a1*y(n-1) - LP filter

    y2 = [ y2(2:L) -filterstate1];
    y1 = [ -tmp y1(1:L-1)];

    t(n)=y1(pickup)+y2(pickup);

end

% this produces the audible output
sound(t,fs);
```

The initial displacement is divided equally on the two waveguides, and at each tick the equation solution is the sum of the two d'Alembert's traveling waves (y1 and y2 in the code above).

2.5 Karplus and Strong

In 1978, Alex Strong devised a simple algorithm that produced a realistic timbre of a plucked string.[2] The plucked string algorithm was presented in 1983 in the Computer Music Journal by Alex Strong and Kevin Karplus [9]. The algorithm was defined as follows:

- The pluck is modeled with a short excitation input, originally a white noise.
- The excitation is sent to output and to a feedback loop through a delay line of length L.
- The output of the delay line is sent to a first order low pass filter.
- The output of the low pass filter is mixed with the excitation to the output.

The algorithm scheme above is represented in Figure 2.5:

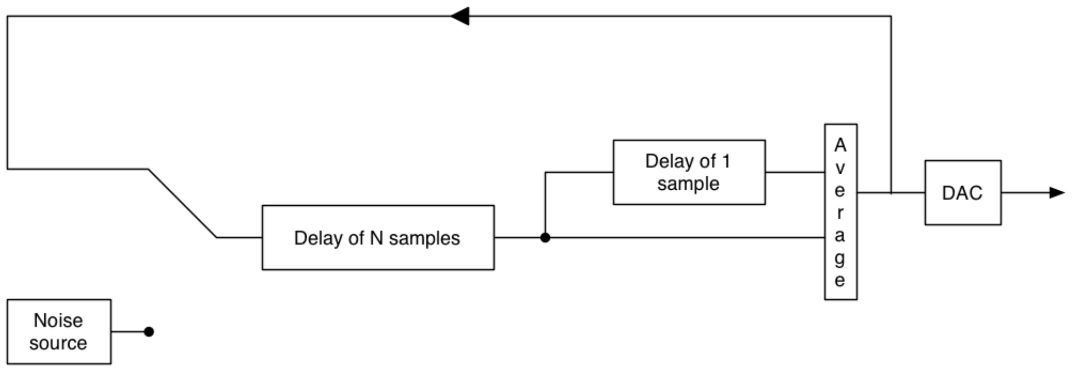


Figure 2.5: *Karplus-Strong algorithm.*

A simple code in Matlab for the Karplus-Strong algorithm is the following:

[2]In fact, Alex Strong, called this the Digitar algorithm, because it sounded like a guitar.

```
fs=44100;               % sampling frequency
T=1/fs;                     % sampling time

duration=5;                    % duration in sec of ...
   synthetised sound
t=zeros(1,fs*duration); % pickup output vector init

L=100;          % delay line length
g=0.99;         % low pass filter gain
a=-0.001;           % low pass coeffcient a
x=[1,zeros(1,length(t)-1)];

b1=[1 a];
a1=[1,a,zeros(1,97),(g+g*a)];
h=filter(b1,a1,x);

plot(h)
sound(h,fs)
```

In the code above, the delay line and low pass filter have been condensed into a single filter with coefficients defined by arrays a1 and b1. The input to the filter is an impulse, but can also be modeled by a random sequence of values between 0 and 1. The FILTER command in MATLAB computes the impulse response of the KS filter.

2.6 Beyond the Karplus-Strong Algorithm

In the same year, 1983, Julius O. Smith and David A. Jaffe published a study [10] in which they presented extensions of the original algorithm with applications to high-power computer equipment. The developments of the original Karplus-Strong algorithm were realized in response to musical needs that arose during the recording of David Jaffe's compositions, one of which was "Silicon Valley Breakdown" (1982). [3]

[3]David A Jaffe - Silicon Valley Breakdown - A film by George Olczak *https : //www.youtube.com/watch?v = _p4DGE5t3x0*. The recordings were done on a four channel tape. The music was entirely computer generated through the Samson Box, a

These extensions were practical insofar as they aimed to model better a plucked string in different areas: dynamics, tuning, plucking point and plucking direction. The original algorithm was exploited and changed as a true physical modelling tool to be used as a music instrument.

2.6.1 Extension of the Karplus-Strong Algorithm

Smith recognized that the Karplus-Strong algorithm was a lossy digital waveguide in physical terms. The low pass filter represented the loss of energy on each string vibration cycle, and it was frequency-dependent, producing a higher damping at higher frequencies, just as in the case of real string.

The difference between the Karplus-Strong algorithm and the previous model of plucked string (modeled with a bidirectional waveguide) was that the white noise burst had to be physically interpreted as the string was plucked with random displacement and velocity all along its length and not only in one point (as in the real world).

The digital waveguide was then chosen as a fundamental building block for its simplicity. In fact, its main element, the delay line, was realized as a circular buffer and was foreseen to have only the following three simple operations:

- write memory
- read memory
- advance pointer

Therefore, this building block was very inexpensive to implement on a hardware/software system in the early 1980s as the computational cost was low. The pseudocode for a software circular delay buffer would then be:

1. *initialize buffer to 0*

synthesizer system built by Peter Samson. The machine consisted in 256 generators and 128 modifiers (filters, amplitude modulators, random number generators etc.) and was controlled via a PDP computer. Featuring 64 Kwords of delay memory it could be used to build large delay lines. It also featured 4 D/A converters for a four-channel sound output.

2. *initialize pointer*
3. *read from memory cell pointed by pointer*
4. *write into memory cell pointed by pointer a new sample*
5. *increment pointer value by 1*
6. *if pointer >= delay line length then pointer=pointer-M*
7. *goto instruction 3*

Figure 2.6 represents a circular buffer or a delay line of length M samples.

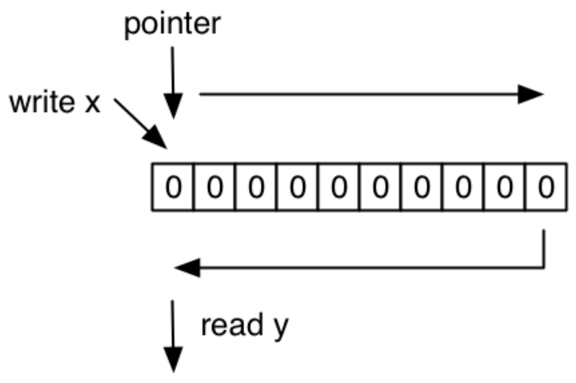


Figure 2.6: *Circular Buffer: three simple operations (write x, read y and advance pointer).*

With the first pass, the initialized buffer becomes filled with values (for example, audio samples), and, as the pointer reaches the end of the buffer, it is set back to point to the first element again. At this point, the old values are read and then replaced by new values. The first sample read at the second pass has been delayed by M samples with respect to the original sample.

The digital waveguide as exposed before is the central block of a string algorithm. A waveguide of length M will produce a sound with a pitch equal to:

$$f = f_s/M \tag{2.10}$$

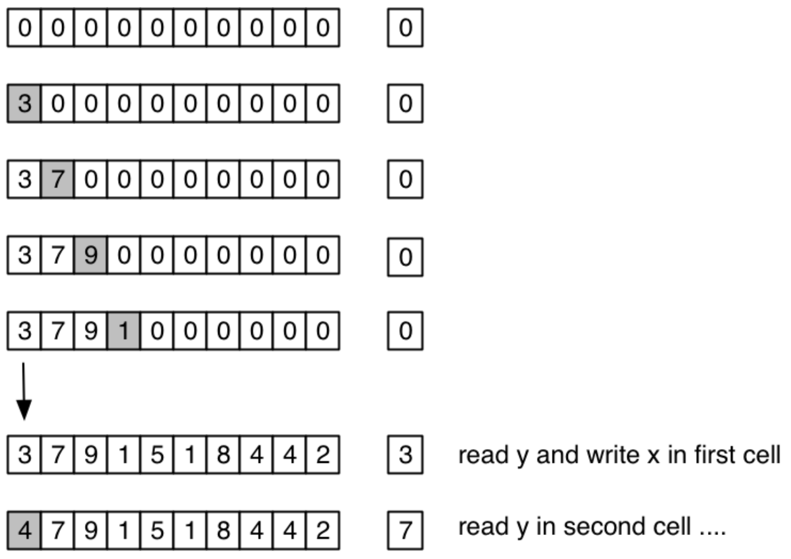


Figure 2.7: *Circular Buffer: the initialized memory locations are filled up from an outside source.*

Where f_s is the sampling frequency. In [10] Smith and Jaffe started looking at the physical phenomena of a guitar string and analyzed where the starting algorithm was lacking in capturing the details of the physical string behavior. All the corrections to the original Karplus-Strong idea were implemented via linear and time invariant filters. The observations or new ideas regarded the following:

- Designing a tuning filter motivated by the fact that the delay line length N is an integer causes tuning problems.
- Designing a better filter for emulating the decay differences between high and low pitches in an instrument.
- Controlling the spectral bandwidth of a string to emulate dynamics via a lowpass filter.
- Emulation of sympathetic vibrations between plucked strings and other open strings in an instrument.
- Emulation of pick position using a comb filter.

- Emulation of pick direction by lowpass filtering up picks and using standard noise burst for down picks.

Furthermore, the digital waveguide represented in Figure 2.3 is used in a new form under the EKS algorithm. Damping is now introduced, and it is consolidated with a scaling element or a filter. The pluck is defined by initial conditions inside the delay lines. The output is now a signal passing into a section of the delay line instead of being the sum of values coming from the bidirectional delay lines. With all the losses consolidated at a single point, it is also possible to commute the reflecting gain blocks to a single point and cancel them out. The two delay lines of N/2 samples are also combined into a single delay line of N samples. In Figure 2.8, it is possible to recognize part of the original KS algorithm which now acquire clearly the physical meaning of a bidirectional digital waveguide representing D'Alembert's wave equation in a new form.

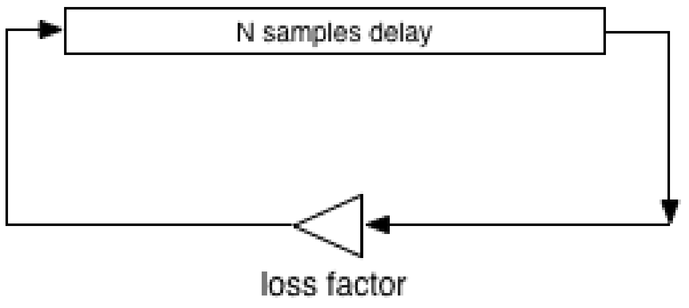


Figure 2.8: *Digital waveguide in the Extended Karplus-Strong algorithm.*

2.6.2 Body Modelling and Commuted Synthesis

In both guitars and pianos, the strings are coupled through a bridge to a resonating acoustic structure (body or soundboard) that is required for the transduction of string vibration to acoustic propagation in air [20]. Modelling the enclosure (or body) is a difficult task, at least if a computational model is desired.

With a signal processing point of view, the body can be seen as a filter of n_{th} order, where n is an integer usually around 500-1000. A single filter describes only 1 specific direction of sound radiation, therefore, different impulse responses are needed to build a directivity pattern.

Two paths can be taken to measure the transfer function from the string output to the listener (the system comprising the bridge and the body of the instrument). For acoustic guitars, the first method is to mute the strings, excite the bridge with a mechanical impulse (tuned hammer) and record the deriving sound. This obtained impulse response fully describes the transfer function for a single direction of radiation. The following is an impulse response of a classical guitar:

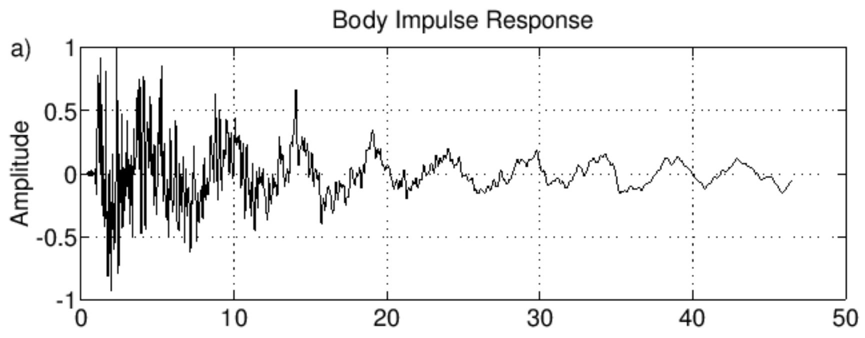


Figure 2.9: *Classical guitar impulse response.*

In previous works [21], it was determined that around 500 or more taps are needed to reconstruct a filter reproducing a satisfying sound. An FIR filter of such a high order is computationally too expensive for a real-time application.

A novel method was then formulated at the same time in two different research groups, CCRMA in Stanford and at the University of Technology of Helsinki. This method was called commuted synthesis. In this technique [11], since the string and the model are close to a linear time invariant system, the body can be commuted with the string in the signal chain:

In this way, though there is no need to design an FIR filter to model the body, the plucking signal is convolved with the resonator (body) impulse

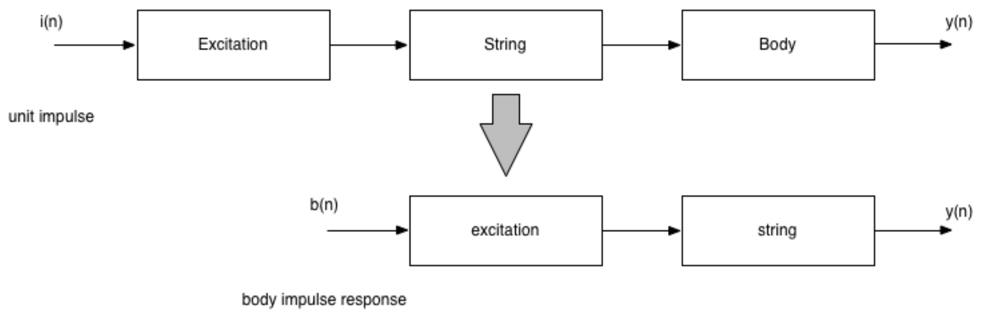


Figure 2.10: *Commuted synthesis.*

response to provide a single excitation signal. The signal is contained in a wavetable and becomes the input for the string model.

It is possible to follow two different methods in applying the concept of commuted synthesis:

- Obtain the body impulse response.
- Sample the guitar as played in different styles by a real player.

In the first case, a single wave file is obtained, representing the LTI body system that has to be convolved with an excitation signal. The body impulse response can be shortened if it is decomposed into the least-damped modes and the rest of the signal. The least-damped modes are identified in frequency and width (or Q factor). The modes are then eliminated via inverse filtering and reproduced via a filter section in cascade to the inverse filtered signal. In this way, the signal is shortened and the most important resonances of the body are now parametric, allowing for more control on the model.

In the second case, the desired range of notes from the guitar has to be recorded with a microphone. These recordings then have to be processed and in particular inverse filtered with the string model [21]. The resulting filtered samples (one for each note) become the excitation input for the model.

2.7 The Acoustic Guitar

The modern guitar is a string instrument of the chordophone family descendant of lute, vihuela, renaissance and baroque guitar. In particular, the acoustic guitar is a guitar using only acoustic means to transmit string energy to air thus producing sound. The string energy is typically transmitted via the bridge to a sound board and to the guitar body. The source is the string that can be plucked with fingers or with a plectrum. The vibration of the string alone cannot be efficiently transferred to air without a soundboard (the guitar top), which increases the vibrating area. The body of the guitar also adds a resonance effect and improves the efficiency of energy transmission at lower frequencies. The air cavity vibrations are coupled with surrounding air through the body hole placed on the guitar top or soundboard. The system (string, bridge, soundboard and body) then increase the coupling of energy from the pluck to air and color the sound with its particular transfer function.

The acoustic guitar family is divided into the main following types:

- Nylon stringed guitars
  - Renaissance guitar.
  - Baroque guitar.
  - Classical guitar.
  - Flamenco guitar.
- Steel stringed guitars
  - Folk guitar (also known as Steel String Acoustic guitar or Western guitar).
  - 12 string guitar.
  - Dobro guitar.
  - Archtop guitar.
  - Lap steel guitar.

This work will focus on building a physical model of the steel string acoustic guitar. The standard tuning for the folk guitar is defined in Table 2.1

Table 2.1: *Standard Tuning for Six String Guitar.*

String	Note	Fund. Frequency
1st	E2	82.4 Hz
2nd	A2	110.0 Hz
3rd	D3	146.8 Hz
4th	G3	196.0 Hz
5th	B3	246.9 Hz
6th	E4	329.6 Hz

The tuning defined in the above table is called ”standard tuning” because it is the most commonly used tuning scheme. Other tunings can be used, especially open tunings (open G, open D or drop D, the standard tuning with E2 dropped to D2). The most common type of Folk guitar is the Flat-top guitar, named after its flat top plate. The style of the Folk guitar is then defined by the body shape (overall size and proportions). The most common body styles are:

- Grand concert (Martin)
- Auditorium (Martin)
- Dreadnought (Martin)
- Jumbo (Gibson)

A Gibson Jumbo SJ-200 Standard, a steel string acoustic guitar, is pictured in Figure 2.11. Figure 2.12 shows the open body of a SJ-200 from the inside, revealing the particular structure under the soundboard. The braces are positioned in order to contribute to overall soundboard vibration and guitar sound. The bracing pattern is characteristic of an acoustic guitar type and the one used, for example, in a classic guitar differs from the one used in a steel string guitar.

Figure 2.11: *Gibson Jumbo SJ-200 Standard.*

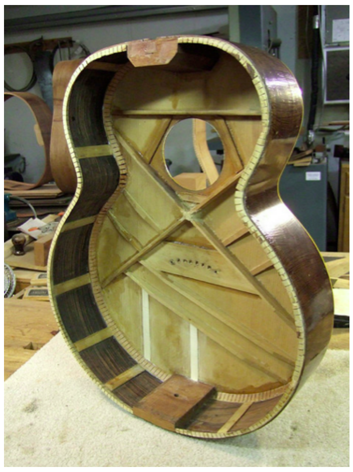


Figure 2.12: *Gibson Jumbo flat top backside view.*

The several interactions in the string-bridge-soundboard-body chain are not the only elements of complexity in modelling the instrument. Another building block is the excitation part (as introduced in Section 2.2) or the playing style. The plucking style has already been divided into finger plucking and plectrum plucking.

There are several ways in which the interaction plectrum/string can be modeled: one of these is to build the interaction in term of mechanical quantities with a lumped 1 d.o.f. model (spring and damping). Other complexities arise from the playing style, for example:

1. Vibrato. This particular style is produced by the player's left hand and is defined by pitch variation and a rate of pitch variation. The pulsating pitch variation is produced physically by moving the finger position on the fret in the direction perpendicular to the string (thus changing the string free length).

2. Glissando. After an initial pluck, the player's left hand slides along the fretboard from the starting position to another. This leads to a continous pitch variation from a starting value to an end value. Frets contribute a particular character to the sound (interruption or variation when the left hand move across one fret), dissimilar to the same glissando done on an instrument like the double bass or violin. Fretless instruments, like the acoustic bass can produce a continuous pitch variation.

3. Hammer on and Pull-off. After an initial pluck, the left hand of the player strikes the plucked string on another fret, giving a legato effect. If the second note is higher than the first note the effect is called hammer on; if the second note is lower than the first note, it is called pull-off (in this case the left hand finger pulls off from the fret while another finger is on a lower fret).

4. Tapping. All string excitations are played on the fretboard, mostly with fingers. In this case the excitation is more percussive, and the most extreme variation of this technique is an eight-finger tapping with both

hands striking notes on the fretboard. The string is pressed against the fret, and this collision is the excitation input to the string itself.

5. Strumming. This is a sweeping action done by a fingernail or a plectrum striking more strings simultaneously. A strumming pattern thus excites 6 strings at once, which are already in motion from the previous sweeping movement.

6. Angle of plectrum. Beyond the plectrum position along the string, two angles define the plectrum position in space, both influencing the excitation of the string.

Ideally, all these styles can be studied and analyzed with a physical modelling technique. In fact, one of the main advantages of physical modelling is that all of the performance information and parameters can be included in the model. The computational complexity of the model rises with the inclusion of performance parameters, but this is accepted as long as the model remains in real time, which is the main requirement for the model itself. If the model can process inputs and outputs in real time, it behaves like an instrument.

In the following sections, only the information regarding pitch, dynamics and plectrum parameters will be considered in the model for simplification.

2.8 Methodology

This section will analyze the methods and tools used to build the model of the acoustic guitar simulator.

2.8.1 String Model

The string model to be used will be a derivation of the Extended Karplus-Strong (or EKS) algorithm with a different method for excitation signal due to the particular body modelling technique. In a real string, there are two orthogonal planes of transverse vibration that are coupled to each other[20].

The Extended Karplus-Strong delay lines up to this point described only one of these planes; for this reason, the string model was refined by coupling the two planes of vibration (defined as horizontal and vertical in respect to guitar top plane). One can easily observe that the string vibration has different decay rates in two planes due to the particular shape of the top beneath the bridge. In fact the top plate is a thin plate, and its stiffness is different for displacements in the two considered planes. The stiffness in the vertical plane is lower, and the displacements due to vibration motion dissipate more energy coming from the string itself. It descends that the vertical plane vibration has a much faster decay when compared to that of the horizontal plane vibration. The pluck input is defined by its position along the string and by an angle formed by the pluck and the flat top. This angle can be used for decomposition of excitation signal into the two planes of vibration. Moreover the coupling of the two planes must be defined through a 2x2 matrix representing the coupling at the bridge.

In Figure 2.13, the bridge coupling is represented with four complex coefficients (varying with frequency) that compose the 2x2 coupling matrix below: H11,H12,H21 and H22. The minus sign denotes forces leaving the bridge, while the plus sign denotes the forces entering the bridge. 1 is the vertical plane, 2 is the horizontal plane.

$$\begin{bmatrix} F_1^-(z) \\ F_2^-(z) \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} F_1^+(z) \\ F_2^+(z). \end{bmatrix} \tag{2.11}$$

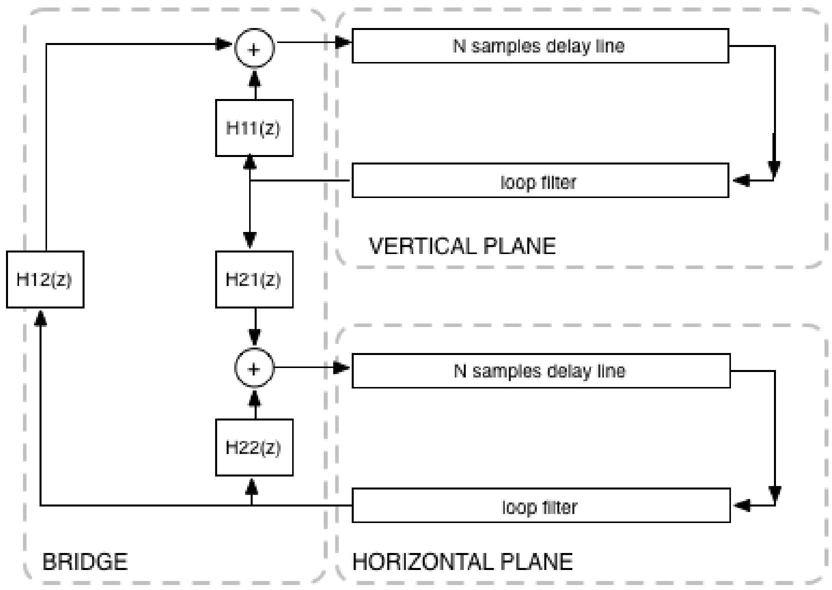


Figure 2.13: *String model with coupled vibration planes.*

2.8.2 Model Calibration

As introduced earlier in Chapter 2, the method chosen for modelling the body is the commuted synthesis method. For this purpose, a Gibson Jumbo SJ-200 standard (which is a flat top steel string acoustic guitar), was chosen as a reference for the model calibration. The model was used to obtain two types of information:

- Impulse Response
- Plucked guitar recordings

The impulse response was used to define the approximated linear and time invariant body (or resonator). The recordings consisted of plucking with a medium thickness pick each string along all the fretboard positions, thus obtaining 6x21 samples. Figure 2.14 provides the sampled range for each string:

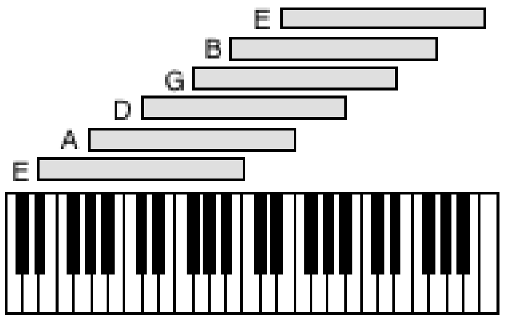


Figure 2.14: *Acoustic guitar strings range.*

The guitar was recorded with a single microphone (a Brauner Phantera) connected to an API pre-amp. The signal was fed into a DigiDesign A/D converter and recorded in ProTools 9 in *waveFORM audio file* format. All the samples were subsequently edited with the GPL software Audacity.

The guitar plucked samples have been used for damping estimation. The damping is, as modeled in the EKS algorithm, dependent on frequency. To measure the damping the procedure adopted in [22] is used via the STFT

(Short Time Fourier Transform). A sequence of FFT is computed on a recorded guitar tone (starting from the attack and ending after about 1.0 second). The signal is windowed and padded, for each frame magnitude spectrum is obtained and peaks are detected. The peaks are then collected in MATLAB and represent the harmonics of the guitar tone (the signal is quasi-periodic, the term partial is also used to describe the narrow-band energy content in a certain frequency). The decay rates of each harmonic or partial are used then to calculate filter gains thus defining the damping filter.

The filter used for simulating tone decay is a one-pole loop filter defined by the following transfer function in the *z-plane*:

$$H(z) = g\frac{a_1}{1 + a_1 z^{-1}}. \tag{2.12}$$

The filter needs for the two coefficients, g and a, to be determined. The procedure presented above is thus repeated for each recorded sample in order to obtain g and a values for each specific sample. The simulator loop filter coefficients g and a are obtained by evaluating a first order polynomial that best fits the experimental data:

$$\begin{aligned} g &= g_0 + g_1 i_{fret}, \\ a &= a_0 + a_1 i_{fret}, \end{aligned} \tag{2.13}$$

where i_{fret} is the fret index (from 0 to 20). Although the one pole is unable to model the complexity of tone decays, from a perception point of view it is able to generate an acceptable simulation. During the recording of the samples it was possible to detect that the decay range was quasi-linear from the longest decay at open string to the shortest one in the last fret position. Exceptions were found in specific fret position in which the plucking triggered particular body resonances making the decay longer than expected.

2.8.3 Building the Model in MAX/MSP

Max/MSP is a commercial software published by Cycling74. It was developed at IRCAM by Miller Puckette towards the mid 1980s for MIDI data control. With the addition of the MSP module in subsequent years audio signal controls were added. Thanks to its open architecture and object-oriented philosophy, Max/MSP allows for a large array of structures to be built. For example, Max has a standard delay line and the signal processing tools needed to build the string algorithm described in previous sections.

In this dissertation, MATLAB will be used mainly for calibration duties and MAX/MSP for building a realtime guitar synthesizer. The main blocks of the synthesizer are the excitation wavetable, which contains all the excitation samples convolved with guitar body impulse responses, and the string models. The first block accepts a midi message with pitch and velocity as an input. Pitch is used to select the sample while velocity is simply used as a reference for excitation gain. The output of the wavetable, the excitation signal is fed in both the string model planes of vibration with amplitudes depending on the plucking angle (their sum is equal to unity).

The two planes of vibration simply contain a string loop featuring a 1 pole lowpass filter for decay (with coefficients depending on the pitch midi data), the delay line of length L (also depending on midi pitch) and an all-pass filter for tuning correction. Also the pick angle will be used to decompose the input into the two planes of vibration. The output is mixed together with the sympathetic vibrations coming from the other open strings. The output y(n) is the synthesized audible output of the guitar synthesizer.

The model that will be developed in the next chapters is illustrated in the figure below:

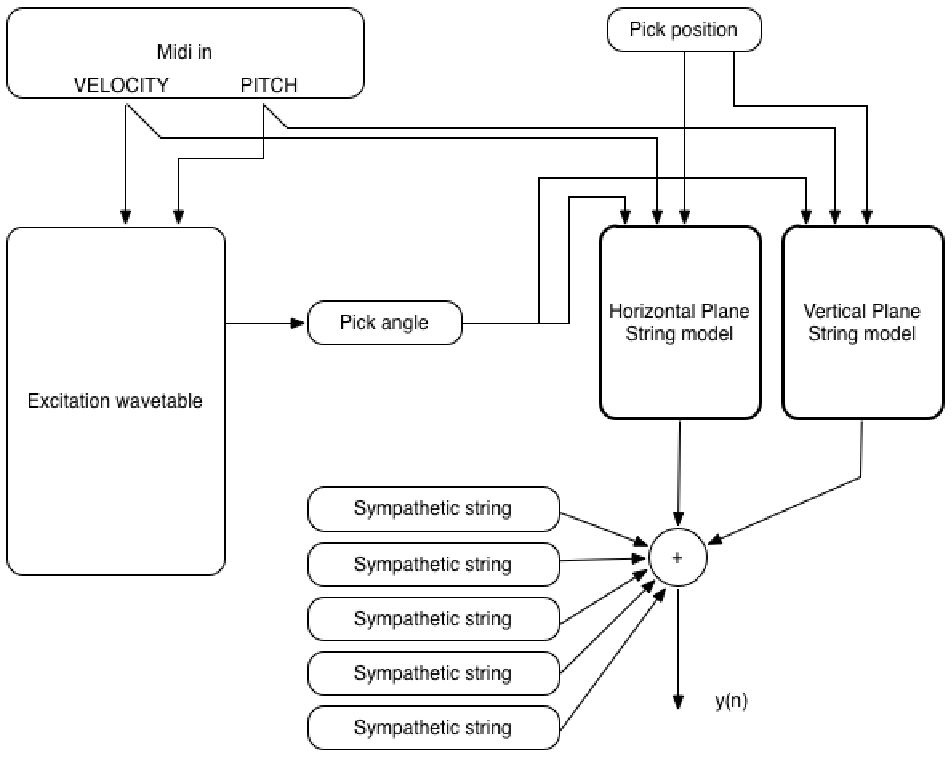


Figure 2.15: *Acoustic guitar string range.*

Chapter 3

Design

3.1 Introduction

In this chapter, the detailed design of the guitar synthesizer will be described in terms of its most important modules. This Chapter will also provide a detailed analysis of the link between the theory briefly exposed in Chapter 2 and the system structure as well as the definition of model parameters. The theory follows all the extensions foreseen in [10].

3.2 String

The first building block is the string in its formulation exposed in the previous chapter: the digital waveguide. The waveguide is defined by an input and an output and parameters. The first parameter is the length of delay line in samples which is tied to the desired fundamental frequency. This parameter is defined as[22]:

$$L = \frac{F_s}{f_0}, \tag{3.1}$$

where f_0 is the fundamental frequency of the string (the perceived frequency) and F_s is the sampling frequency. As with the Karplus-Strong algorithm, a lowpass filter is added to simulate the energy decay due to internal and external losses during vibration.

3.2.1 Delay Line

The delay line can be seen as a variable length circular buffer. One input is the excitation or x(t) input to the string block; the other input is the variable delay length defined in samples. The L-value derived from the equation above is an integer value, and the fraction is rounded to the closest integer value.

$$L = round(\frac{F_s}{f_0}). \tag{3.2}$$

3.2.2 Damping filter

The damping filter of choice is a lowpass with 1 pole defined by the following equation:

$$y_n = g(1+a)x_n - ay_{n-1}, \tag{3.3}$$

and the following transfer function:

$$H(z) = g\frac{1+a}{1+az^{-1}}. \tag{3.4}$$

The filter is defined by two coefficients g (gain) and a. These two coefficients will be defined in the calibration of the synthesizer exposed in Chapter 4. The plot contained in Figure 3.1 represents the frequency response for g=0.8 and a=-0.01.

The damping filter function emulates the loss of energy in each cycle due to internal and external losses of the string system. A single filter sums up the energy decay for string vibration, string friction with surrounding air and losses at the guitar bridge.

3.2.3 Tuning Filter

The fact that the delay line is fed with an integer value produces an approximation of the fundamental frequency as an output. This error increases with frequency as the delay line length is shorter. For a frequency of 440Hz, the value of L is 100.2273 truncated to 100, which leads to a quantized frequency

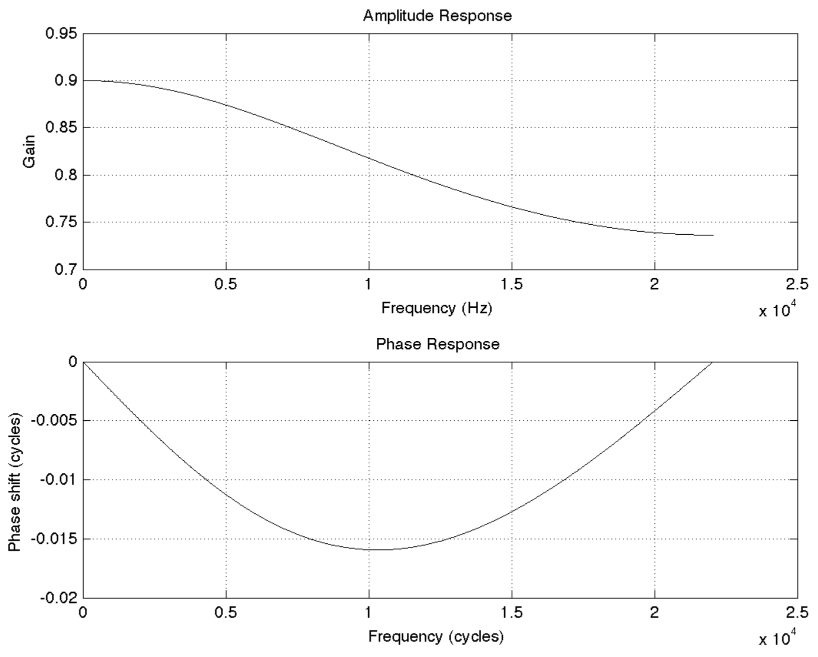


Figure 3.1: *One-pole lowpass filter frequency response.*

equal to 441Hz.

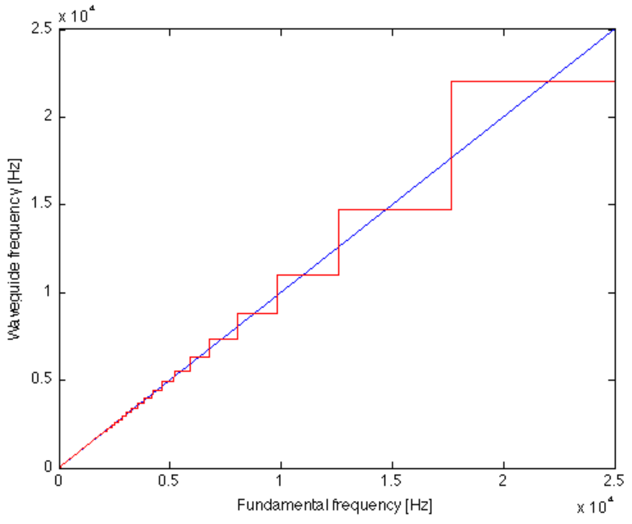


Figure 3.2: *Quantized frequency vs Fundamental frequency.*

Figure 3.2 shows the increasing quantization error with frequency.

The damping filter introduces a delay varying with frequency, due to its phase alteration of processed samples. The phase for the one-pole LP filter is:

$$\Theta(\omega) = -\tan^{-1}\left[\frac{-asin(\omega T)}{1+acos(\omega T)}\right]. \tag{3.5}$$

The phase delay in samples is then:

$$P(f) = -\frac{\Theta(\omega)}{\omega T}. \tag{3.6}$$

The fundamental frequency or the first partial of the waveguide is:

$$f_0 = -\frac{F_s}{L+P(f)}. \tag{3.7}$$

The value of L is derived from 3.2. f_0 is different from the value contained in Equation 3.1 and has to be tuned. The control range of the delay will be from 0 to 1 samples.

A solution to tune the waveguide correctly is to add an all-pass filter to the string loop which will not affect the frequency response in magnitude but will affect the phase response in order to obtain the following:

$$L + P(f) + P_{AP}(f) = \frac{F_s}{f_0}. \tag{3.8}$$

The difference equation for a first-order all-pass filter is

$$y(n) = Cx(n) + x(n-1) - Cy(n), \tag{3.9}$$

and the transfer function:

$$H(z) = \frac{C + z^{-1}}{1 + Cz^{-1}}. \tag{3.10}$$

In Equation 3.8, The phase delay produced by the all-pass filter $P_{AP}(f)$ is presented. $\frac{F_s}{f_0}$ is the exact value of the fundamental frequency and L is the integer value of the delay line obtained as:

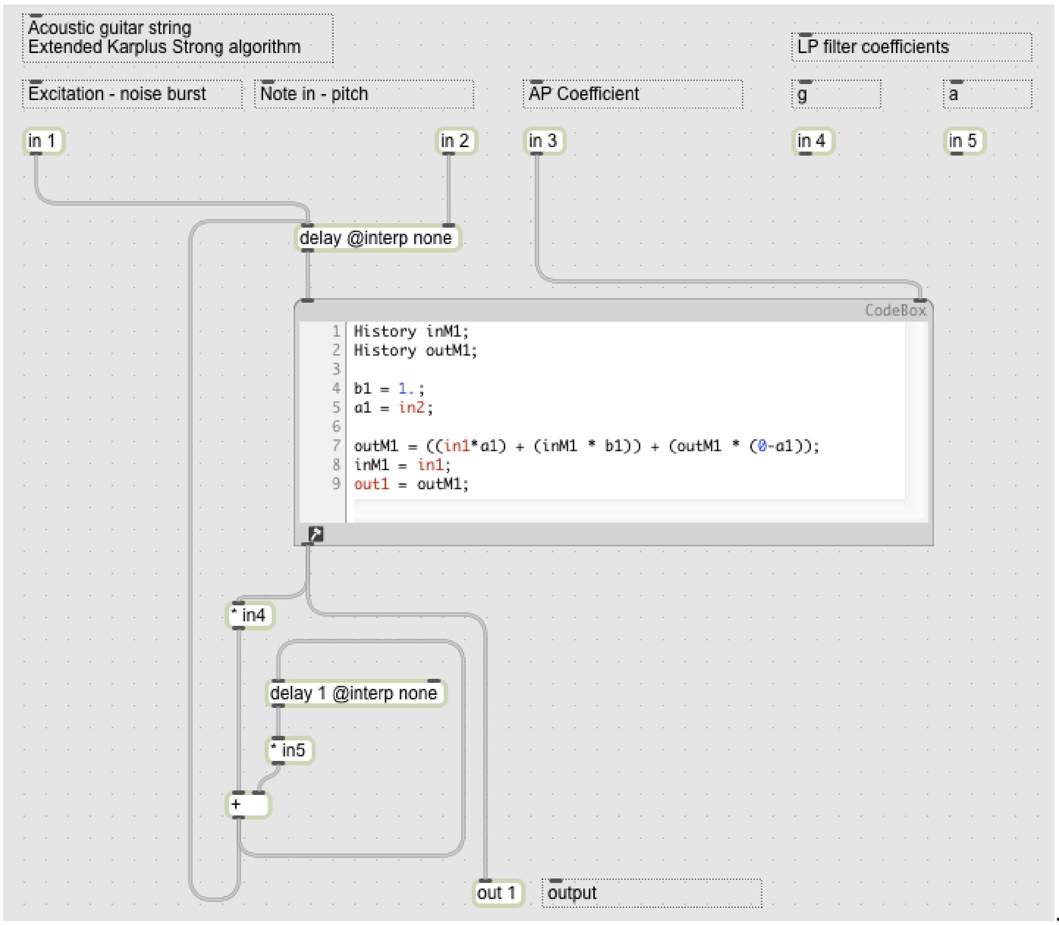
$$L = Floor(\frac{F_s}{f_0} - P(f) - \epsilon). \tag{3.11}$$

ϵ moves the control range from 0 to 1 sample towards ϵ to 1+ϵ. In fact, a desired value of delay equal to 0 means that $C = 1$ and $H(z) = 1$ due to pole-zero cancellation and possibly an unstable filter. Thus, for a certain fundamental frequency, L, $P(f_0)$ and $\frac{F_s}{f_0}$ will be computed and desired $P_{AP}(f_0)$ will be obtained from equation 3.11. In [19] the all-pass filter coefficient is obtained as a function of frequency f_0 and $P_{AP}(f_0)$ as:

$$C = \frac{sin(\frac{\omega T - \omega T P(f_0)}{2})}{sin(\frac{\omega T + \omega T P(f_0)}{2})}. \tag{3.12}$$

In the Max/MSP block $P_{AP}(f)$ will be computed from two sub patches and then fed into the EKS string block, represented in Figure 3.3

In Figure 3.3, one can recognize the delay line (delay function with delay line length in2 as an input), the all-pass filter section (written in the Max/MSP codebox) and the lowpass filter (the bottom section).



