ICA 2016 BUENOS AIRES
Buenos Aires – 5 to 9 September 2016
Acoustics for the 21st Century...

PROCEEDINGS of the 22nd International Congress on Acoustics

Electroacoustics and Audio Engineering: Paper ICA2016-559

Plate reverberation: Towards the development of a real-time physical model for the working musician

Michele Ducceschi[(a)], **Craig J. Webb**[(b)]

[(a)]Acoustics and Audio Group, University of Edinburgh, UK, michele.ducceschi@ed.ac.uk
[(b)]Acoustics and Audio Group, University of Edinburgh, UK, craig.webb@ed.ac.uk

Abstract

Reverberation is an essential effect for sound design and music production, and commercially available software offers an unprecedented range of solutions for artists. Plate reverberation represents an attractive choice as the modal density of large plates is constant over the audible range, creating a uniform response in the frequency domain. However, available plug-ins rely on either sampled impulse responses or simple delay algorithms. In this paper, a pure physical model of a rectangular plate is used at the core of a real-time effect plug-in. The user has a choice of intuitive parameters to select the dimensions of the plate, the tension, and decay ratios. The input and output positions can be changed dynamically, during the runtime of the simulation. This represents a clear improvement over static algorithms based on impulse responses. Optimisation of the model using CPU vector intrinsics is demonstrated, allowing real-time computation of large-scale plate reverbs on consumer hardware using a standard plugin architecture.

Keywords: plate reverb, physical modelling, real-time, plugin

Plate reverberation: Towards the development of a real-time physical model for the working musician

EDIT JUNE 2019. CORRECTED SOME NOTATION AND TYPOS

1 Introduction

In recent years, physical modelling techniques have come to prominence in the realm of sound synthesis [3]. Other than recreating actual instruments, physical modelling can be used to simulate analog effect units, such as the plate reverb. This effect has been widely used since the 1950s, and continues to be used today. The appetite for software counterparts of this analog effect can be estimated by the growing number of impulse-response based plug-ins available for purchase, including for example WAVES [1], or EMT® 140 Classic Plate Reverberator Plug-In [8]. With respect to previous physical models making use of finite difference schemes [2, 4, 6], a modal approach is used here to solve the dynamic equations. This choice is beneficial in a number of ways, mainly in offering the possibility to easily implement frequency-dependent losses, as well as presenting a natural parallel structure. Reduction of the number of degrees of freedom is also possible, resulting in faster and cheaper simulations. The paper is organised as follows: Section 2 presents the plate equations, as well as the resolution in terms of the modes. Section 3 presents the numerical algorithm. Computational considerations are given in Section 4, and Section 5 presents the user interface design. Finally, Section 6 discusses a pedagogical example.

2 Model Equations

A model for a plate reverb unit can be derived by considering a flat, rectangular plate undergoing small vibrations. Mathematically, this can be expressed as the Kirchhoff plate equation with loss and tension, in the following way

$$\rho h \frac{\partial^2 w}{\partial t^2} = T_0 \Delta w - D \Delta \Delta w - 2c\rho h \frac{\partial w}{\partial t} + \delta(x-x_p)\delta(y-y_p)P(t). \tag{1}$$

In the equation, the flexural displacement of the plate is indicated by $w(x,y,t)$; the symbol Δ denotes the Laplace operator in Cartesian coordinates. The plate occupies a rectangular region of space, here called $\mathscr{D} \triangleq [0,L_x] \times [0,L_y]$, and whose boundary will be denoted by Γ. Constants appear as: volumetric density ρ, thickness h, tension T_0, flexural rigidity $D \triangleq Eh^3/12(1-\nu^2)$, where E is Young's modulus and ν is Poisson's ratio; c represents a loss parameter (and its form will actually be cast in a frequency-dependent form when the modal approach is described in Section 2.2.) Finally, $P(t)$ represents the input audio at the transducer location (x_p,y_p).