```
History inM1;
History outM1;

b1 = 1.;
a1 = in2;

outM1 = ((in1*a1) + (inM1 * b1)) + (outM1 * (0-a1));
inM1 = in1;
out1 = outM1;
```

Figure 3.3: *Digital waveguide in Max/MSP.*

Pin	Function	Description
in1	input	excitation / trigger
in2	input	note pitch in Hz
in3	input	AP filter coefficient C
in4	input	damping filter coeff g
in5	input	damping filter coeff a
out1	output	string filter signal output

Table 3.1: *KS String Loop.*

3.3 Dynamic Filter

The loudness of the digital waveguide output is a function of the amplitude of the initial excitation. Simply changing the amplitude of the output does not produce a timbral variation due to a change of dynamics. For example, two different notes have been recorded on the test guitar: the first played at lowest dynamic possible and the second near to the maximum dynamic. The energy decay relief distribution has been computed for both the recorded tones and showed in Figures 3.4 and 3.5: [1]

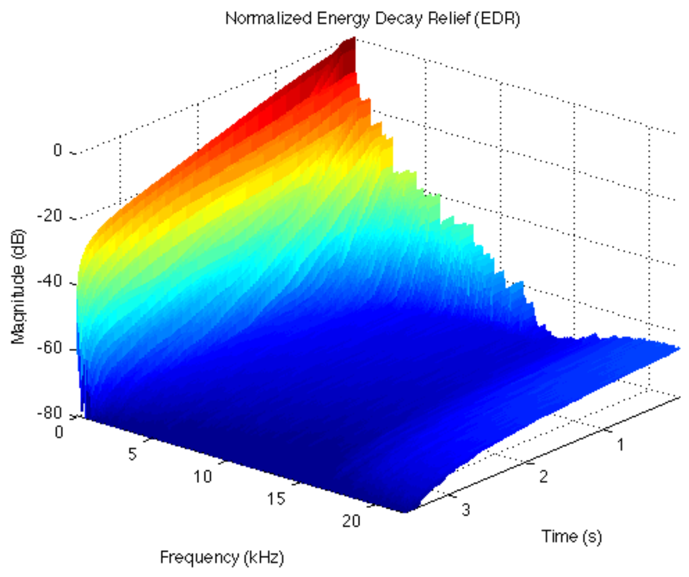


Figure 3.4: *Energy decay relief for lightly plucked string.*

The EDR plots show that the higher partials of the stronger pluck contain more energy than the lighter pluck. Therefore, apart from amplitude, a change of the spectrum content must be simulated as well. This will be done with a simple one-pole, low pass filter. The cutoff frequency or bandwidth will

[1] EDR is time-frequency distribution calculated via STFT (Short Time Fourier Transform). Its plot represents the variation of frequency spectrum through the time frames of STFT. The STFT is a FFT of a time frame of a signal, windowed with a particular function (for example the Hanning windows). The code used in this chapter has been derived from the code by Julius O. Smith MATLAB function available at https://ccrma.stanford.edu/~jos/vguitar/Using_Energy_Decay_Relief.html

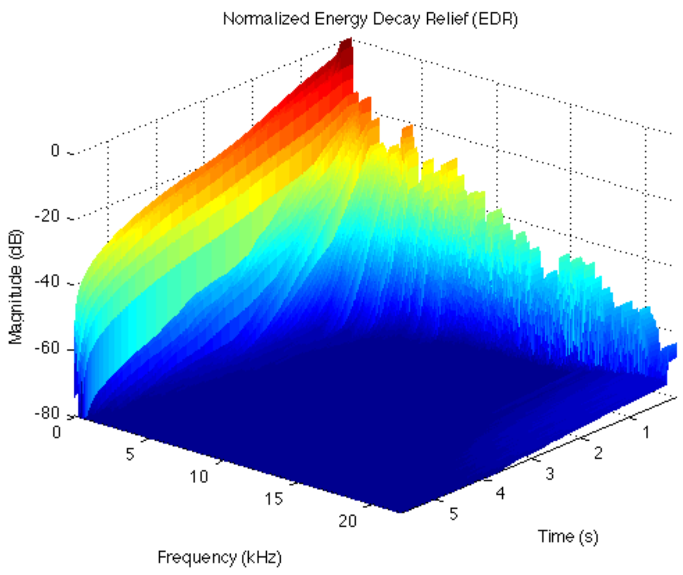


Figure 3.5: *Energy decay relief for hard plucked string.*

be proportional to the desired dynamic level, which in turn will be controlled via the MIDI velocity parameter.

The difference equation of the dynamic filter is the following:

$$y(n) = (1 - R)x(n) - Ry(n), \tag{3.13}$$

where R is the dynamic level limited between 0 and 1. When R goes towards unity, the filter produces a reduction in loudness and in spectral bandwidth. The dynamic level is defined between 0 and fs/2. In Max/MSP, the velocity level will be mapped to this quantity as in Figure 3.6 (velocity 7, dynamic level 1088 Hz)

It is not possible to use a fixed lowpass filter for all the synthesizer pitches at a desired dynamic level L. In this case, lower notes will be perceived louder than higher notes. The parameter R for the lowpass filter will be a function of both the pitch (fundamental frequency) and the dynamic level L.

The first step is then to design a lowpass filter with bandwidth L at a frequency f_m equal to:

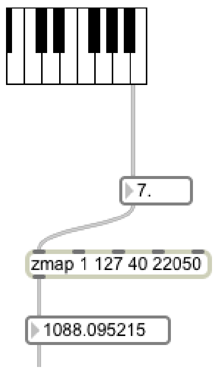


Figure 3.6: *Mapping velocity to dynamic level.*

$$f_m = \sqrt{f_l f_u}, \tag{3.14}$$

where f_l is the lower frequency bound and f_u is the upper frequency bound. Then we compute the gain G_L for this filter at frequency f_m.

$$H_d(z) = \frac{1 - R_L}{1 - R_L z^{-1}}, R_L = e^{-\pi LT}, G_L = \left| H_L(e^{j2\pi f_m T}) \right|. \tag{3.15}$$

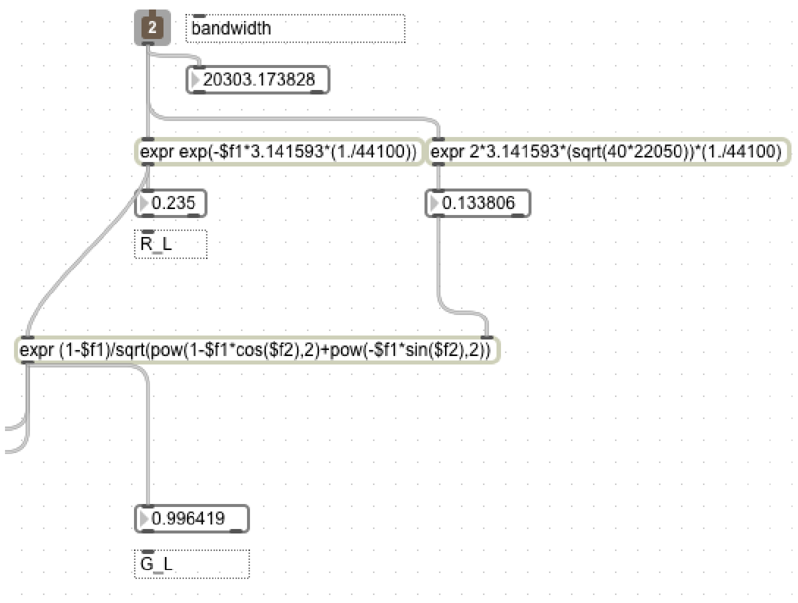
The gain for a desired fundamental frequency f is then computed and has to be equal to G_L:

$$G = \frac{1 - R}{|1 - Re^{j2\pi fT}|}. \tag{3.16}$$

Equation 3.16 is solved for R. Repeating this procedure for any value of f, a family of filters that has the same amplitude at the respective fundamental frequency is obtained.

The above procedure has been encapsulated in a patch called dynafilter in Max/MSP which computes the value of R and a single pole lowpass filter build as a Gen object.[2]

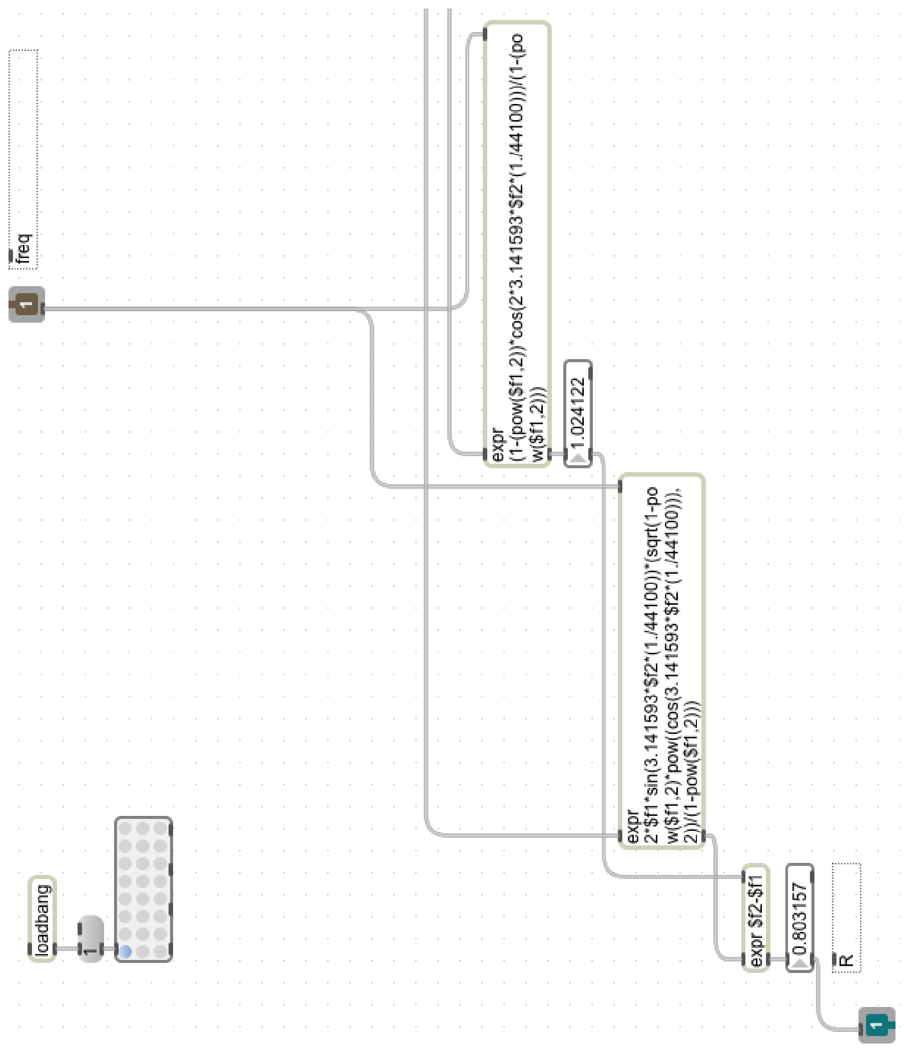
[2]A Gen is Max/MSP low-level language for audio processing. The performance of a Gen object is closer to compiled C code, and the code itself is valid on both Windows and Macintosh machines. Gen differently from MSP works at sample level processing one sample at a time



```
expr exp(-$f1*3.141593*(1./44100))
expr 2*3.141593*(sqrt(40*22050))*(1./44100)
expr (1-$f1)/sqrt(pow(1-$f1*cos($f2),2)+pow(-$f1*sin($f2),2))
```

Figure 3.7: *Dynafilter patch - first section.*

The value of R enters the low pass filter patch together with the output of the digital waveguide in Figure 3.9.



```
expr
(1-(pow($f1,2))*cos(2*3.141593*$f2*(1./44100)))/(1-(po
w($f1,2)))
```

```
expr
2*$f1*sin(3.141593*$f2*(1./44100))*(sqrt(1-po
w($f1,2)*pow((cos(3.141593*$f2*(1./44100))),
2))/(1-pow($f1,2)))
```

Figure 3.8: *Dynamic filter patch - second section.*

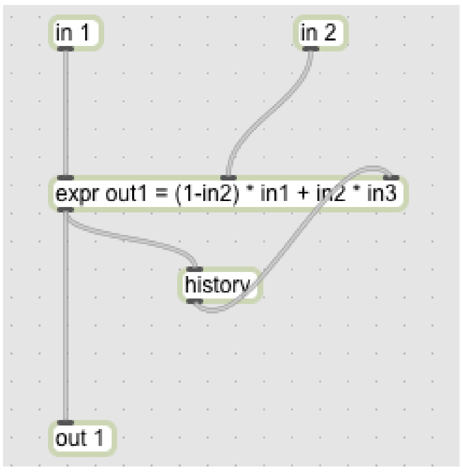


Figure 3.9: *Gen lowpass filter.*

3.4 Horizontal and Vertical Polarization

The KS algorithm described in the previous paragraphs represents a vibration occurring in a single plane. In particular, in the acoustic guitar we found that it is effective to model the transverse vibration onto two orthogonal planes, as the string vibrates both in transverse and in longitudinal directions. The latter can be neglected as its contribution is not important in acoustic guitar strings. One of the two planes of transverse motion is the horizontal plane parallel to the guitar body top plate. The other plane of transverse motion will be referred to as the vertical plane. The two planes are coupled ideally at the bridge where the coupling occurs, considering the string end point at the nut perfectly rigid for simplicity. The coupling is done building an admittance matrix, using experimental measurements. The bridge is a 2x2 matrix, and each element is a filter, if the velocity variable is used in the waveguide.

Here the string planes will be considered decoupled. They are triggered by the same input but oscillate independently one from the other. In the Max/MSP model, the signal will be divided in the two waveguides and their outputs will be summed again to obtain the final displacement vector.

The difference between the two waveguides is due to the decay rate. The horizontal motion will decay over a longer timespan than the other will because the bridge is more rigid in its direction than it is in the vertical plane. A way to simulate this behavior is to use different damping parameters for the two planes. The damping parameters are linked to the energy loss due to the bridge motion for every vibration cycle. Also, the string plucked in the vertical plane will produce a stronger tone than does in the horizontal plane. A guitar tone will then have a compound decay rate or will show the two-stage decay. In the first part, the stronger vertical motion will be predominant but will decay faster leaving only the horizontal motion with a slower decay (Figure 3.10).

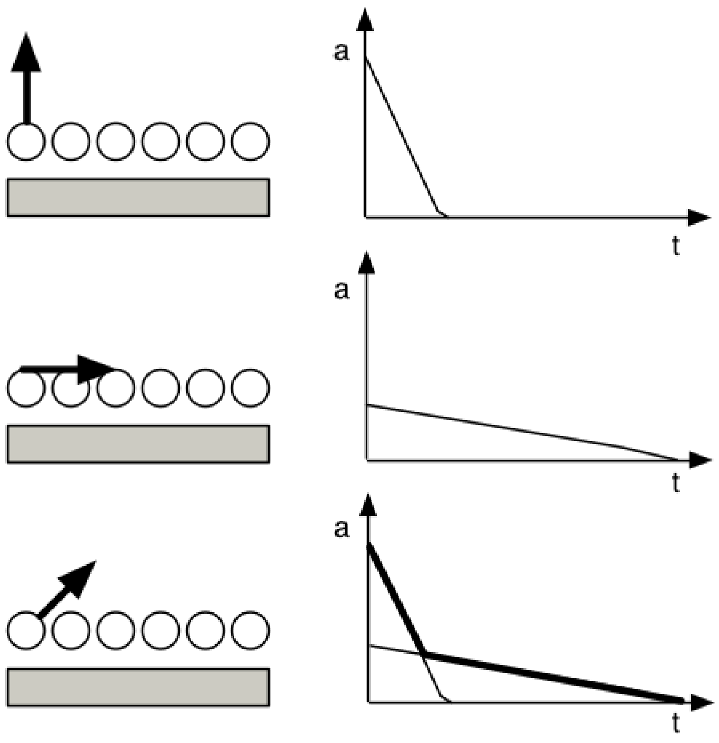


Figure 3.10: *Two-stage decay: the resulting amplitude decay is a compound decay of vibration in the horizontal and vertical planes*

3.5 Excitation

The excitation in the original KS algorithm was modeled with a noise burst that physically matched a random excitation along the entire string. Different choices are possible for modelling the excitation for the guitar synthesizer, and these are strongly connected to the body modelling.

- Noise burst triggered from an external message (ex.: MIDI note on message)
- Commuted synthesis: the excitation and the body resonator are aggregated in a single wave sample. This sample corresponds to the guitar body impulse response recorded after hitting the body top plate with a tuned hammer.
- Commuted synthesis: the excitation and the body resonator are aggregated in a single wave sample. The sample corresponds to the recording of a played note on the real guitar inverse filtered with the string patch. Thus, for each note, the system triggers its matching sample that is fed into the string model.

In each case, the system gets the MIDI note on message as a trigger event. The note on message is a list of two-integer values: pitch and velocity. When the external controller sends a note off message the list if formed by the same pitch and a value of velocity equal to 0.

3.6 Body

With respect to what is exposed in the excitation paragraph, the body can be modeled with:

- FIR Filter with n taps;
- Body impulse response (commuted synthesis) injected into the string model;
- Body impulse response (commuted synthesis) factored into main resonant modes and residual impulse response;
- A library of inverse filtered guitar samples (commuted synthesis) injected into the string model.

The first solution is computationally expensive, resulting in a very large filter (for example N=500). For the second choice, the body impulse response is simply triggered into the string model. As an example, Figure 3.11 represents a test impulse response recorded from an acoustic guitar. The impulse response is a *.wav* file sent into the string patch input together with the excitation signal. This solution is not parametric, and the body response cannot be controlled in any way.

The third option is to look at impulse response and filter out the lowest modes with an inverse filtering procedure in MATLAB. The lowest modes are also the least damped modes, and the first mode is the Helmholtz air resonance.

Applying the factoring returns a residual impulse response that is now shorter in time and closer to a noise burst. The extracted modes have to be reintroduced via parametric filtering after the string block. In this case, the resonance value and width can be controlled since the IR is parametric. This gives the user the ability to change the instrument timbre.

In [20], Smith proposes a sample procedure coded in MATLAB used to perform body impulse response factoring by creating an inverse filter using the estimated peak frequency in Hz and estimate bandwidth in Hz. The command filter applied to the IR produces the residual response. In this case, the code is applied to filter out the two lowest peaks of the spectrum (107.66 and 204.56 Hz). The function *ffg* is contained in Appendix B3.

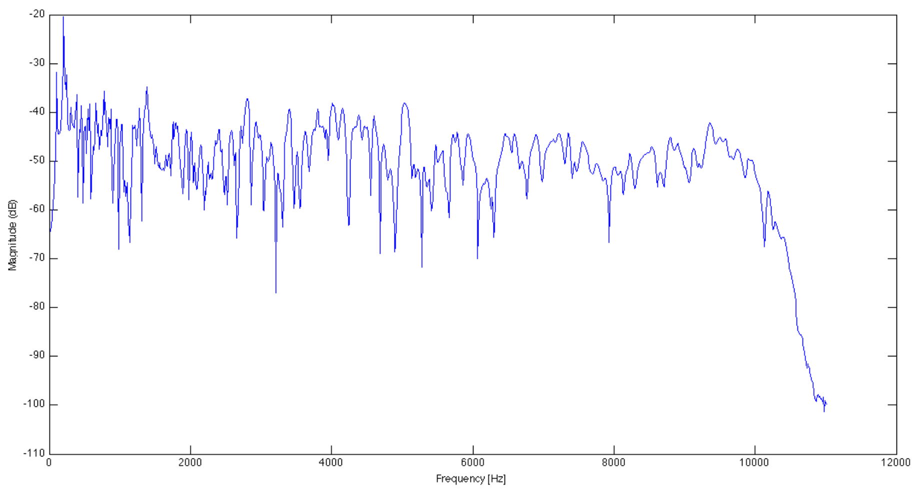


Figure 3.11: *Body IR frequency spectrum - magnitude.*

```
% read ir
[Y,FS,NBITS]=wavread('gtrbody.wav');

% fft of ir
dt=1/FS;
[mod,fase,freq] = ffg(Y,2048,dt);

[pks,locs] = findpeaks(mod);

% first resonance

freq1 = 107.6660;                    % estimated peak ...
    frequency in Hz
bw   = 5;                                % peak bandwidth ...
    estimate in Hz
```

```
R = exp( − pi * bw / FS);                   % pole radius
z = R * exp(1i * 2 * pi * freq1 / FS);      % pole itself
B1 = [1, −(z + conj(z)), z* conj(z)];       % numerator
r = 0.9;                                            % ...
    zero/pole factor (notch isolation)
A1 = B1 .* (r .^ [0 : length(B1)−1]);       % denominator

residual1 = filter(B1,A1,Y);                % apply inverse ...
    filter

% first resonance

freq1 = 204.5654;                   % estimated peak ...
    frequency in Hz
bw   = 5;                               % peak bandwidth ...
    estimate in Hz

R = exp( − pi * bw / FS);                   % pole radius
z = R * exp(1i * 2 * pi * freq1 / FS);      % pole itself
B2 = [1, −(z + conj(z)), z* conj(z)];       % numerator
r = 0.9;                                            % ...
    zero/pole factor (notch isolation)
A2 = B2 .* (r .^ [0 : length(B2)−1]);       % denominator

residual2 = filter(B2,A2,residual1);    % apply inverse filter

% fft of final ir
[mod2,fase2,freq] = ffg(residual2,2048,dt);