2.1 Energy and boundary conditions

In the case of the plate with no loss and forcing, a standard energy analysis leads to the following energy balance for the plate [5]

$$\frac{d}{dt}\left[\underbrace{\frac{\rho h}{2}\left\|\frac{\partial w}{\partial t}\right\|_{\mathscr{D}}^2+\frac{T_0}{2}\|\nabla w\|_{\mathscr{D}}^2+\frac{D}{2}\|\Delta w\|_{\mathscr{D}}^2}_{\mathfrak{H}}\right]=T_0\oint_\Gamma\frac{\partial w}{\partial t}\nabla w\cdot\mathrm{d}\Gamma-D\oint_\Gamma\frac{\partial w}{\partial t}\nabla\Delta w\cdot\mathrm{d}\Gamma+D\oint_\Gamma\Delta w\nabla\frac{\partial w}{\partial t}\cdot\mathrm{d}\Gamma, \quad (2)$$

where the notation of norm for a well-behaved (vector) function $g(x,y)$ was introduced as

$$\|g\|_{\mathscr{D}}^2\triangleq\int_{\mathscr{D}}g\cdot g\,\mathrm{d}\mathscr{D} \quad (3)$$

The dot notation indicates the euclidean product of two vectors. (Notice that, with a slight abuse of notation, the norm here is used interchangeably when g is a either a vector or a scalar; in the latter case, the dot product reduces to multiplication of two scalar functions). In the boundary terms, Γ represents the outwardly-oriented boundary (a vector). When the boundary terms vanish, one is left with

$$\frac{d}{dt}\mathfrak{H}=0\quad\rightarrow\quad\mathfrak{H}=\mathfrak{H}_0, \quad (4)$$

which gives energy conservation for the total energy. The total energy is clearly non-negative, being the sum of non-negative quantities, and thus one may write

$$0\leqslant\left\|\frac{\partial w}{\partial t}\right\|_{\mathscr{D}}\leqslant\sqrt{\frac{2\mathfrak{H}_0}{\rho h}}, \quad (5)$$

which is a bound on the growth of the solution.

2.2 Modes

In general, one can attempt to find a solution by decomposing w onto a series of modes, in the following way [7]

$$w=\sum_{m=1}^{M}\Phi_m(x,y)A_m\sin(\omega_m t+\zeta_m) \quad (6)$$

where M is, in theory, infinite, but it is understood to be a finite integer for any practical application. The expansion above is valid is valid in the lossless, unforced case. The modal phases ζ_m and modal amplitudes A_m may be determined from initial conditions. The modal shapes $\Phi_m(x,y)$ must satisfy the prescribed boundary conditions, and they also have to form a complete, orthogonal set over $\mathscr{D}$ (in the sense of L_2 function spaces.) It is also assumed that $\|\Phi_m\|_{\mathscr{D}}=1$, $\forall m$. When (6) is substituted into (1) (with $c=P=0$), one obtains

$$\omega_m^2\Phi_m=\frac{D}{\rho h}\Delta\Delta\Phi_m-\frac{T_0}{\rho h}\Delta\Phi_m,\quad\forall\, m\in[1,M]. \quad (7)$$

This equation is the eigenvalue problem that must be solved in order to find the eigenfrequencies ω_m. In order to do so, both sides are multiplied by Φ_n, and an inner product is taken. Because of the assumption that the modes are orthogonal, and because they satisfy the boundary conditions, one gets (after integration by parts)

$$\omega_m^2 \|\Phi_m\|_{\mathscr{D}}^2 = \frac{T_0}{\rho h} \|\nabla \Phi_m\|_{\mathscr{D}}^2 + \frac{D}{\rho h} \|\Delta \Phi_m\|_{\mathscr{D}}^2 . \tag{8}$$

When loss and forcing are present in the system, the assumption that the time component oscillates sinusoidally is not any more valid. Hence, (6) must be modified in the following way

$$w = \sum_{m=1}^{M} q_m \Phi_m(x,y), \tag{9}$$

for unknown q_m. This is now substituted back into (1), and an inner product is taken with Φ_n on both sides. Recalling (7), one gets

$$\ddot{q}_m + \omega_m^2 q_m + 2c_m \dot{q}_m - \frac{\Phi_m(x_p,y_p)}{\rho h} P(t) = 0, \quad \forall m \in [0,M]. \tag{10}$$

where the dot notation is used to indicate the time derivative (the q_m's are functions of time only.) This is a system of uncoupled harmonic oscillators, with loss and forcing. Notice that the loss coefficients c_m can now be set mode by mode, and not just globally like in (1). Finally, notice that, by virtue of (8), and by modal orthogonality arguments, the total energy in (2) can be written as

$$\mathfrak{H} = \sum_{m=1}^{M} \mathfrak{H}_m = \sum_{m=1}^{M} \frac{\rho h}{2} \left[\left(\frac{\partial q_m}{\partial t} \right)^2 + \omega_m^2 q_m^2 \right], \tag{11}$$

and hence the energy is the sum of independent contributions coming from each one of the modes.