% plot original ir against residual ir
figure(1)
plot(freq(1:300),20*log10(mod(1:300)))
hold
plot(freq(1:300),20*log10(mod2(1:300)),'r')
```

Without commuted synthesis technique, the body impulse response should be modeled with a large body filter (the first option at the beginning of this section) that would be computationally expensive for a realtime application.

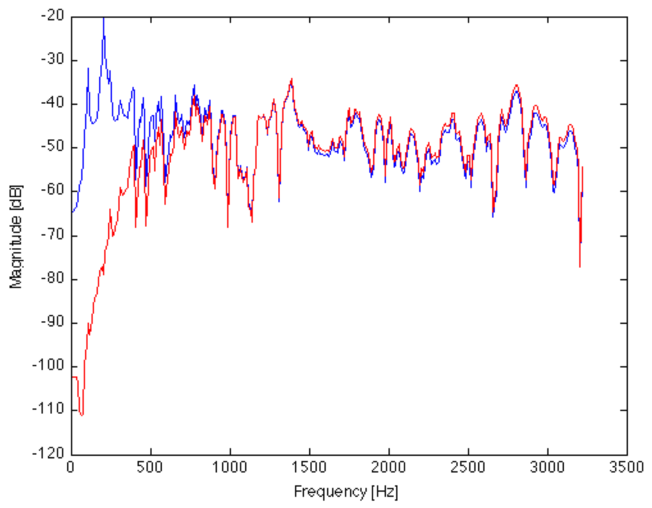


Figure 3.12: *Body IR frequency spectrum after inverse filtering.*

It has to be remembered that the hypotheses under which the commuted synthesis is used are the linearity and time-invariant properties of both the body and the string. While being non-linear, both body and string are considered close enough to a linear behavior for the application of this particular technique. In the next chapter, the body impulse response will be recorded in an experimental set-up as part of the synthesizer calibration.

The first resonance is the Helmholtz frequency. The body cavity and the top plate hole are the element of an Helmholtz resonator, which behaves like a lumped acoustic system under certain conditions. The 1 d.o.f. system is defined by the following differential equation:

$$m\frac{d^2\xi}{dt^2} + R\frac{d\xi}{dt} + s\xi = SPe^{j\omega t}, \tag{3.17}$$

m represents the air mass, R the term due to losses, s the stiffness and the right term the driving force produced by the pressure wave due to the flat top vibration at the bridge.

The body impulse response is loaded into Max/MSP in a memory cell with the buffer command and is then triggered with the groove command generating a signal into the EKS string loop.

3.7 Midi Dispatcher

The guitar synthesizer modeled in Max/MSP has been designed to fetch data from a MIDI device in terms of note pitch and velocity messages. Each note on message is analyzed and sent to a poly object that can manage up to six concurrent voices.

Each string is modeled by an instance of the string module representing one particular physical string in the acoustic guitar. The midi dispatcher module routes the midi message to the relevant string instance in order to generate sound. The strings are also limited in range matching the physical range of the guitar fretboard. Thus, one string patch is limited to play thus only one note at a time. The MIDI ranges for each string are show in Table3.2.

Table 3.2: *String MIDI Ranges.*

String	Open String Note	Lower Range	Upper Range
1st	E2	40	60
2nd	A2	45	65
3rd	D3	50	70
4th	G3	55	75
5th	B3	59	79
6th	E4	64	84

The MIDI message list composed of pitch and velocity is analyzed by a sub patch which detects the string compatible to the pitch and generates a list in ascending order. For a pitch equal to 52, the list will be '1 2 3'. This means that string 1,2,3 can play that pitch value. The lowest string available in the list (available means not busy) becomes the route destination for the MIDI message. If no strings are available in the list, then no output is produced by the synthesizer.

3.8 Sympathetic Strings

It is widely known that a plucked string can excite other open strings at certain pitch values. The indirectly excited open strings are defined as sympathetic strings. From a physical perspective, the sympathetic vibration happens when the i_{th} partial of the plucked string is equal to the j_{th} partial of another open string. To simulate this behavior in a simple, effective way, the output of each string block is fed into a six-by-six mixer matrix. The mixer has six inputs and six outputs and each element has a gain factor. The gain factor is a scaling factor (from 0 to 1) in order to keep the system stable, thus avoiding overflow.

The six mixer outputs are then mixed together with the six strings in the final synthesizer section prior to being fed to audio output.

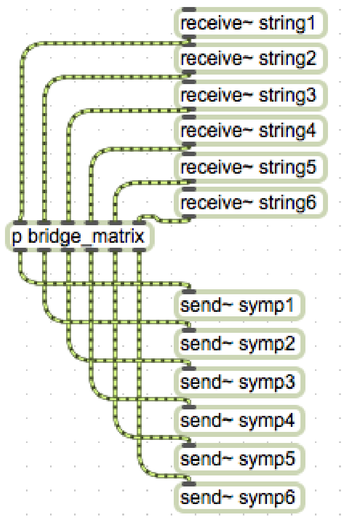


Figure 3.13: *Matrix subpatch - 6 x 6 mixer.*

3.9 Pluck Angle

As mentioned in the previous chapter, the plucking excitation beyond its position along the string, is also determined by an angle formed by the pluck itself and the flat top. This angle can be used for the decomposition of the

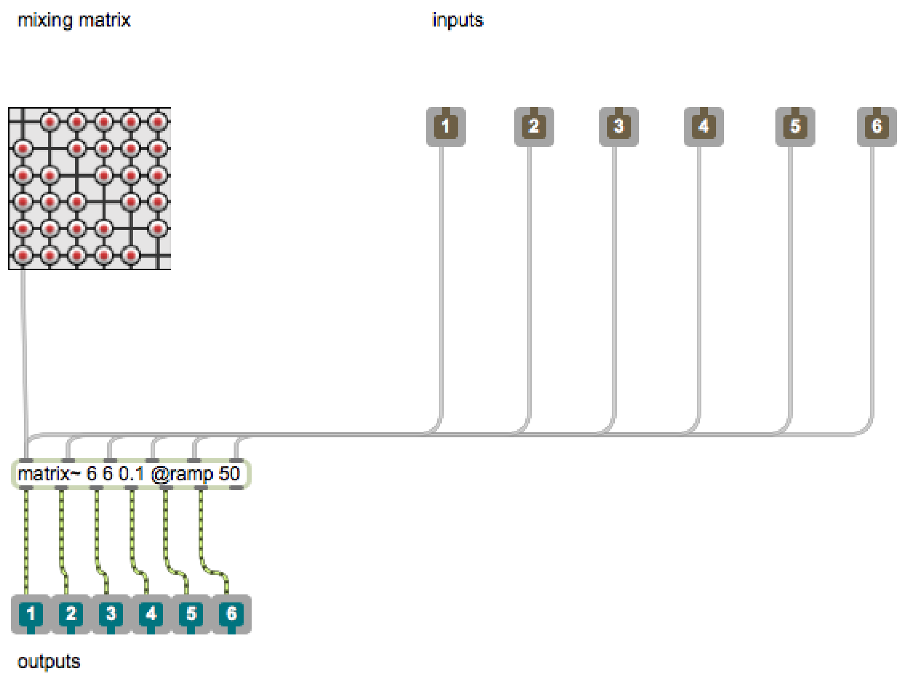


Figure 3.14: *Matrix subpatch - 6 x 6 mixer - detail.*

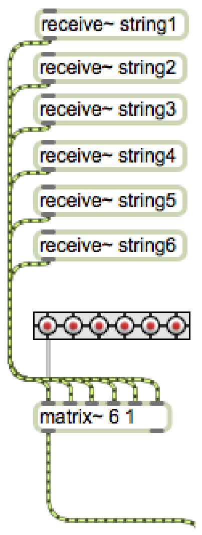


Figure 3.15: *Main mixer - mixing plucked string output to L and R output.*

excitation signal into the two planes of vibration H and V. The horizontal plane is parallel to the flat top, and the vertical plane is perpendicular to it. Usually, a player plucks the string with the pick almost parallel to the vertical plane. In this way, the string moves parallel to the fretboard. Instead, if the string is plucked in the vertical direction, the string is likely to touch the frets, thus giving birth to non-linear behavior. The model presented here simply defines a variable for the angle and uses it to decompose the input signal into two components that will feed, respectively, the horizontal plane waveguide and the vertical plane waveguide.

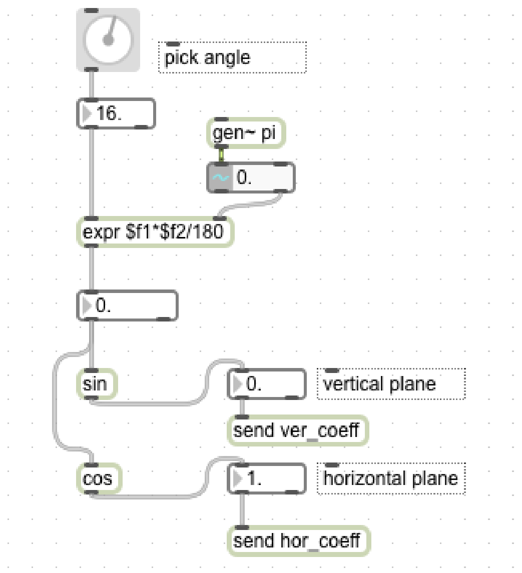


Figure 3.16: *Pluck Angle.*

3.10 Pluck Position

The pluck position creates a slight timbre variation in a real guitar. Usually, the player moves the striking point between the bridge up to the body end, near the last frets. First of all, a floating point variable is defined to indicate the pluck position. Since the pluck defines the boundary condition of the sting dynamic system, its value will influence how the string is excited. There is a relationship between the initial displacement and the subsequent string partials. If a string is plucked in its middle point, all the even harmonics (f=2F, 4F, etc.) will be suppressed as the midpoint is a node for those harmonics.

A feedforward comb filter has the following difference equation:

$$y(n) = x(n) - gx(n - M), \tag{3.18}$$

where g is the gain of the comb filter. For g=0.9 and M=10, the magnitude and phase response of the filter is plotted in Figure 3.17.

For M=10, a magnitude suppression will occur at values of sampling frequency divided by 10. The difference equation of the comb filter is modified into the following form:

$$y(n) = x(n) - gx(n - aL), \tag{3.19}$$

where L is the length of digital waveguide and a is the pluck position expressed as the string fraction between the bridge and the pluck point. If the string is plucked in the middle, then a=0.5. For a fundamental frequency of 880Hz and a equal to 0.5 Figure 3.18 reports the magnitude/phase plot.

In Max/MSP, a control for the pluck position is defined using a slider object sending the position variable a into the comb filter block.

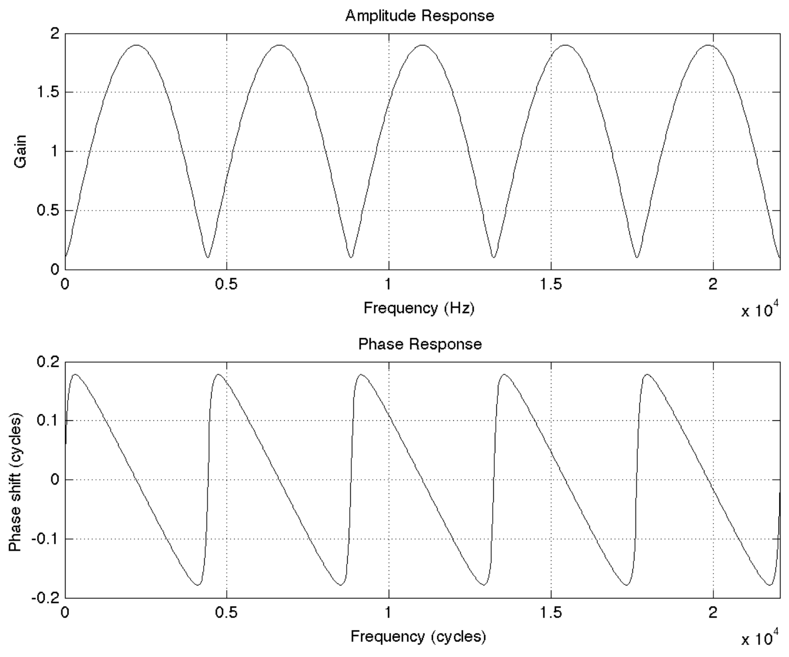


Figure 3.17: *Feedforward comb filter: g 0.9 M 10.*

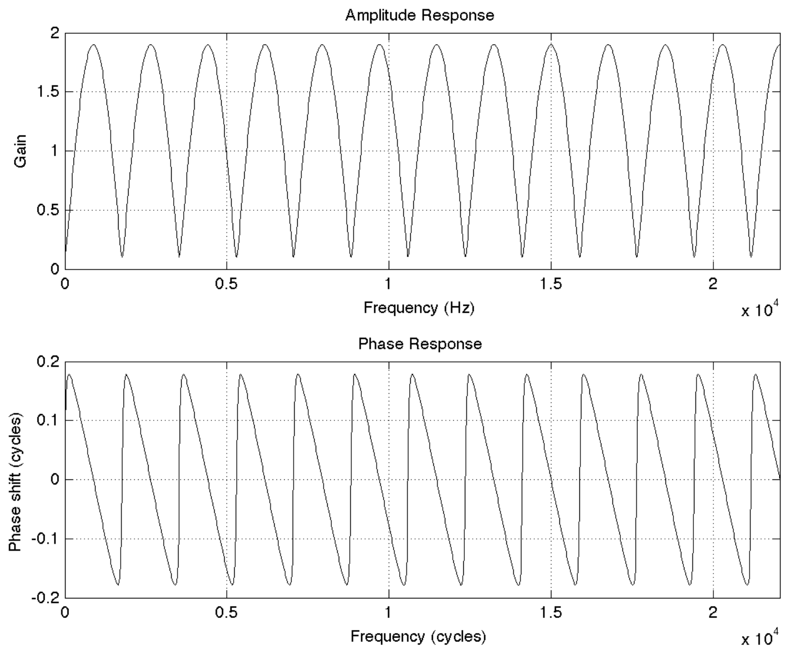


Figure 3.18: *Feedforward comb filter g 0.9 a 0.5 L 200.*

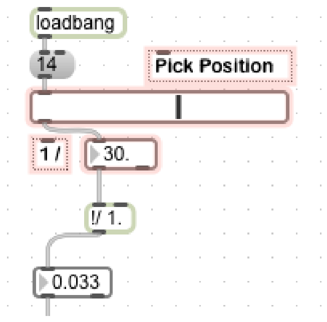


Figure 3.19: *Pluck position control.*

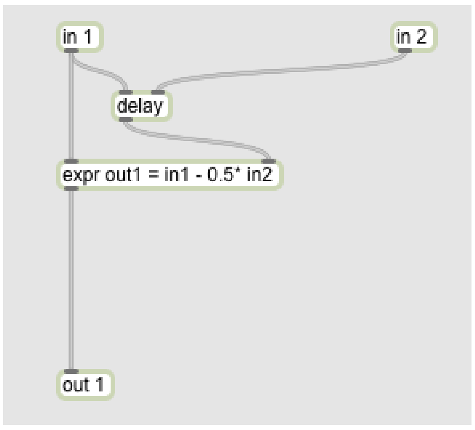


Figure 3.20: *Comb filter patch.*

Chapter 4

Model Calibration and Results

4.1 Damping Filter Calibration

This section will detail the calibration of string loop filtering. The loop filter or damping filter is the element that simulates all the energy losses for the string system. The damping filter is defined by two real coefficients: g and a. Every string was modelled to have motion in both the horizontal and vertical planes, thus resulting in two filters per string and four coefficients (g1, g2, a1, a2).

The string vibrations decay exponentially and the two-stage decay can be observed. These phenomena are due to string polarization as the decay rates of the two polarizations are slightly unequal.

It can be observed that:

- The horizontal decay will be faster and disappear around the first second of displacement.
- The vertical decay will be predominant after the horizontal motion disappear and will have a much more prolonged sustain.

In order to accomplish the calibration, g1 and g2 will be different, while a1 and a2 will be considered identical for simplicity. The filter design will aim at finding g1. g2 will be optionally tuned in order to replicate the two-stage decay behavior if needed.

In [21] Välimäki et al. report a loop filter design procedure that will be implemented here. The procedure is based upon the STFT (Short Time Fourier Transform), a FFT technique used in order to capture the a signal spectrum that is not constant through time. As a prerequisite, extensive sampling of the test guitar was conducted resulting in 21 x 6 samples recorded and edited in wav uncompressed format. The files were then analyzed with MATLAB in order to track partial peak trajectories in time.

4.1.1 Theory

A guitar tone signal, like that of any other musical instrument, changes in time its amplitude envelope and its spectrum content. The STFT represents a good approach for tracking the time evolution of the frequency spectrum. The Fourier transform is localized in time, not being computed on the entire signal x(t) but upon portions of signals obtained multiplying x(t) for a translating signal window.

$$STFT(\tau, f) = \oint_{\infty}^{-\infty} x(t) * g(t-\tau)e^{-j2\pi ft}dt \tag{4.1}$$

The spectrum of the signal is computed inside the time window defined by the g function, thus the spectrum content refers to a small segment centered at time τ. The STFT results in a discrete sequence of DFTs, in which the window g is defined and segments of the signal x(t) are analyzed in each frame. The parameters needed to define the STFT are the FFT length N, the window function and the time advance per frame (or hop size).

In order to obtain a good frequency resolution, δf needs to be reduced, thus implying considering larger δt of time for the window and then a lower resolution in time. If δt is reduced to obtain a better time resolution, δf increases. Two events separated by an interval smaller than δt are not detectable. By analogy, two frequencies separated by an interval smaller than frequency resolution are not represented correctly when analyzed. A suitable compromise must be found. For this reason, the window (Blackman type) includes 2048 samples, and the hop size is equal to the samples producing an 83% overlap between two adjacent windows. In order to obtain a good frequency resolution, the signal frame is padded with zeros up to 4096 samples.

For each time frame, peaks are detected in the frequency spectrum and collected with their location and amplitude value. For each time frame, each harmonic is represented by a paired amplitude and frequency (or index value of the frequency vector). The sequence of these pairs is called the partial envelope. In theory, each partial should decay exponentially in a linear scale and linearly in a deciBel scale. For each envelope, a polynomial fit is performed looking for a 1st order function (a line), and the slope of interpolating function is computed. Following [21], the gain of the damping filter at a frequency k is computed as:

$$G_k = 10^{B_k L/20H} \tag{4.2}$$

Where L is the delay line length, B_k is the computed slope, H the hop size. For a signal x(t) a number of G_k is computed, defining the magnitude response to the damping filter sampled at certain frequencies. The phase content will be discarded as suggested by the method reference since it is more important to match the decay rate of the harmonics than of their phase.

4.1.2 Algorithm

The STFT is computed using the *spectrogram* MATLAB function, which outputs three variables: the FFT computed at different time frames, the frequency vector and the time vector. The FFT magnitude is then obtained and scaled to dB.

The function used to locate the peaks in the FFT magnitude spectrum is *findpeaks* in MATLAB. For each calculation, *findpeaks* produces an array containing locations and peaks values above a minimum threshold value. The size of these arrays is likely to be different at each time frame. The tracking of peaks is realized in a very simple way and its limitations might affect the precision of the obtained results. The construction of a peak envelope is done entirely on the frequency value. Therefore, peak values with the same frequency value are collected and ordered at their respective position in the time vector.

In particular, the following trajectories may appear in the peaks/locations vector for a collection of time frames (referring to Figure 4.1):

- Partial 1 begins at time frame 2 and is tracked to the end
- Partial 2 is tracked from the beginning and it is interrupted at frame 3
- Partial 3 is interrupted at frame 2 and found again at frame 4
- Partial 4 is tracked for each time frame under analysis

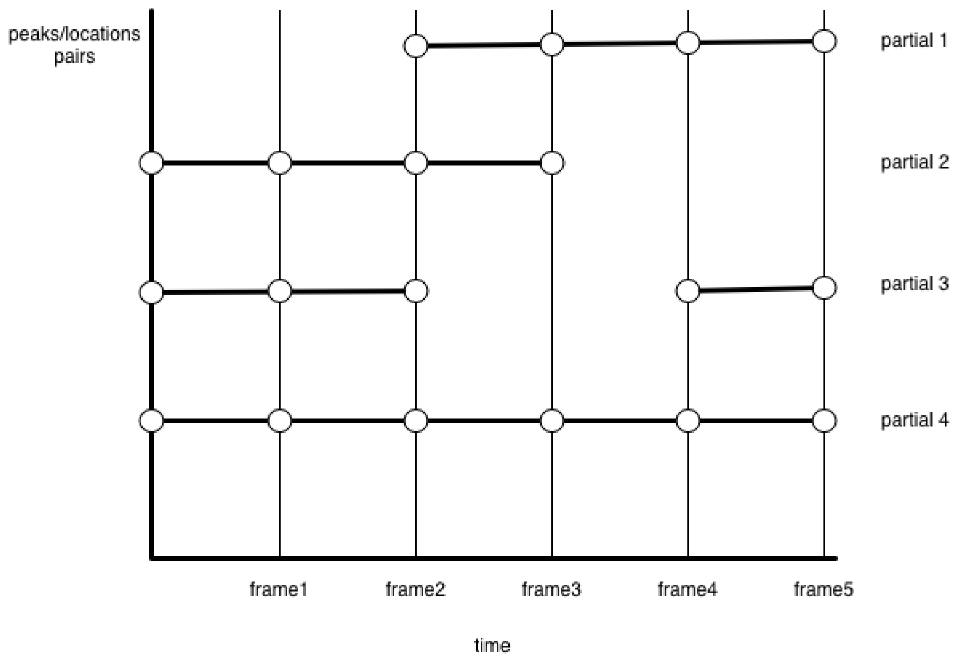


Figure 4.1: *Peaks tracking.*

Figure 4.2 represents the computed trajectories for first 5 partials of the A2 string plucked in the open position.

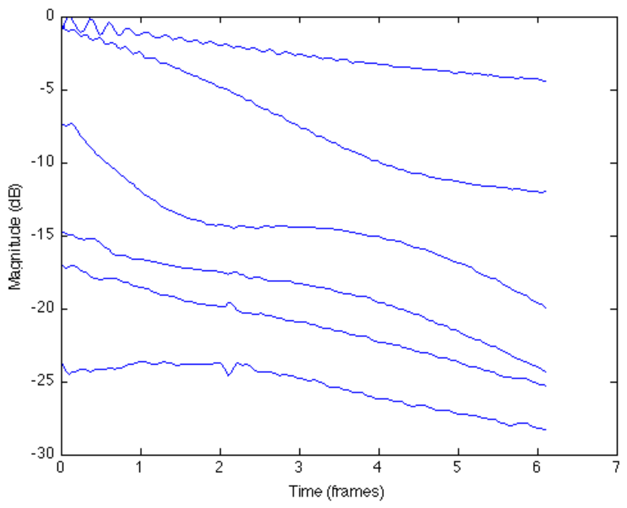


Figure 4.2: *Partials trajectories.*

4.1.3 Data Analysis and Filter Design

The guitar samples are imported from stereo wav files into MATLAB considering a single channel as seen, for example, with the left channel. The fundamental frequency and the nominal string length in samples are computed from the string number and the fret position. A first windowing of the signal is executed considering the portion of sound starting at 0.2 seconds up to 1.5/2.0 seconds. This portion of samples is divided in frames for the STFT, considering 30ms segments. The output of the STFT spectrogram function is converted to Magnitude dB and normalized to the maximum value of magnitude.

The FFT is computed by considering segments of length M, windowed by the Blackman function. At every iteration, the segment window on the original signal is advanced by the hop size R. The total number of iterations is equal to the number of frames *nframes*.

```
for m=1:nframes
  xt = x(xoff+1:xoff+M);                              % extract ...
      frame of input data
```

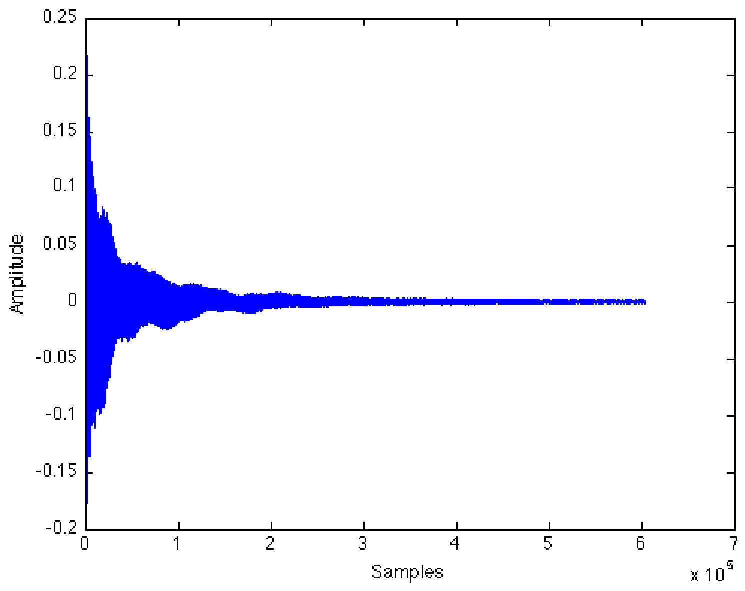


Figure 4.3: *Original string sample in wav format.*

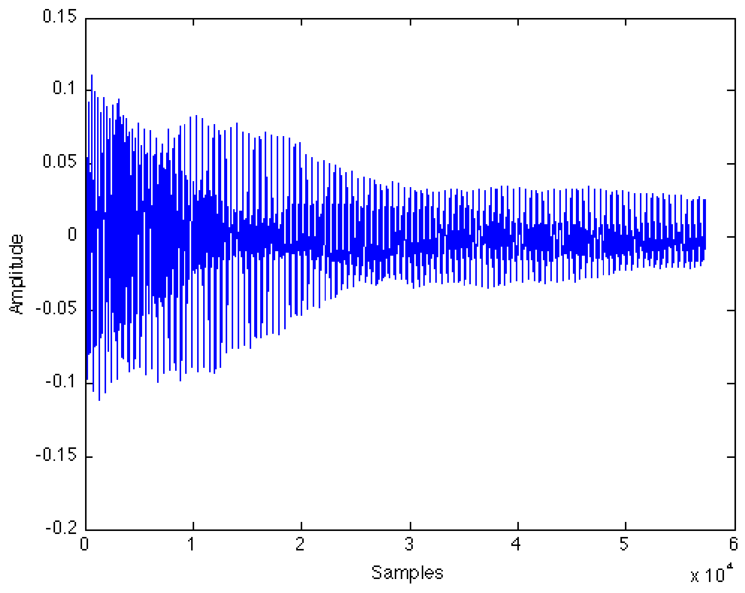


Figure 4.4: *Windowed sample.*

```
    xtw = w .* xt;                                      % apply ...
        window to current frame

    xwzp = [xtw(Mo2+1:M); zeros(N−M,1); xtw(1:Mo2)];

    Xtwz(:,m) = fft(xwzp,N);                              % STFT ...
        for frame m
    xoff = xoff + R;                                      % ...
        advance in−pointer by hop−size R

    moddB(2:(N/2),m) = 20*log10(abs(Xtwz(2:(N/2),m))*(2/N));
    moddB(1,m)       = 20*log10(abs(Xtwz(1,m))*(1/N));