3 Numerical Solution

In the previous section, the modal decomposition was developed in a general form. So long as one is able to obtain the modal shapes Φ_m, the eigenfrequencies can be calculated by means of (8), and then the time evolution of the modal coordinates can be calculated from (10). In general, a closed form solution for Φ_m is not available, but a few cases represent an exception. Consider simply-supported boundary conditions, $w = \Delta w = 0$ on Γ. For such conditions, the functions

$$\Phi_m(x,y) = \sqrt{\frac{4}{L_x L_y}} \sin \frac{m_1 \pi x}{L_x} \sin \frac{m_2 \pi y}{L_y}, \quad (m_1,m_2) \in \mathbb{Z}^+ \tag{12}$$

satisfy the requirements of completeness and orthogonality (the scalar multiplying the sine functions is such that $\|\Phi_m\|_{\mathscr{D}} = 1 \ \forall m$.). Inserting expression (12) into (8), one gets

$$\omega_m = \sqrt{\frac{T_0}{\rho h} \left(\frac{m_1^2 \pi^2}{L_x^2} + \frac{m_2^2 \pi^2}{L_y^2} \right) + \frac{D}{\rho h} \left(\frac{m_1^2 \pi^2}{L_x^2} + \frac{m_2^2 \pi^2}{L_y^2} \right)^2}, \quad (m_1,m_2) \in \mathbb{Z}^+ \tag{13}$$

3.1 Semi-discrete problem

In order to solve (10), a suitable time stepping scheme is obtained in the following way. Time is evaluated at discrete, equally-spaced intervals by means of a sampling rate $f_s \triangleq 1/k$, where k is called the time step. The modal coordinates q_m are now evaluated at the discrete times kr, with $r \in \mathbb{Z}^+$. The derivative operators are then substituted with the following difference operators (although different choices are available [3])

$$\ddot{q}_m(t) \to \frac{q_m(k(r+1)) - 2q_m(kr) + q_m(k(r-1))}{k^2}, \quad \dot{q}_m(t) \to \frac{q_m(k(r+1)) - q_m(k(r-1))}{2k}. \tag{14}$$

Substituting (14) into (10) gives the following update equation

$$\left[\frac{1}{k^2} + \frac{c_m}{k}\right] q_m(k(r+1)) = \left[\frac{2}{k^2} - \omega_m^2\right] q_m(kr) + \left[\frac{c_m}{k} - \frac{1}{k^2}\right] q_m(k(r-1)) + \frac{\Phi_m(x_p, y_p)}{\rho h} P(kr). \tag{15}$$

The discrete energy is

$$\mathfrak{h} = \sum_{m=1}^{M} \mathfrak{h}_m = \sum_{m=1}^{M} \frac{\rho h}{2} \left(\frac{(q_m(kr) - q_m(k(r-1)))^2}{k^2} + \omega_m^2 q_m(kr) q_m(k(r-1)) \right) \tag{16}$$

which is a discrete counterpart of (11). Notice that here, as opposed to the continuous case, the total energy is of undetermined sign. In order to guarantee non-negativity of each of the modal energies $\mathfrak{h}_m$ (and hence of the energy as a whole) one must impose [3]

$$\omega_m < \frac{2}{k}. \tag{17}$$

This should be regarded as a stability condition for scheme (15): if such condition is enforced, one may bound the growth of the discrete solution, in an analogous way as (5).

Hence, fixing the sampling rate poses an upper limit on M, the total number of modes. Notice that the scheme is fully explicit, and therefore possesses a natural parallel structure.