end
```

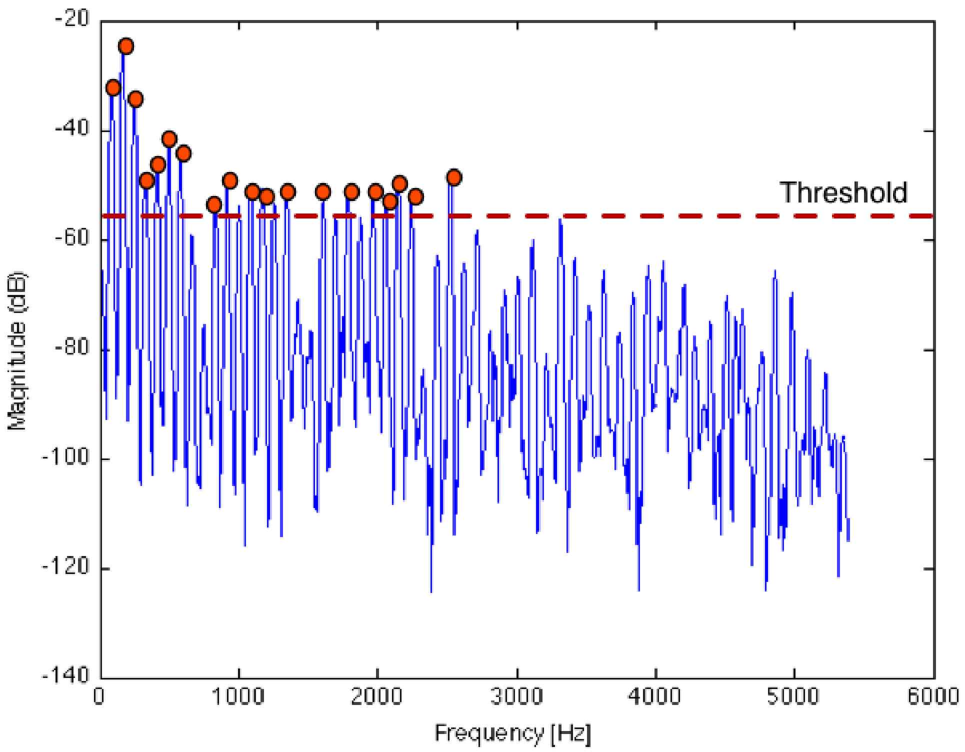


Figure 4.5: *FFT of a time frame and detected peaks*

In Figure4.5 the FFT of first frame of signal is represented, and peaks are highlighted. Peaks locations and amplitude are tracked along the frames computed by the FFT. The harmonic partials trajectories obtained are fitted with a linear function, and for every partial, a slope value m is computed. Using Equation 4.2, the gain of the low pass filter at that frequency is obtained.

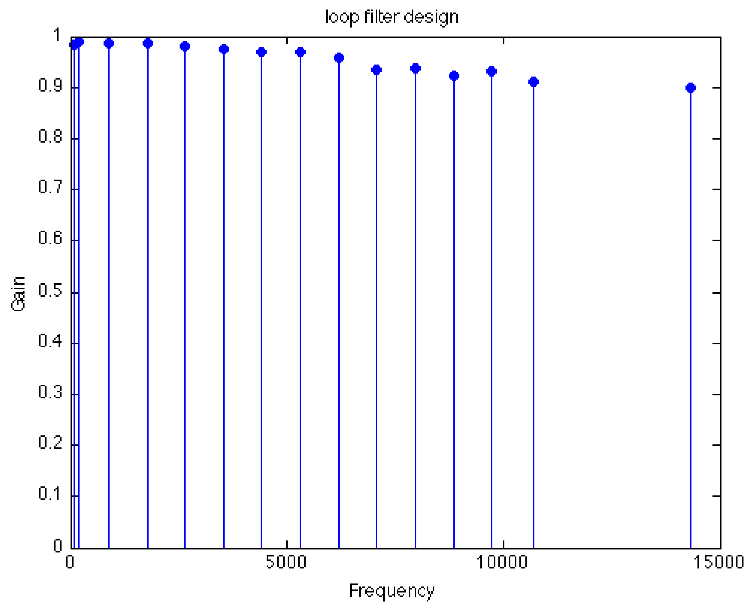


Figure 4.6: *Gains of partials obtained from trajectories slopes.*

The value of coefficient g is obtained from the value of G_0 or from the median value of G_0 and G_1. The a coefficient is obtained by minimizing the weighted least-squares error defined as equation 4.3. $H(\omega_k)$ is the filter magnitude weighted by the left term of the summation. The chosen weight gives more importance to slower decays, the ones that are perceived clearly by human hearing.

$$E = \sum \frac{1}{1 - G_k} \left[|H(\omega_k)| - G_k \right]^2 \tag{4.3}$$

The formula 4.3 computes the error of filter magnitude.The error function is plotted for values of the coefficient a between -1 and 1, and the minimum gives the approximated a value for the filter (Figure 4.7).

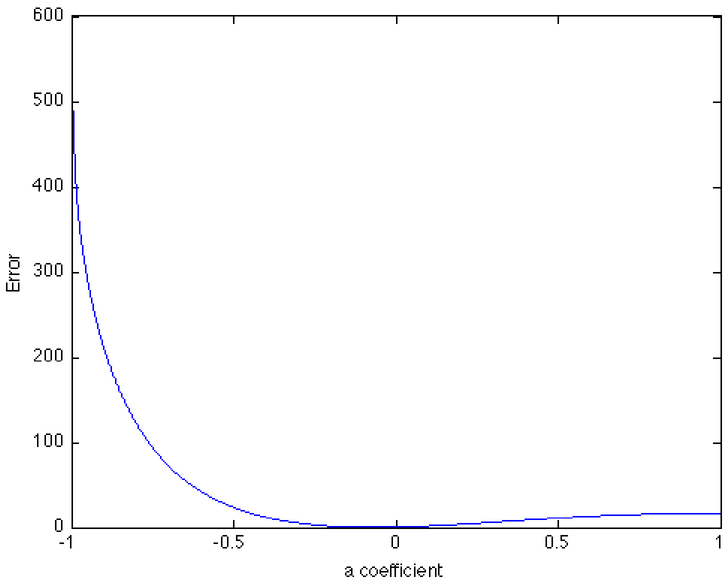


Figure 4.7: *Error function.*

Figure 4.8 shows the plot of G_k values together with the filter obtained from the design procedure.

The automated procedure saves the string number, fret number, g and a coefficient value into a $.DAT$ file.

4.1.4 Data Fitting

In order to fit the data (g and a coefficients) along the fretboard, a separate MATLAB routine performs interpolation of computed values with a 2nd order polynomial function. The results are shown in Figures 4.9 and 4.10.

To obtain the two coefficients, equation 4.4 is used in the realtime model:

$$\begin{aligned} g &= g_2 f^2 + g_1 f + g_0 \\ a &= a_2 f^2 + a_1 f + a_0 \end{aligned} \tag{4.4}$$

In equation 4.4 f is the fret number (f varying from 1 to 21). The pitch of the played note is converted to the fret number and for every note on message, the corresponding g and a coefficients for the string loop are obtained from

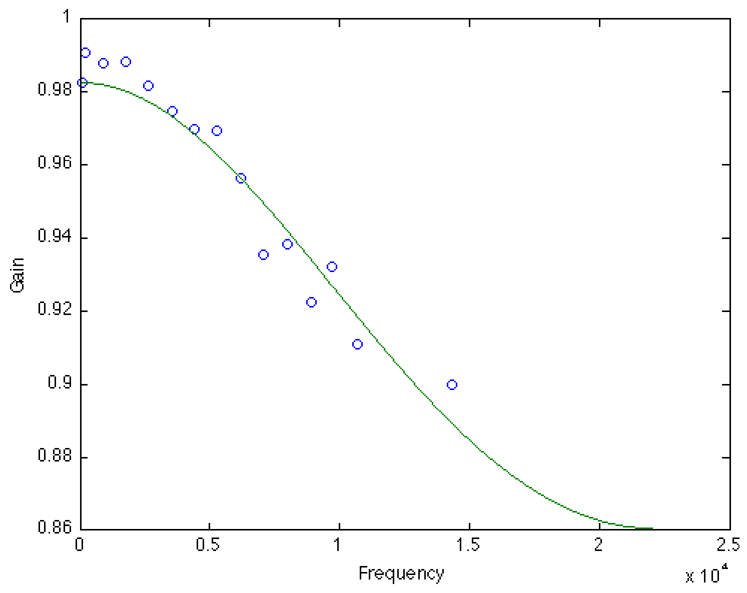


Figure 4.8: *Filter Design.*

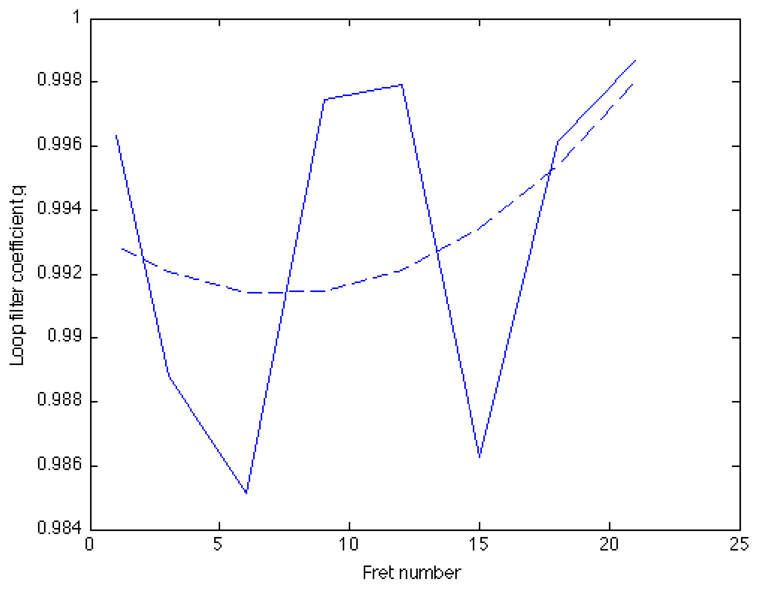


Figure 4.9: *The filter coefficient g for string E2: filter design (solid line) 2nd order polynomial fit (dashed line).*

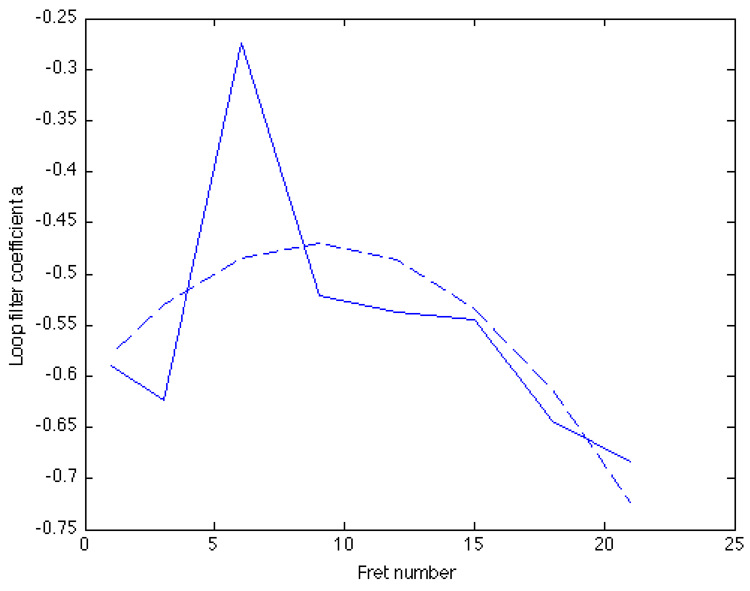


Figure 4.10: *The filter coefficient g for string E2: filter design (solid line) 2nd order polynomial fit (dashed line).*

the equation 4.4.

The two obtained coefficients are applied to the entire string, yet not considering their application related to horizontal and vertical polarization. The filter design has been performed on a signal that is the sum of the two planes of vibration. A possible solution to identify coefficients for both planes is to analyze two different segments of a sampled tone with STFT (the attack in which vertical decay is predominant, and the sustain in which the horizontal decay is predominant). The amplitude decay has to be divided into two halves, but this might not easyly identify two stages for every sample insofar as they are not always clearly separated, and the first stage might be too short in time for STFT analysis.

The obtained values of g and a will then be used for the horizontal polarization model, and the vertical plane coefficients will be adjusted in order to obtain the correct decay time, thus varying only the g coefficient. The a coefficient is considered equal for both planes.

Table 4.1: *Second order polynomial coefficients for g.*

	g_2	g_1	g_0
string 1	0.00004	-0.00054	0.99335
string 2	0.00003	-0.00064	0.99942
string 3	-0.00007	0.00185	0.98513
string 4	0.00003	-0.00092	1.00176
string 5	0.00002	-0.00050	1.00015
string 6	0.00008	-0.00159	1.00125

Table 4.2: *Second order polynomial coefficients for a.*

	a_2	a_1	a_0
string 1	-0.00172	0.03067	-0.60670
string 2	0.00254	-0.04530	-0.32087
string 3	0.00185	-0.05034	-0.08251
string 4	0.00037	-0.00818	-0.16793
string 5	0.00109	-0.02065	-0.09310
string 6	0.00017	-0.00435	-0.06550

4.2 Body Impulse Response

The body is modelled by obtaining its impulse response, which fully defines the body itself if the guitar system is assumed to be linear and invariant. Different methods are available for obtaining an impulse response for the guitar body. The differences regard both the impulse generation and the measurement techniques. A simple form of impulse can be achieved with a tuned hammer, usually striking the top plate near the bridge. In this case, a simple setup was organized, and the IR measurement procedure was performed by means of a MLS sequence generated in MATLAB [15].

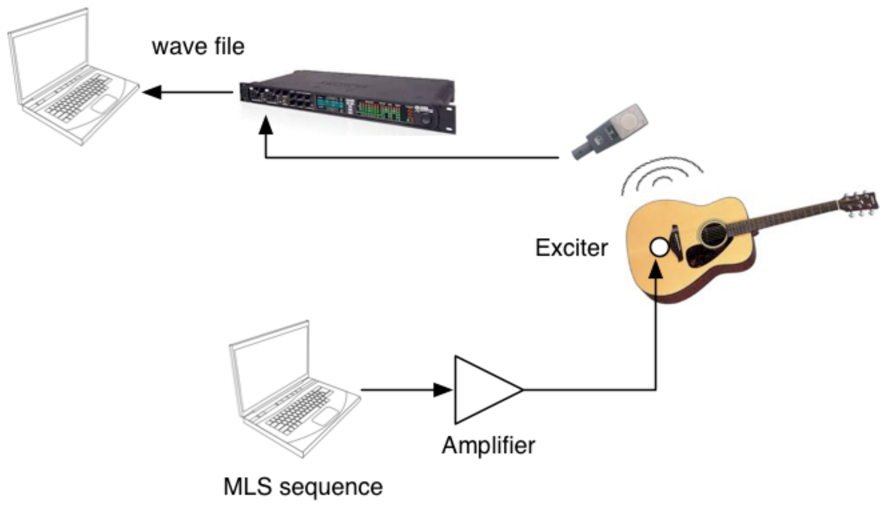


Figure 4.11: *Recording System.*

The excitation signal or MLS sequence is considered to be the input x(t) to the system, in this case:

$$y(t) = x(t) * h(t) \tag{4.5}$$

The output y(t) is recorded, and, since the x(t) excitation signal is a defined function (a pseudo-random defined sequence), deconvolution is applied to obtain the h(t) impulse response signal. The excitation is applied to the guitar under the bridge with a DML piezoelectric transducer, so that the

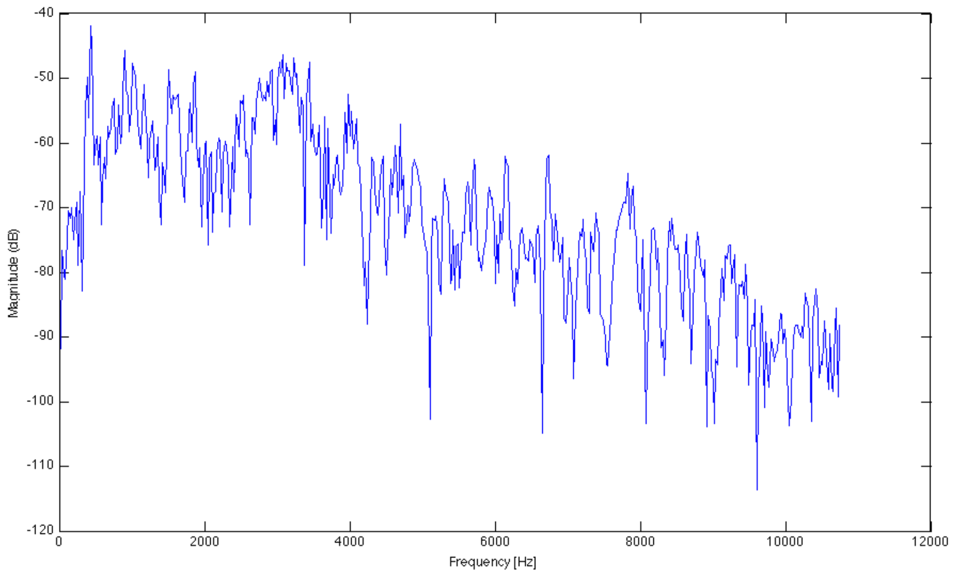


Figure 4.12: *Frequency response for IR of the test guitar.*

excitation is repeatable and well-defined.

A first computer generates the MLS sequence in MATLAB and plays the sequence via the sound card through an amplifier and finally to the DML exciter placed under the guitar bridge. The guitar has been placed on a vertical guitar stand with the strings damped. A condenser microphone has been placed at 1m from the guitar to record the output of the excited soundboard. An *AKG414* has been chosen for its virtually flat frequency response from 20 to 20 KHz. The signal is then acquired through a pre-amp powered with 48V into a *MOTU* sound card on a second computer and recorded with GPL software Audacity. In the final step, the wav file has been treated with *Voxengo Deconvolver* in order to eliminate the original impulse x(t) from the file.

Figure 4.12 shows the impulse response and its frequency spectrum.

The prepared experimental set-up showed the limitations of the used guitar and the exciter. Looking at the frequency response, it is possible to observe that the lower resonances are not well defined around 100 and 200

Hz. Typically, at these frequencies it is possible to observe resonances due to the air cavity and the two plates of the body. The Helmholtz resonance is applied to a resonance around 100Hz and is determined by the cavity volume and the diameter of the sound hole. At this frequency, the air moves in phase with the top plate [16]. The reason for the lack of the resonances can be due to the following:

- The exciter used did not provide enough energy to trigger the lowest resonances.
- The room used for the recording altered the recorded response at the microphone.

A synthetic impulse response was thus created adding two resonances with MATLAB using the concept of body model factoring. Rather than factoring out the resonances from an existing IR, here two resonances are added to obtain an arbitrary impulse response. The code used is the same showed in Chapter 3 [20][1].

Figure 4.13 shows the impulse response frequency spectrum after the MATLAB filtering process. Two resonances have been added at about 100 and 200 Hz.

The impulse response now has to be convolved with an excitation signal. The most simple type of excitation is an impulse signal although it is possible to approximate the real excitation by creating a series of functions and obtaining different samples. The excitation represents the force over time as input to the string model, and the player can vary this to produce different sounds by tweaking different parameters:

- Position of pluck point.
- Interaction between the pick and the string.
- Initial displacement (force).

[1] `http://ccrma.stanford.edu/~jos/pasp/Matlab_Code_Inverse_Filtering.html`

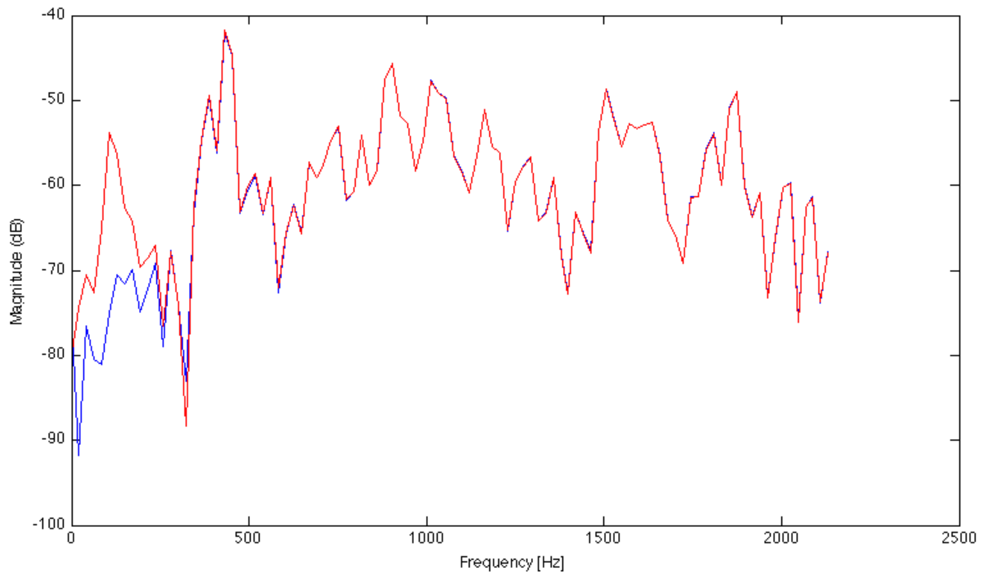


Figure 4.13: *Frequency response for IR of the test guitar.*

The interaction modelling is outside the scope of this work therefore the excitation will be modelled through a definition of force duration and its distribution over time. This signal is generated in MATLAB and subsequently convolved with the previous impulse response to obtain an aggregate excitation wavetable. The wavetable obtained will be used as an input in the Max/MSP patch for the string model and will be triggered by the incoming MIDI event.

Figures 4.14 and 4.15 represent two possible excitation signals with different distribution over plucking time.

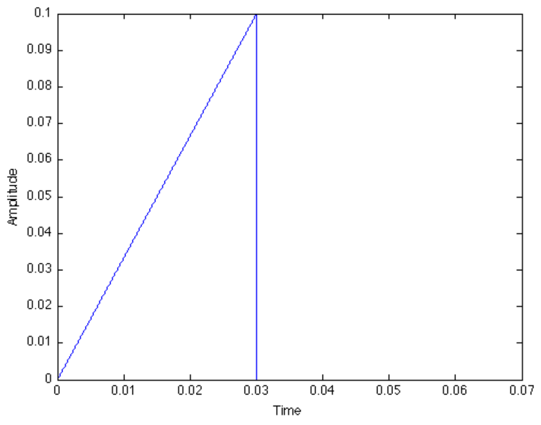


Figure 4.14: *Excitation signal - linear distribution over time.*

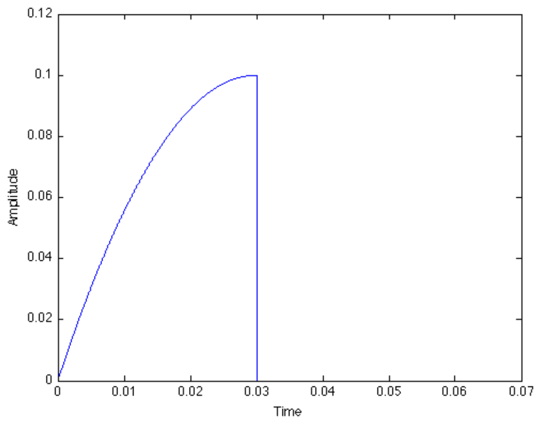


Figure 4.15: *Excitation signal - quadratic distribution over time.*

Chapter 5

Discussion

In this chapter, the output tones generated from the real time synthesizer will be analyzed in MATLAB and compared with the sampled reference guitar. The analysis will consider the amplitude envelope through time, in particular the two-stage decay as well as the frequency spectrum evolution during sound decay.

5.1 Decay Analysis

The first analysis will take into consideration a sampled tone against a synthesized one with respect to the amplitude envelope. The test tone was set at $246.9Hz$ corresponding to B3. The two tones are plotted in MATLAB in Figure 5.1.

The amplitude envelope for this tone shows that the sample features a weak two-stage decay, which was obtained by calibrating the g parameter for the vertical polarization (together with a small detuning factor). The comparison shown in Figure 5.1 shows that the two decay shapes are similar. The sampled tone features a beating that is evident around the boundary between the two stages and a more complicate envelope during the attack. The reason for these differences will be explained in the EDR analysis.

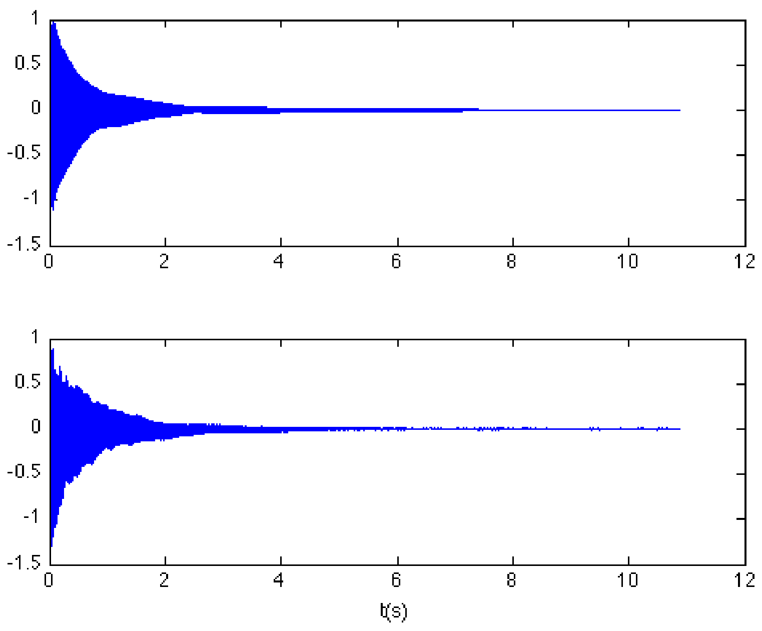


Figure 5.1: *Amplitude: comparison between a sampled B3 tone against a synthetic tone.*

5.2 Pitch

This section presents a comparison between the pitch obtained from sampled and synthesized tones. Pitch detection can be obtained with different methods, one of which is the autocorrelation method. In this method, the highest value of autocorrelation function is found within a region of interest.

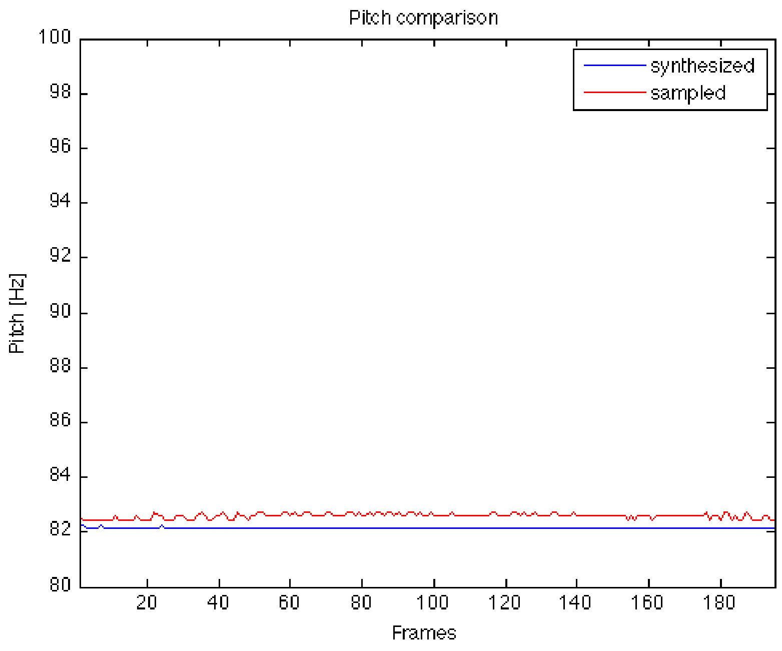


Figure 5.2: *Pitch comparison.*

In Figure 5.2, a comparison between two E2 tones has been obtained with a MATLAB implementation of an autocorrelation algorithm. In this case, the perfect E2 tone is equal to $82.41Hz$. The pitch detection shows that the synthesised tone is slightly lower than the sampled one. No measurement has been taken from the sampled guitar to estimate the precision of the tuning. The plotted quantity is an indication of a mathematical parameter and not the perceived pitch which is a subjective parameter. The difference between the two tones is around $0.2 - 0.3Hz$, which is lower than perceived threshold in that range of frequencies (around %3 at $100Hz$).

In Figure 5.3, a comparison between two B4 flat tones is plotted. The

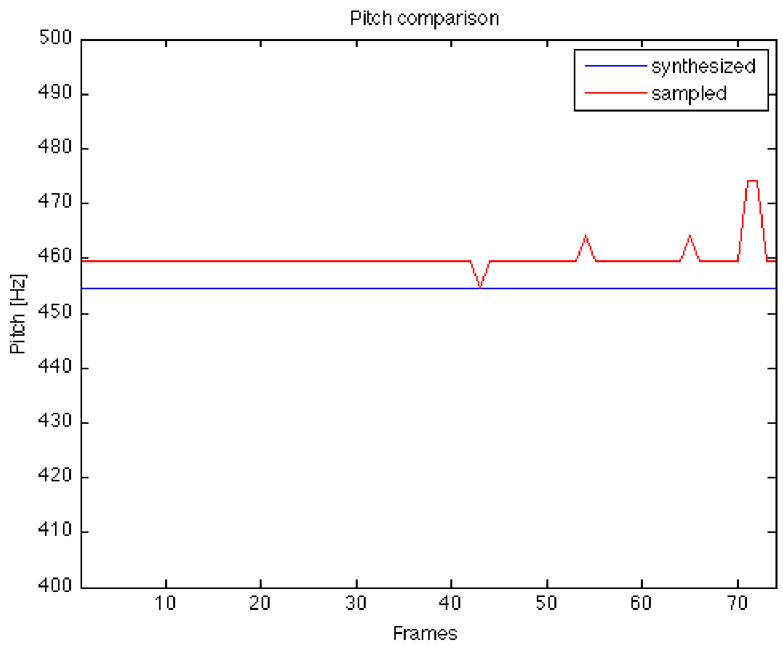


Figure 5.3: *Pitch comparison.*

predicted fundamental pitch of a perfect B4 flat is $466.2Hz$. The analysis output $460Hz$ for the sampled tone and about $454.6Hz$ for the synthetic tone. A difference of $5Hz$ is now evident also at a perception level and the cause of the mistuning has to be found inside the all-pass tuning filter. An accurate analysis of the filter is needed to improve the precision at higher tones.

5.3 EDR Analysis

The hypothesis assumed for the body or resonator imply a linear and time invariant system. This means the resonator is completely defined by its impulse response. Analysis of the body impulse response shows that it is composed of several harmonics decaying exponentially. The Energy Decay Relief (EDR) is the distribution over time and over the sampled spectrum of the total energy of a signal and is based on a STFT. A single point of the EDR is the signal energy at a defined time frame and defined frequency band.

All the EDR comparisons refers to the following parameters:

- Pitch position 1/4
- Pluck angle 55 deg (from vertical plane)
- Detuning 0.5 Hz (vertical pitch = horizontal pitch - detune factor)

Once again, the comparison is done using the reference sampled tone versus the synthetic tone. In Figures 5.4 and 5.5, the comparison represents a sampled E2 tone played on the 6th string and a E2 tone played by the synthesizer.

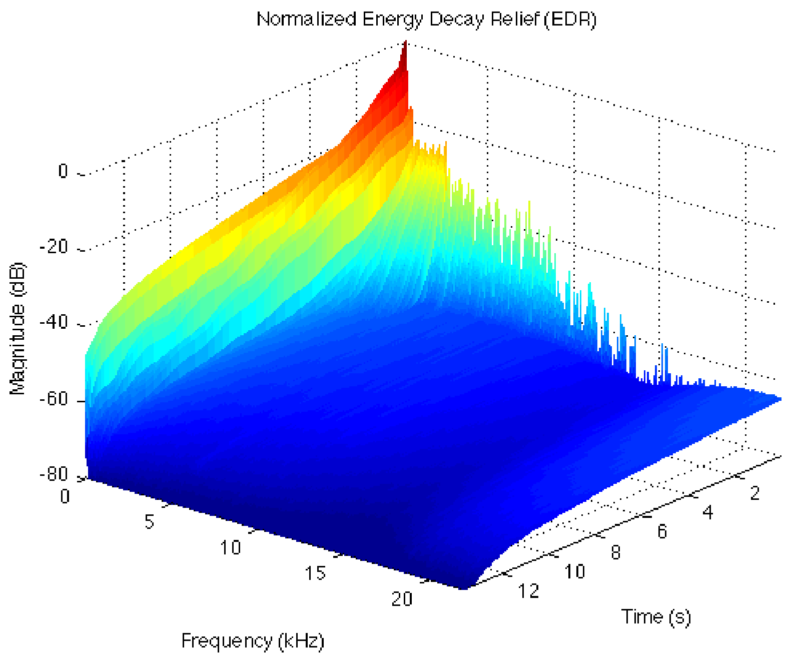


Figure 5.4: *E2 open: sampled tone.*

The sampled tone shows that there are a few less-damped partials (the lowest being the strongest one in this case). A varying decay slope can be observed for two harmonics, while, for the others, what emerges is a ringing

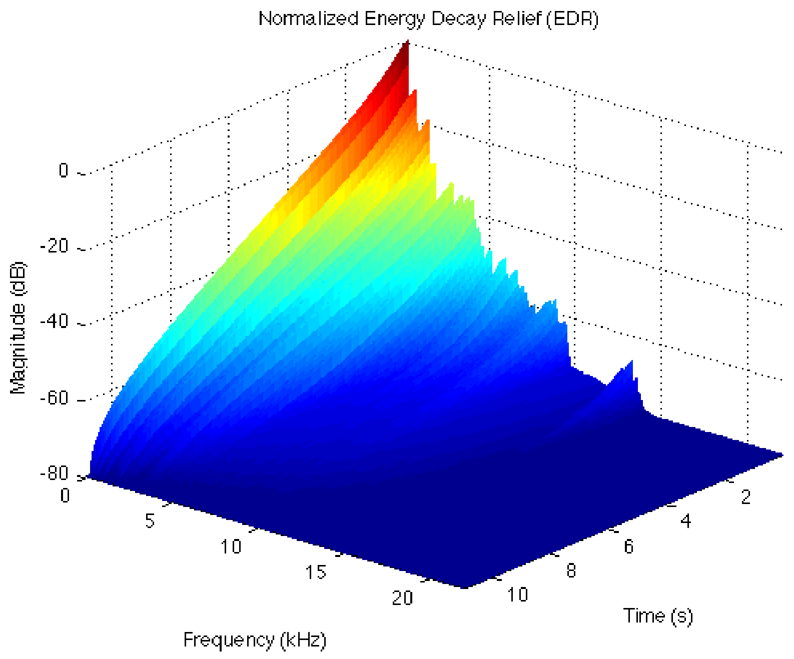


Figure 5.5: *E2 open: synthetic tone.*

effect, that corresponds to an exchange of energy between close frequencies. Moving towards the upper region of the spectrum makes visible more harmonics whose decay rate is much higher than of the lower ones. The last partials above $10KHz$ decay within the first second if not sooner. Floor noise contribution is also present in the recording, which affects the frequency content for the entire spectrum.

The analysis of the synthesized tone leads to the following considerations:

- The distribution of decays over the partials is more uniform than that found in the sampled tone. The two-stage decay is not present for the fundamental partial due to missing calibration for the vertical polarization gain.
- Some partials are missing between 10 and 15 KHz.
- The decay seems to be the same for all the partials below 5 Khz. Also, the decay of the lowest modes is too fast. This is due to calibration

of the loop filter as well as to the fact that the loop filter with 1 pole cannot correctly describe the complex decay patterns.

- No spectral content above 15 KHz. This may be also caused also by the bandwidth of the excitation and body impulse response. The synthesized tone has no noise floor which helps in discerning missing components from the spectrum.
- In general, high frequency decay is too slow above 10 KHz.

The next tones analyzed are the E4 played on the open 1st string. The sampled tones feature more partials than the previous, covering almost the entire sampling bandwidth. In this case, the complexity of decay patterns is even more evident, in particular for the lower partials. The decay rapidly falls above 5 Khz. The lowest partial has a very long decay, and the last part has a slope close to 0.

The synthesized tone is missing the lower content, and again all the partials have a linear decay. There are partials missing in the bandwidth between 12 and $15KHz$ and above $16KHz$. In general, the decay seems too fast for all the partials.

The reason for the missing fundamentals below $2KHz$ may be found in the body and bridge modelling. The model works under the assumption that the body is a linear system and the model foresees only a one-directional path from the string to the body itself. The string has non-linear characteristics that have not been included into the proposed model and the string itself is treated as a linear element for commuted synthesis.

In the real guitar, the lowest modes resonances $(0, 0)$ and $(1, 0)$ (Helmholtz air mode and see-saw mode) can be triggered by a plucked string depending on the direction of excitation. For example, a pluck in the vertical plane excites a $(0, 0)$ mode and a $(1, 0)$ mode if applied to the treble and bass sides. A pluck in the horizontal plane excites the $(1, 0)$ mode [16]. It is evident that a single excitation file is not capable of simulating all these different plucking conditions.

A detailed representation of the lowest partials can be observed in Figures 5.8 and 5.9.

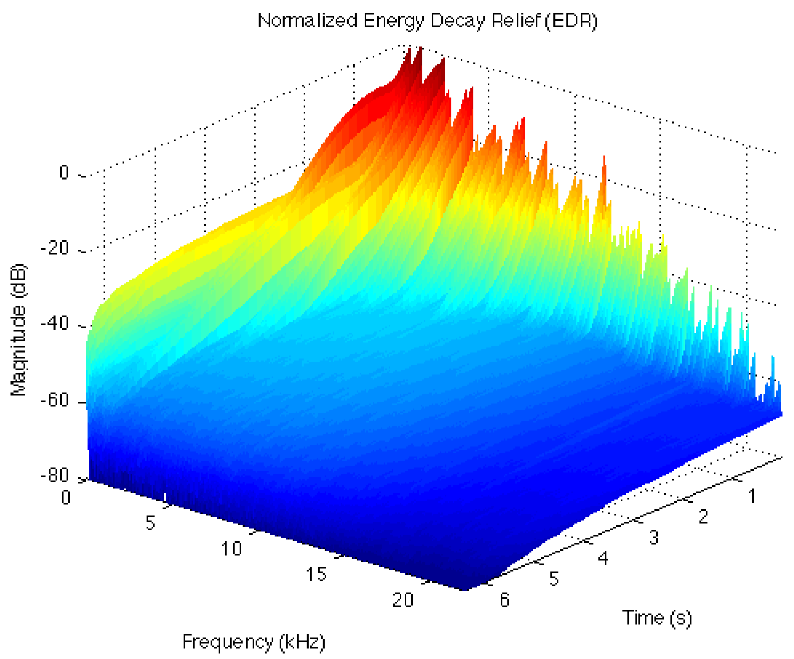


Figure 5.6: *E4 open: sampled tone.*

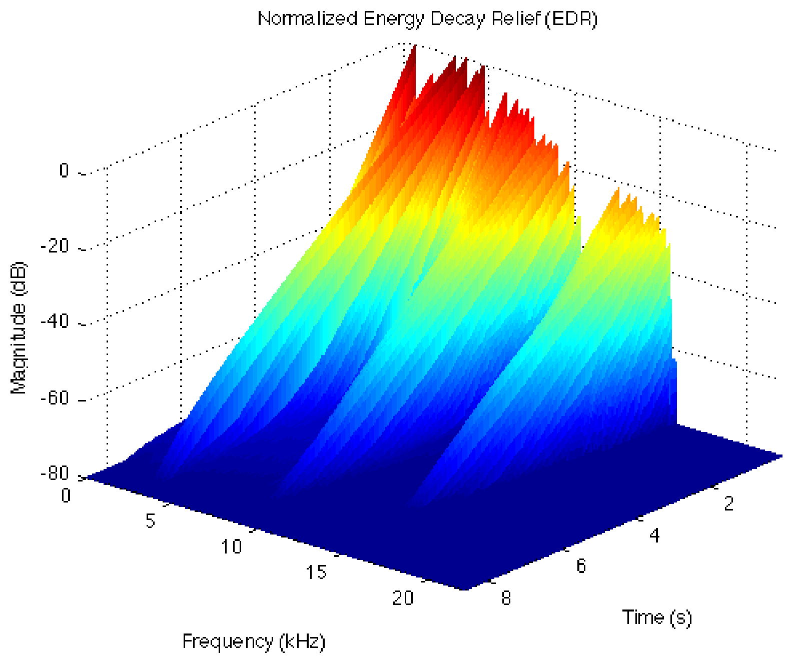


Figure 5.7: *E4 open: synthetic tone.*

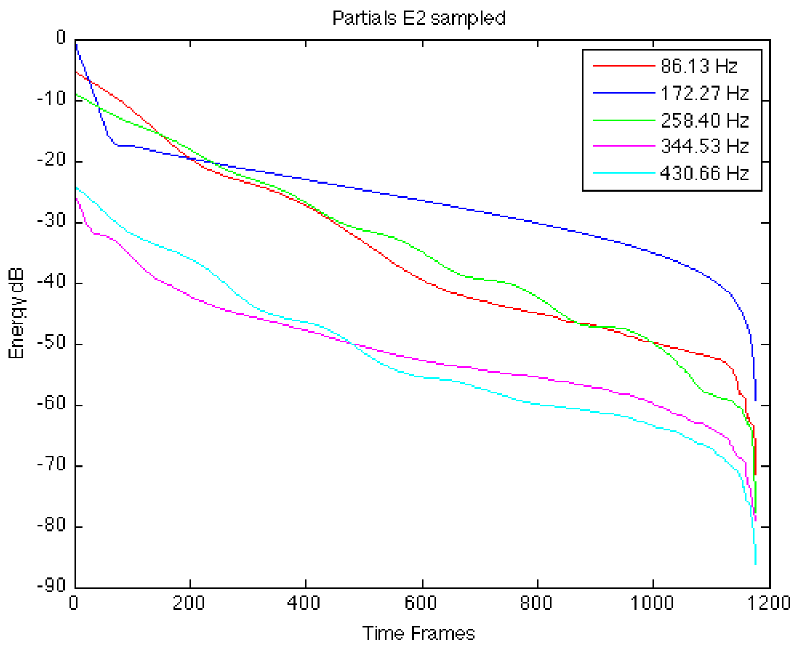


Figure 5.8: *E2 open: sampled tone.*

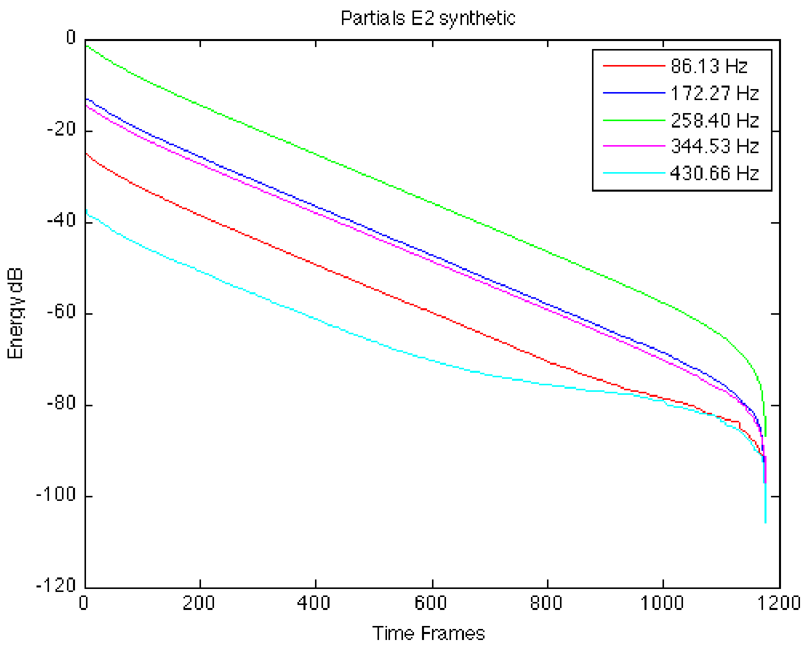


Figure 5.9: *E2 open: synthetic tone.*

All the partials in the sampled tone feature complicated patterns, including ringing and two-stage decay. In particular, the second partial for the two-stage decay as well as the third and fifth partials feature a ringing pattern. The fundamental frequency is mixing these two aspects exhibiting ringing in the first part and a slight slope change in the second part.

The synthetic tone does not contain all the complex information of the sampled tone. The results are close for what concerns the decay of the first, third and fifth partial. The second partial instead decays rapidly in comparison to the sampled partial.

The remaining comparisons are contained in Appendix 3.

The analysis showed the limitations of the current proposed model for what concerns the partial decay patterns. This is due to the design choice to implement the string as two separate entities representing the planes of vibration. The crucial point to a better model would be the design of a bridge filter or an admittance matrix. All six strings, each one modeled onto the two planes of vibration, could be connected to the same admittance matrix[2].

The bridge matrix could model the exchange of energy at the extremities of the six string between the two planes of vibration.

Improvements can be also achieved regarding the body impulse response. The experimental setup has to be repeated in an anechoic chamber with different exciters and sensing devices. The anechoic chamber prevents any coloration of sound during to wall reflections. Once the impulse response is obtained, the resonator could be designed to match the IR itself. Further considerations about model parameters will be detailed in the next chapter.

Chapter 6

Conclusions

6.1 Outcomes

The purpose of this dissertation was to create a physical model of a guitar string using the algorithm analyzed in the existing literature [9] [10] [22]. The Karplus-Strong algorithm, together with additional extensions, was implemented with two software tools: MATLAB and Max/MSP. In particular, the latter was used to create a real-time MIDI synthesizer that can accept standard MIDI input from an external controller and produce an output to an audio card on a common PC. MATLAB was used to perform the calibration procedure, to test the single string model and to analyze the results obtained from the synthesizer.

It was possible to add a set of initial parameters to the model for decay, body characterization, pluck angle, pluck position, dynamic range and sympathetic mixing gain. Some of these parameters are available to the user insofar as they represent physical parameters of a guitar player (pluck angle, pluck position, dynamics). The modular structure of the code in Max/MSP permits easy changes to the existing blocks or the addition of new features without having to rewrite the other modules. Whenever needed, the Gen compiling environment was used instead of the classic MSP environment. Gen is a Max/MSP low level language for audio processing and processes one sample at a time (unlike common MSP modules).

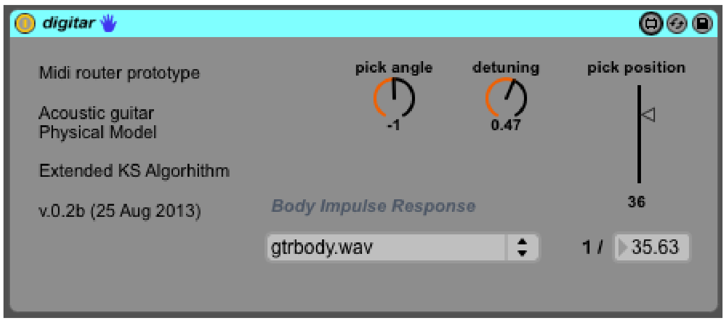
6.2 Limitations

The analysis presented in Chapter 5 pointed out the limits of the string and body models in emulating the reference instrument. In particular, the EDR analysis of partial decay showed that a bridge model is needed to model the exchange of energy between the two polarization planes. Modelling the two-stage decay is important also at a perception level in that decay cannot be modeled in the same way for all the partials, but complex patterns are found in partials and are analyzed separately with STFT. Modelling the bridge is also compliant with the physical modelling philosophy and is more meaningful than having two decoupled waveguides and simply summing up their combined output. The proposed string damping filter is defined by two coefficients on each plane, for a total of four coefficients for the complete string model. The calibration procedure estimated only two coefficients, and a procedure was needed to estimate at least the gain coefficient of the second filter (considering the pole coefficient with the same value on both planes). If the gains of two planes are equal, the decay will be exponential. If the two gains differ (horizontal decay slower than vertical decay), a two-stage decay is produced. Another parameter is the tuning of the two waveguides. Detuning a polarization plane produces beatings in the resulting tone, which gives a more natural effect[13]. The detuning and the damping filter coefficients have to be chosen to match the sampled tones, but this cannot be done directly with a procedure.

The use of commuted synthesis for modelling the body resonator produces a hybrid model, mixing physical modelling and sampling techniques. A better solution from a modelling point of view would be to design a resonator with some parameters available to the user. This will increase the computational load of the patch, while commuted synthesis is used for it simplicity and efficiency.

6.3 Max4Live

The Max/MSP patch has been integrated into Ableton Live using the Max4Live add-on package. The patch is available to the user as a MIDI virtual instrument without needing to compile a single line of code. The GUI is designed by selecting which control elements have to be shown to the user, and the entire patch is connected to special interface elements (midiin and plugout) which link the Max/MSP patch with Ableton MIDI and audio busses.



Midi router prototype

Acoustic guitar
Physical Model

Extended KS Algorhithm

v.0.2b (25 Aug 2013)

Figure 6.1: *Max4Live virtual instrument*

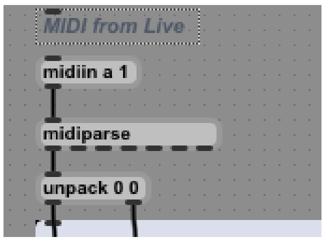


Figure 6.2: *Max4Live virtual instrument: the input of a Max4Live instrument is the midiin element. MIDI messages are routed from Ableton to the selected track using the virtual instrument*

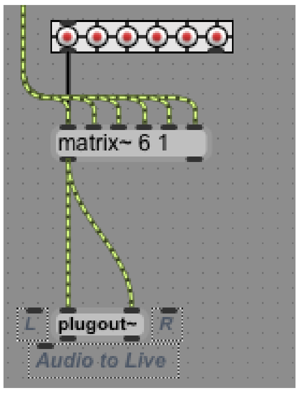


Figure 6.3: *Max4Live virtual instrument: the output of a Max4Live instrument is the plugout element. The generated audio becomes an audio output in Ableton*

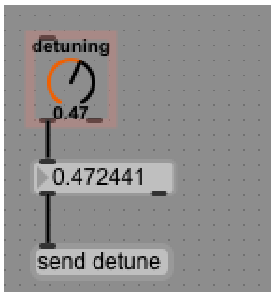


Figure 6.4: *Max4Live virtual instrument: the dial element contained in the GUI here is presented together with Max/MSP elements*

6.4 Future Developments

A possible improvement on the string block would be to integrate the polarization planes with a passive bridge filter. The design of the filter can be accomplished only by measuring the velocity of bridge displacement with a micro-accelerometer mounted on the top plate or other sensing devices. Once measurements have been done, the passive 2x2 matrix can be computed.

In this work, the waveguides accept a fixed floating point pitch as an input. This value corresponds to the MIDI pitch value. An improvement to the waveguide could be to implement dynamically changing delay lines to simulate glissando effects. This would bring up the issue of simulating the collision between the string and the frets in order to generate a more realistic sound.

Another development concerns the commuted synthesis technique. In [21], the excitation signal was obtained by inverse-filtering the sampled tones with an inverse transfer function of the string. In [22], a sinusoidal model was subtracted from a sampled tone. If an estimation of the pluck position can be performed, its effect can be subtracted from the excitation signal and added later as a parameter.

Sympathetic vibrations have been implemented through a mixing matrix (six inputs and six outputs). In order to eliminate feedback loop, the elements on main diagonal of the matrix have been set to zero. The remaining gain coefficients are real numbers between 0 and 1. Measurements have to be performed to evaluate these gains separately for a future development of this module.

In the current version of the synthesizer, the MIDI module analyzes the incoming messages and routes the note on message to the correct string module. This module could be written more efficiently as an external in Max/MSP, which is a new object created using the Max/MSP development tools and C programming language. PureData, an open source project, is a descendant of the Max family written by Miller Puckette that shares many characteristics with Max/MSP. Rewriting the entire patch in PureData will lead to an open source instrument preferable to a commercial solution. The efficiency of both solutions and the quality of their audio outputs to have a complete scenario must be assessed.

Appendix A

Project plan

Title: *Developing a Physical Model for a Plucked Acoustic Guitar*
Student: *Marco Chrappan Soldavini*
Supervisor: *Dr Ian Drumm*

A.1 Introduction

Physical Modelling is a particular kind of sound synthesis in which the physics of musical instruments is applied to sound generation. Instead of creating the sound directly with oscillators and filters, it aims to model physical processes that generate sound in the instruments.
The results of Physical Modelling are:

- Expressive and realistic sounds,
- No need for dedicated hardware or lots of memory to store samples,
- Possible real-time implementation of Physical Modelling on generic personal computers.

This project will involve the creation of a physical model for an acoustic guitar. The model will be designed in order to be in real-time and to accept MIDI commands from an external controller. In particular, the model will be calibrated against a real instrument, an acoustic flat-top guitar using wav samples recorded in a studio. The aim is to build the foundations for a guitar simulator that can be easily expanded. For this reason, a programming language specific for realtime audio processing will be used for development of the synthesizer.

A.2 Objectives

1. Develop a physical model for a plucked string in MATLAB and Max/MSP using delay lines and filters;
2. Develop a simplified model for string boundary conditions;
3. Develop a simplified model for the guitar body;
4. Calibrate the model against a reference instrument of choice;
5. Perform comparisons between recorded instruments and synthesized guitar tones.

The following objectives may be attempted, if time permits:

1. Develop a polyphonic MIDI system for receiving data from an external MIDI controller;
2. Develop a real-time standalone plugin for Ableton Live DAW software that accepts MIDI input from recorded and live performances.

A.3 Equipment Requirements

Reference guitar, PC workstation, MATLAB, MAX/MSP, Studio microphones for recording guitar samples, A/D converters, DAW Software.

Week → Activity ↓	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Prepare project plan	▨															
Literature review	▨	▨	▨	▨												
KS Algorithm Matlab				▨	▨											
EKS Algorithm MAX/MSP						▨	▨									
Guitar sampling								▨								
MAX/MSP Model								▨	▨	▨	▨	▨	▨	▨		
Model Calibration									▨	▨	▨					
Body IR												▨	▨			
Write final report									▨	▨	▨	▨	▨	▨	▨	▨

Figure A.1: *Plan*

Appendix B

STFT MATLAB Code

B.1 STFT