3.2 Dynamic output

Extracting the output is, in some sense, inverting the projection operation that leads to the modal equations. In a basic configuration, a plate reverb unit has two pickups which record the displacement of the plate. Let a given pickup be placed at (x_o, y_o). Owing to (9), one has that the displacement at the output point is given by

$$w_o = \sum_{m=1}^{M} q_m \Phi_m(x_o, y_o). \tag{18}$$

As opposed to impulse-response based plugins, the output point can be rendered dynamically in a very easy way, by considering (x_o, y_o) to be time-dependent. This leads to a natural phase movement with complex auditory cues, see also section 6.

4 Computational Testing

The algorithm for computing the plate system consists of two main stages. First, a pre-time iteration stage that builds vectors based on the modal frequencies. Second, the core time iteration stage that calculates the displacement and takes the output. There is some cross-over between these two stages, depending on the level of dynamic behaviour that is required. For example, if the input position is made fully dynamic (being computed during run-time, instead of resetting the system) then elements of the pre-time iteration stage need to be moved into the time iteration itself. For the purpose of initial testing, both the input and output positions were treated as static to judge the base level of computation required for the real-time system.

The standard dimensions of the plate used for testing were 2m by 1m, with a thickness of 0.5mm. This leads to a system that uses 10,000 modal frequencies in its basic form. The system was written in C++ and two versions were tested; one a standard code and a second that used manual AVX intrinsics to vectorise the calculation of the displacement. Only single-threaded code was considered at this time; multi-threaded usage will be explored at a later date due the complications involved with spawning multiple threads inside of a real-time plug-in environment. The codes were tested on two Intel processors; a 2.2GHz Core i7 “Sandy Bridge” and a 3.7GHz Xeon “Ivy Bridge-EP”. The results in table 2 show the time taken to compute 1 second of sound at 44.1kHz. The standard code was tested using both -O0 and -O3 compiler optimisation flags, and all codes were tested at both single and double precision floating-point.

	i7-Double	i7-Single	Xeon-Double	Xeon-Single
Standard - O0	3.2	3.1	2.5	2.4
Standard - O3	1.1	0.7	0.8	0.5
AVX - O3	1.1	0.7	0.8	0.5

Table 1: Timing results for a 10,000 mode scheme, at single and double precision. Values in seconds.

The results clearly show the benefit of vectorization, with AVX intrinsics working on 4 double or 8 single precision variables simultaneously. The straightforward nature of the main displacement calculation allows the compiler to apply AVX instructions automatically when using –O3 optimisation, a useful benefit. Whilst the timing results show that both the Xeon and i7 processors are capable of computing the simulation in under 1 second (although only at single precision for the i7), this does not necessarily translate to real-time performance within a plug-in environment. Initial testing in an Audio Unit plug-in showed that the single precision simulation running on the i7 would lead to incomplete buffers, drop-outs in the output audio. A slightly reduced simulation size that takes 0.6 seconds to compute in stand-alone code was found to be the maximum simulation size capable of running without drop-outs on the older i7 processor.

4.1 Reduction of the number of degrees of freedom

For a typical plate unit, at audio rate, the number of modes is of the order of $10,000$. The range $20-20,000$Hz spans approximately eleven octave bands, but because the modes are more or less evenly spaced along the frequency axis, the last 3 octave bands are highly populated (for the 2m by 1m by 0.5mm plate, they contain about 80% of the total modes.) According to (10), each one of the modes vibrates indipendently of the others. Thus, one may remove a number of the high-frequency modes without impacting the quality of the output. A possible rule to decide whether a mode is kept or discarded can be given in terms of perceptually meaningful parameters: fix a distance between two frequencies, $ncent$ (measured in cents): take the first mode, at f_1, and discard all the modes whose distance from f_1 is less than $ncent$. Once the new frequency f_i is obtained, repeat the procedure, and so on. This rule is particularly useful when $ncent$ is frequency dependent (so it can be made larger for high-frequency bands.) In general, a careful selection of $ncent$ can reduce the number of modes from the original 10^4 to the range of $4,000$ to $5,000$, resulting in substantially less CPU usage.

5 Design and Implementation of the Plug-In

The plate reverb system was implemented as an Audio Unit version 2 plug-in for OSX. The implementation was created by directly sub-classing the core audio framework, specifically the AUBase class rather than the standard AUEffectBase approach that produces a kernel for each audio channel (something that is definitely not desirable in this case for obvious computational reasons). A custom view was then designed and created using the Cocoa framework and Objective-C. The central design approach for the user interface was to provide an intuitive object-based view. The system allows the user to re-size the plate, move the input forcing positions and output positions to any point on the plate, and also specify the delay time across 8 frequency bands. By providing a graphical representation of these elements, we can minimise the number of slider controls. Figure 1 shows the final design of the interface, as displayed in Logic Pro X.