```
% opening file dialog
[FileName,PathName,FilterIndex] = uigetfile('*.wav','Select ...
    wav sample to be analyzed');

% note frequencies vectors
E2=[82.51 87.31 92.50 98.00 103.8 110.0 116.5 123.5 130.8 ...
    138.6 146.8 155.6 164.8 174.5 185.0 196.0 207.7 220.0 ...
    233.1 246.9 261.6];
A2=[110.0 116.5 123.5 130.8 138.6 146.8 155.6 164.8 174.6 ...
    185.0 196.0 207.7 220.0 233.1 246.9 261.6 277.2 293.7 ...
    311.1 329.6 349.2];
D3=[146.8 155.6 164.8 174.6 185.0 196.0 207.7 220.0 233.1 ...
    246.9 261.6 277.2 293.7 311.1 329.6 349.2 370.0 392.0 ...
    415.3 440.0 466.2];
G3=[196.0 207.7 220.0 233.1 246.9 261.6 277.2 293.7 311.1 ...
    329.6 349.2 370.0 392.0 415.3 440.0 466.2 493.9 523.3 ...
    554.4 587.3 622.3];
B3=[246.9 261.6 277.2 293.7 311.1 329.6 349.2 370.0 392.0 ...
    415.3 440.0 466.2 493.9 523.3 554.4 587.3 622.3 659.3 ...
    698.5 740.0 784.0];
```

```
E4=[329.6 349.2 370.0 392.0 415.3 440.0 466.2 493.9 523.3 ...
    554.4 587.3 622.3 659.3 698.5 740.0 784.0 830.6 880.0 ...
    932.3 987.8 1047.0];

% input string fundamental frequency
prompt1 = 'Please enter string number (1=E2 6=E4):';
string = input(prompt1);

prompt2 = 'Please enter fret number (1=open string 21=last ...
    fret):';
fret = input(prompt2);

switch string

    case 1
        f0=E2(fret);
    case 2
        f0=A2(fret);
    case 3
        f0=D3(fret);
    case 4
        f0=G3(fret);
    case 5
        f0=B3(fret);
    case 6
        f0=E4(fret);
end

fprintf('Working on fundamental frequency: %f Hz\n',f0);

frameSizeMS = 30; % minimum frame length, in ms
overlap = 0.83; % fraction of frame overlapping
windowType = 'blackman'; % type of windowing used for each frame

[signal,fs,bits] = wavread(FileName);
L   = round(fs/f0);
signal = signal(floor(0.1*fs):floor(1.5*fs),1);

% calculate STFT frames
```

```
minFrameLen = fs*frameSizeMS/1000;
frameLenPow = nextpow2(minFrameLen);
M = 2^frameLenPow; % frame length = fft size
eval(['frameWindow = ' windowType '(M);']);

R=floor(overlap*M);

[B,F,T] = spectrogram(signal,frameWindow,R,2*M,fs);

% Magnitude in Decibels

B_dB = 20*log10(abs(B));
offset = max(max(B_dB));
B_dBN = B_dB-offset;

% Maximum number of peaks for frame

max_peaks=200;
nframes=size(B,2);

peaks_matrix=zeros(2*nframes,max_peaks);

% peaks locations matrix (odd rows: peak value , even rows: peak
% location - column peaks, rows frames)

% the following loop computes for each frame the location of ...
    freq. spectrum
% peaks filling peaks_matrix

for frame_i=1:nframes
    [pks,locs] = ...
        findpeaks(B_dBN(:,frame_i),'minpeakheight',-50);

    npeaks = length(pks);

    for i=1:npeaks
        peaks_matrix(2*frame_i-1,i)=pks(i);
        peaks_matrix(2*frame_i,i)=locs(i);
    end
```

```
end

locations=[];

% this section computes how many frequency index occurences ...
    are present in
% peaks_matrix ie finding unique entries

for index=1:nframes
    locations=[locations peaks_matrix(2*index,:)];
end

unique_v = unique(locations);

if unique_v(1) == 0
    unique_v = unique_v(2:length(unique_v));
end

unique_count = length(unique_v);
loc_count = [unique_v ; histc(locations,unique_v)];
fprintf('computed %f unique partials\n',size(loc_count,2));

% good_matrix reorder all data per partial - odd row partial ...
    magnitude,
% even row time index

good_matrix=zeros(2*unique_count,max(loc_count(2,(2:unique_count))));
row_pointer=1;

t_offset=M/(2*fs);
t_hop=R/fs;

for i=1:unique_count
    pointer=1;
    disp(sprintf('working on %.1f frequency ...
        index',F(unique_v(i))));

    for j=1:nframes
        for k=1:max_peaks
```

```
            if peaks_matrix(2*j,k)==unique_v(i)

                good_matrix(2*row_pointer-1,pointer)=...
                peaks_matrix(2*j-1,k);
                good_matrix(2*row_pointer,pointer)=...
                (t_offset+(j-1)*t_hop);
                pointer=pointer+1;

                break
            end
        end
    end
    row_pointer=row_pointer+1;
end

fprintf('computed peaks and frequency matrix\n',row_pointer-1);

% fitting data with polyfit and calculating magnitude
fprintf('fitting data with polyfit\n',row_pointer-1);
row_pointer=1;

for i=1:10
    if loc_count(2,i) > 10
    figure(1)
    plot(good_matrix(2*i,1:loc_count(2,i)),...
    good_matrix(2*i-1,1:loc_count(2,i)));
    hold on
    end
end
hold off

for i=1:unique_count

    if loc_count(2,i) > 10
        [m,b]=polyfit(good_matrix(2*i,1:loc_count(2,i)),...
        good_matrix(2*i-1,1:loc_count(2,i)),1);
        if m(1) < 0
            G(1,row_pointer)=10^((m(1)*L)/(20*R));
            G(2,row_pointer)=F(loc_count(1,i));
            row_pointer=row_pointer+1;
        end
    end

end

fprintf('computed Gk for %f partials\n',row_pointer-1);
% G contains magnitude (1st row) and frequency (2nd row) ...
   computed along the
% kth partial (kth column)

% g is computed as a median value of three lowest partials

g0=G(1,1);

n_modes=size(G,2);
a=[-1:0.0001:1];
err=0;

% computing error

for i=1:length(a)

    for k=1:n_modes
        w=1/(1-G(1,k));
        filter_r=abs(g0*(1+a(i)))/sqrt(1+a(i)^2+...
        2*a(i)*cos(2*pi*G(2,k)*(1/fs)));
        err=err+w*(filter_r-G(1,k))^2;
    end

    error(i)=err;
    err=0;

end

% plotting error function
figure(2)
title('error function')
plot(a,error)
```

```
[Z,I] = min(error);
xlabel('a coefficient');
ylabel('Error');

% computing magnitude response

f=[0:G(2,n_modes)];
a1=a(I);
b0=g0*(1+a1);
for j=1:length(F)
mag(j)=abs(b0)./(sqrt(1+a1^2+2*a1*cos(2*pi*F(j)*(1/44100))));
end

% plotting freq response (loop gains at partial frequencies ...
    and magnitude
% response of filter design
figure(3)
title('loop filter design')
plot(G(2,1:n_modes),G(1,1:n_modes),'o',F,mag)
xlabel('Frequency');
ylabel('Gain');

% opening result file and append parameters g and a1
fid=fopen('LP_filter.dat','a');
fprintf(fid,'%f\t%f\t%.1f\t%f\t%f\r\n',string,fret,f0,g0,a1);
fclose(fid);
```

B.2 Coefficient FIT

```
fid=fopen('LP_filter.dat');
C = textscan(fid,'%f %f %f %f %f');
fclose(fid);

string = C{1};
fret = C{2};
freq = C{3};
g0 = C{4};
a1 = C{5};
```

```
for index=1:6

[g0m,g0b]=polyfit(fret(index*8-7:index*8),g0(index*8-7:index*8),2);
[a1m,a1b]=polyfit(fret(index*8-7:index*8),a1(index*8-7:index*8),2);

g0_fit=g0m(1)*fret(index*8-7:index*8).^2+...
g0m(2)*fret(index*8-7:index*8)+g0m(3);
a1_fit=a1m(1)*fret(index*8-7:index*8).^2+...
a1m(2)*fret(index*8-7:index*8)+a1m(3);

figure(2*index-1)
plot(fret(index*8-7:index*8),g0(index*8-7:index*8))
hold on
plot(fret(index*8-7:index*8),g0_fit,'--')
hold off
xlabel('Fret number');
ylabel('Loop filter coefficient g');

figure(2*index)

plot(fret(index*8-7:index*8),a1(index*8-7:index*8))
hold on
plot(fret(index*8-7:index*8),a1_fit,'--')
hold off
xlabel('Fret number');
ylabel('Loop filter coefficient a');

fid=fopen('coeff_fit.dat','a');
fprintf(fid,'%f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\r\n',...
index,g0m(1),g0m(2),g0m(3),a1m(1),a1m(2),a1m(3));
fclose(fid);

end
```

B.3 ffg Function: Fast Fourier Transform

```
function [mod,fase,freq] = ffg(gr,N,dt)

N = floor(N/2)*2;
grf = fft(gr,N);

mod(2:(N/2))  = 2/N*abs(grf(2:(N/2)));
mod(1)        = abs(grf(1))*2/N;

freq = (1/dt)/N*(0:(N/2-1));

fase(2:(N/2))  = atan2(imag(grf(2:(N/2))),real(grf(2:(N/2))));
fase           = fase*(180/pi);
```

Appendix C

EDR Comparisons

Energy decay relief comparisons for tones obtaining from sampling a Gibson SJ200 against tones synthesised by Max/MSP.

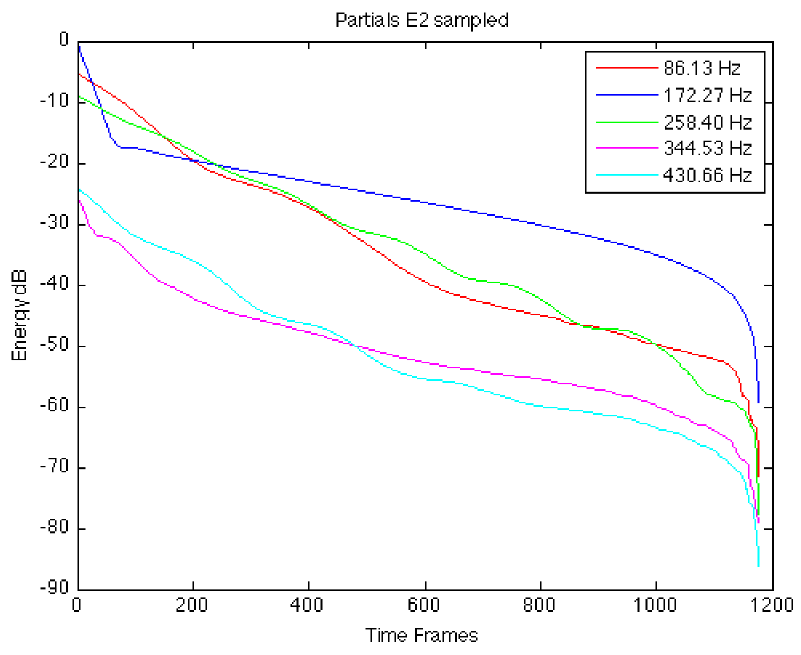


Figure C.1: *E2 open: sampled tone.*

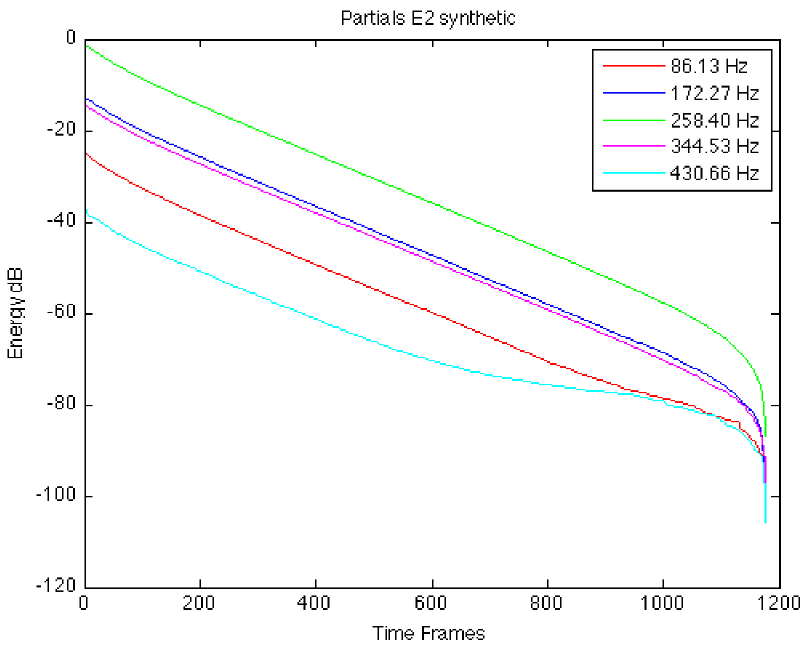


Figure C.2: *E2 open: synthetic tone.*

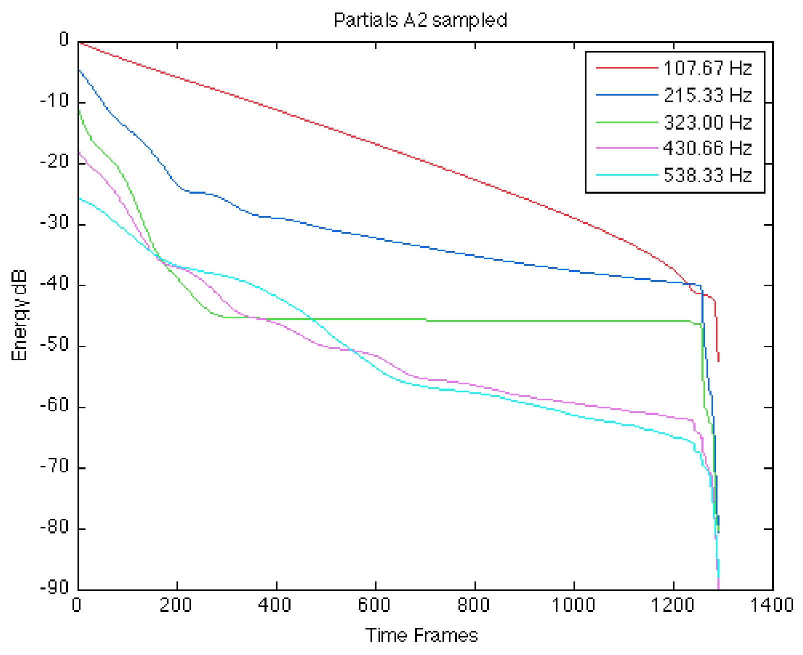


Figure C.3: *A2 open: sampled tone.*

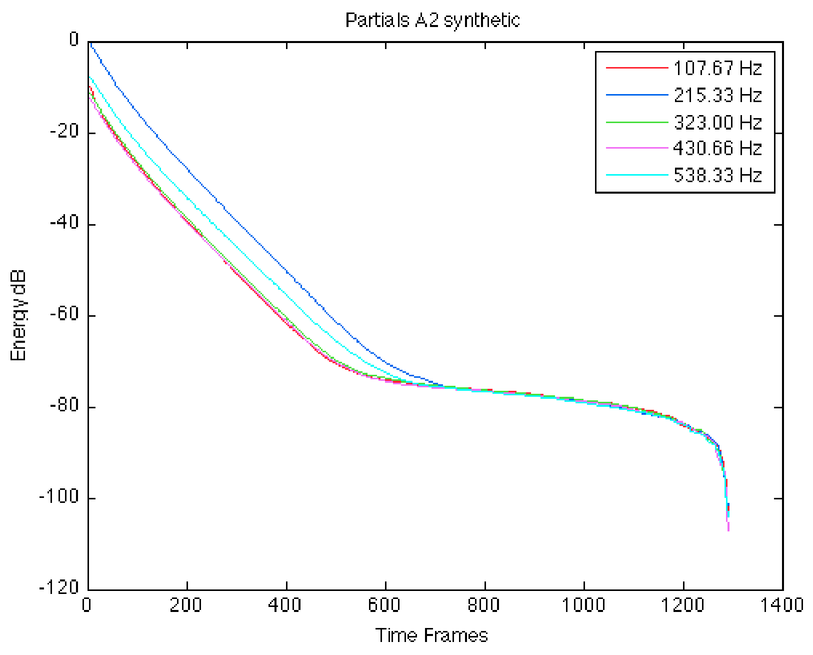


Figure C.4: *A2 open: synthetic tone.*

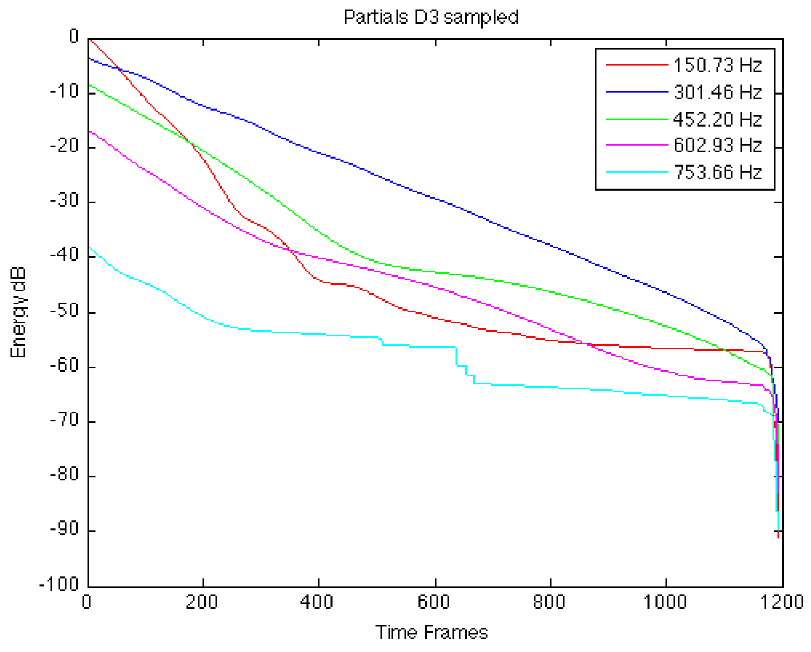


Figure C.5: *D3 open: sampled tone.*

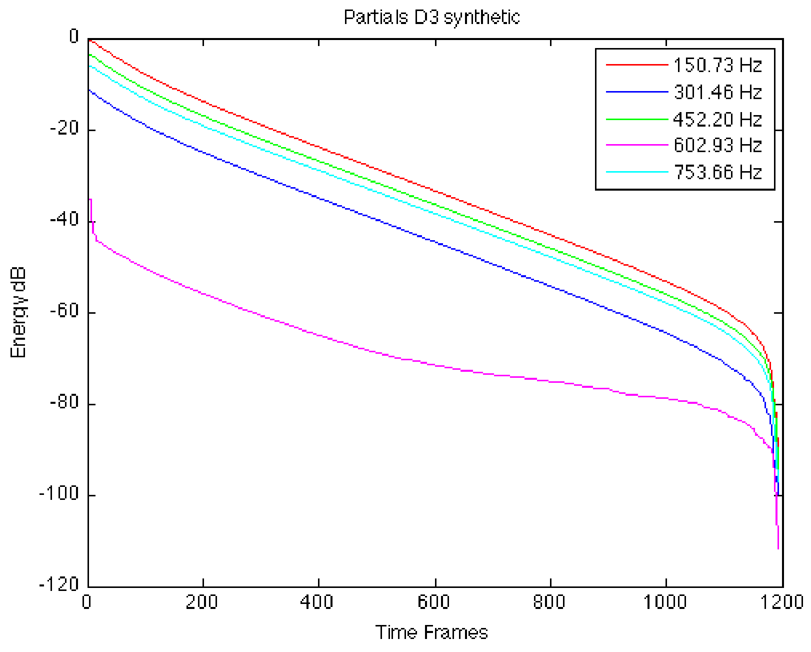


Figure C.6: *D3 open: synthetic tone.*

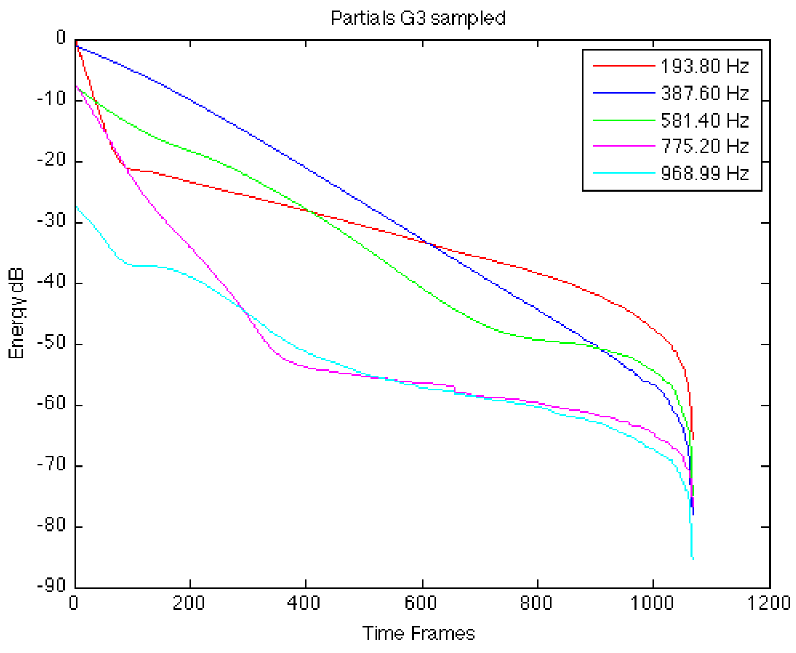


Figure C.7: *G3 open: sampled tone.*

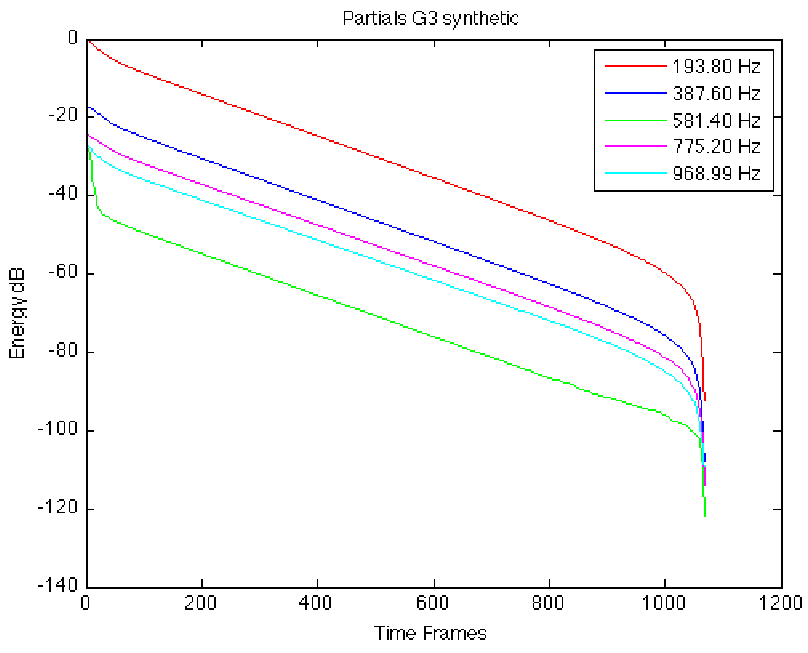


Figure C.8: *G3 open: synthetic tone.*

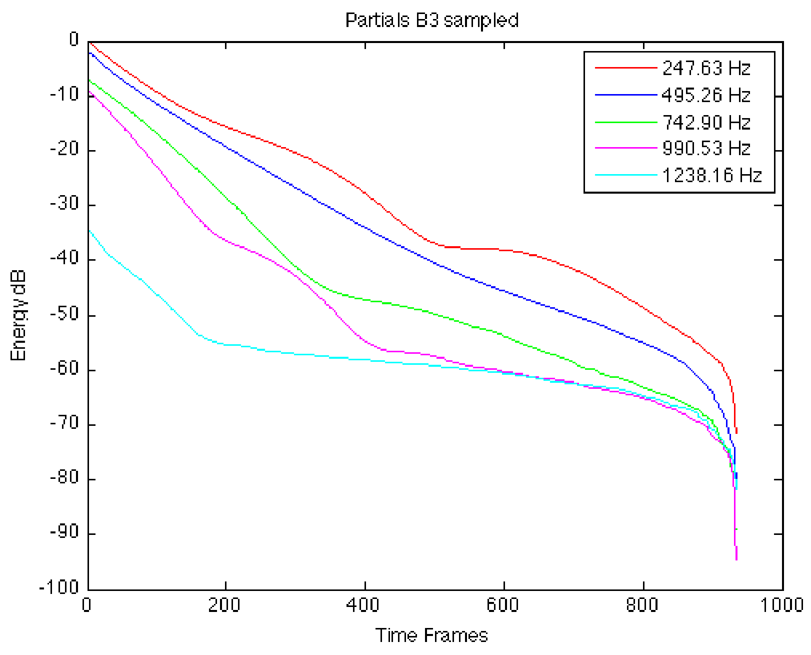


Figure C.9: *B3 open: sampled tone.*

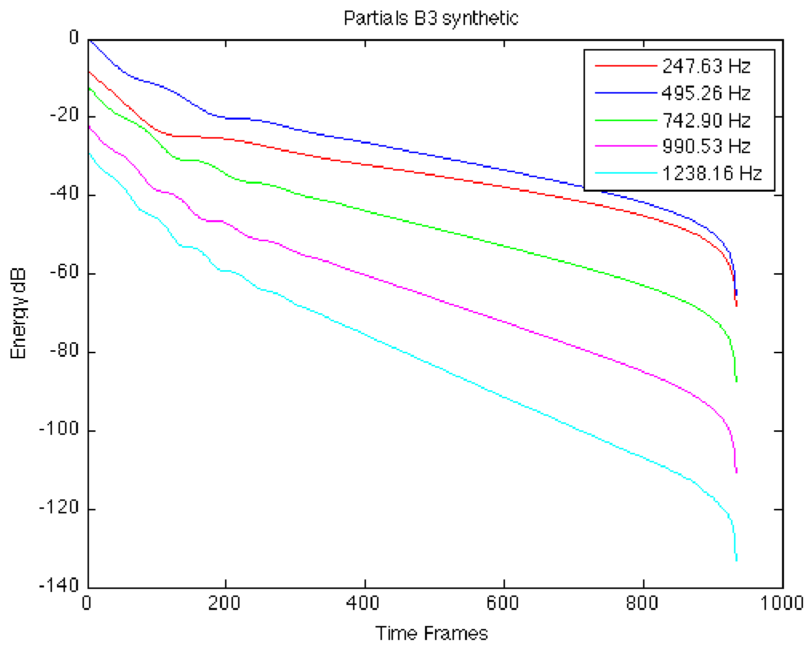


Figure C.10: *B3 open: synthetic tone.*

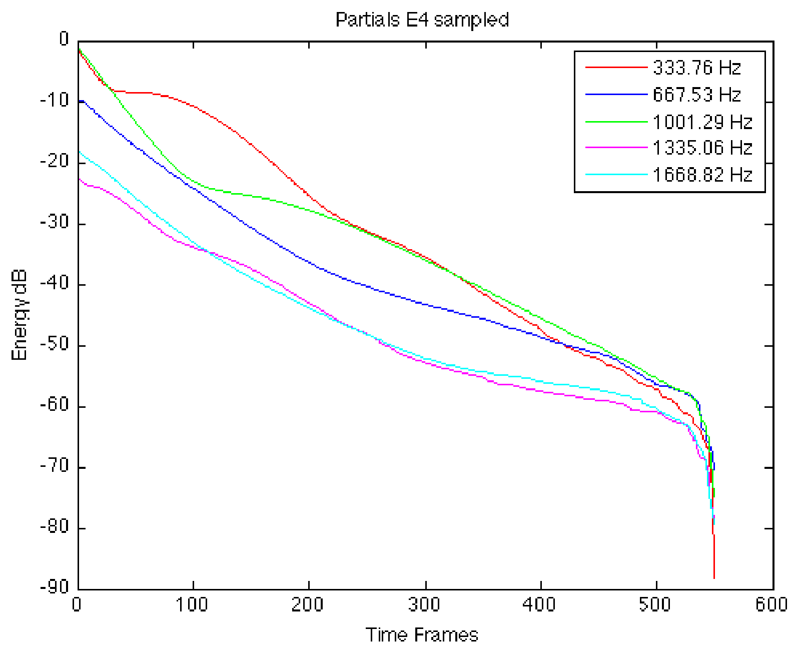


Figure C.11: *E4 open: sampled tone.*

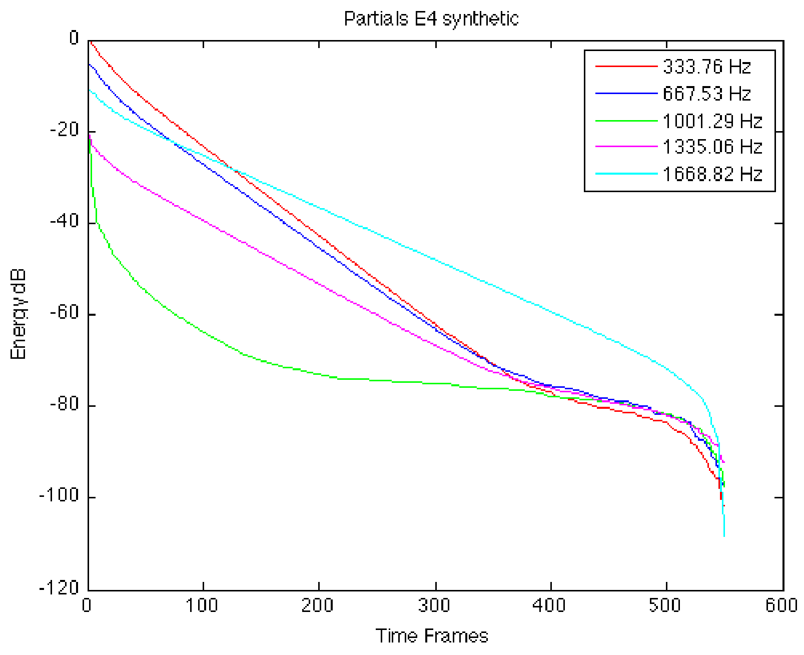


Figure C.12: *E4 open: synthetic tone.*

Bibliography

[1] Bank, B. (2006). *Physics-based Sound Synthesis of String Instruments Including Geometric Nonlinearities.* (PhD thesis), Budapest University of Technology and Economics, Budapest.

[2] Bank, B. (2008) *Passive admittance matrix synthesis for block-based modeling.* COST SID STSM ICT0601-3163 (Report), University of Verona, Verona.

[3] Cipriani, A. and Giri, M. (2009). *Musica Elettronica e Sound Design. Volume 1 e 2.* ConTempoNet s.a.s., Roma.

[4] Collecchia, R. (2012). *Numbers and Notes: An Introduction to Musical Signal Processing.* Perfectly Scientific Press, Portland.

[5] Cucuzzoli, G. and Lombardo, V. (1999). *A Physical Model of the Classical Guitar Including the Player's Touch.* Computer Music Journal, Summer 1999, pp. 52-69.

[6] Diana, G. and Cheli, F. (1999). *Dinamica e Vibrazione dei Sistemi Meccanici.* UTET Universita', Torino.

[7] Drumm, I. (2012). *Acoustics of Music, Course Notes.*. University of Salford, Salford.

[8] Erkut. C., Välimäki, V., Karjalainen, M. and Laurson, M. (1998). *Estimation of Physical and Expressive Parameters for Model-based Sound Synthesis of the Classical Guitar.* Computer Music Journal, Fall 1998, pp. 17-32.

[9] Karplus, K. and Strong, A. (1983). *Digital Synthesis of Plucked-String and Drum Timbres.* MIT Computer Music Journal, Vol.7 No.2, pp. 43-55.

[10] Jaffe, D.A. and Smith, J.O. (1983). *Extensions of the Karplus-Strong Plucked-String Algorithm.* MIT Computer Music Journal, Vol.7 No.2, pp. 56-69.

[11] Karjalainen, M. and Smith, J.O. (1996). *Body Modeling Techniques for String Instrument Synthesis.* ICMC Proceedings, 1996.

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

[13] Järveläinen, H. and Karjalainen, M. (2002). *Perception of Beating and Two-stage Decay in Dual Polarization String Models.* In Proceedings of the International Symposium on Musical Acoustics: 9-13 December 2002, Mexico City.

[14] Roads C. (1996). *The Computer Music Tutorial.* MIT Press, Boston.

[15] Roma, M., Gonzalez, L. and Briones, F., (2008). *Software Based Acoustic Guitar Simulation by means of its Impulse Response.* In Proceedings of 10th Meeting on Audio Engineering of the AES Portugal, Lisbon, December 12 and 13, 2008. Retrieved from http://www.deetc.isel.ipl.pt/aes2008/versao_02/contrib/aut/MiguelRoma.pdf.

[16] Fletcher, N.H. and Rossing, T.D., (1997). *The Physics of Musical Instruments.* Springer 2010, ISBN 978-1-4419-3120-7

[17] Smith, J.O. and Serra, X., (1987). *PARSHL: An Analysis/Synthesis Program for Non-Harmonic Sounds Based on a Sinusoidal Representation.* In Proceedings of the International Computer Music Conference (ICMC-87), Tokio. Retrieved from https://ccrma.stanford.edu/ jos/parshl.

[18] Smith, J.O. (2006). *Physical Modeling Sound Synthesis.* AES 2006 Masterclass. Retrieved from https://ccrma.stanford.edu/ jos/pdf/AES-Masterclass.pdf.

[19] Smith, J.O. (2007). *Introduction to Digital Filters with Audio Applications.* W3K Publishing, http://books.w3k.org/, 2007, ISBN 978-0-9745607-1-7.

[20] Smith, J.O. (2010). *Physical Audio Signal Processing for Virtual Musical Instruments and Digital Audio Effects.* W3K Publishing, http://books.w3k.org/, 2010, ISBN 978-0-9745607-2-4.

[21] Välimäki, V., Huopaniemi, J., Karjalainen, M. and Jánosy, Z. (1996). *Physical Modeling of Plucked String Instruments with Application to Real-time Sound Synthesis.* JAES, vol.44, no.5 pp. 331-353.

[22] Välimäki V. and Tolonen, T. (1997). *Development and Calibration of a Guitar Synthesizer.* Paper presented at AES 103rd Convention, New York, 26-29 September 1997.

[23] Weinreich G. (1977). *Coupled Piano Strings.* J. Acoust. Soc. Am., Vol.62, No.6, December 1977, pp. 1474-1484.