The plate size is adjusted using the “grab” handle in the top-right of the plate, with the input and output discs keeping their relative positions. The discs with cross-hairs are the two input forcing positions; for a mono audio input both positions receive the same values, whilst for a stereo input the left and right channels are separated. The remaining discs are the left and right channel output positions. The frequency-dependent decay parameters are adjusted by varying the height of the bars in the chart view to the right. The centre frequency for each band is shown on the overlaid information sheet (Figure 2). With the reduced mode count, it is possible to use a maximum plate size of 3m by 2m, at 0.5mm thickness, whilst still allowing sufficient CPU capacity to dynamically compute the input and output positions during runtime.

6 Examples

In this section some of the features of the plug-in are presented. The reference plate is a steel plate of dimensions 2m by 1m by 0.5mm, Young’s modulus $E = 2 \cdot 10^{11}$Pa, Poisson’s

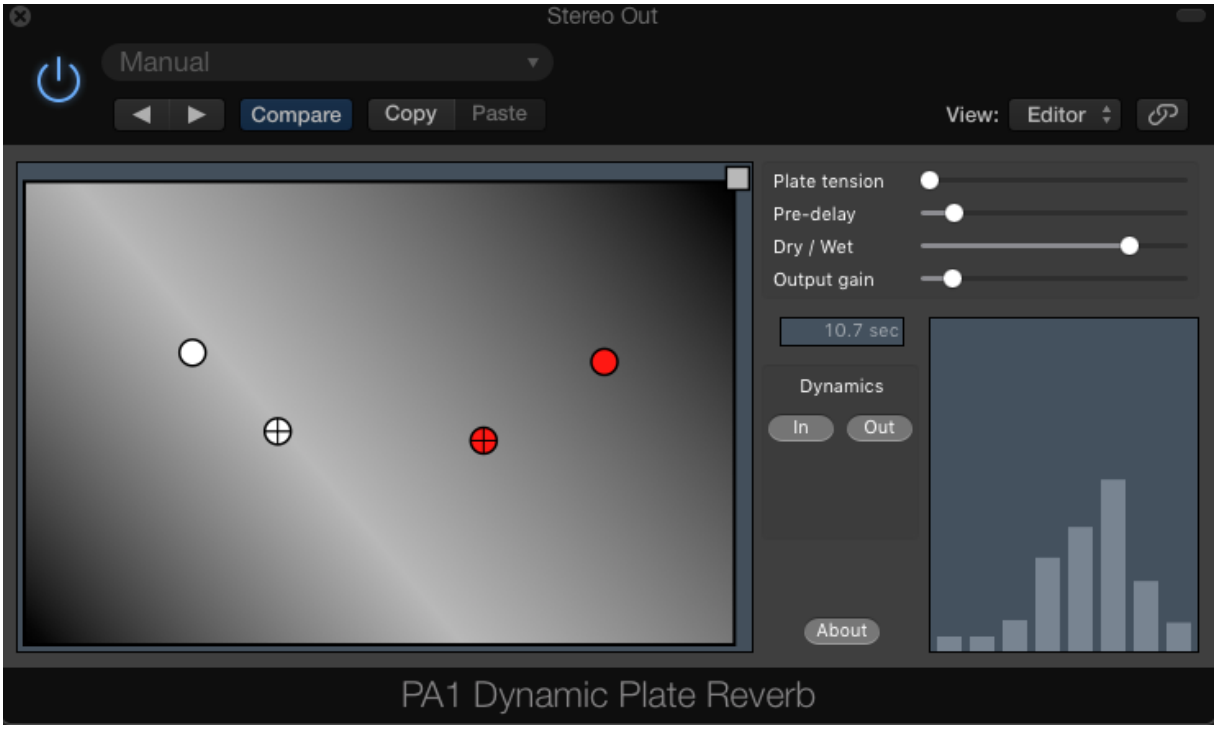

Figure 1: Graphical user interface design, as displayed in Logic Pro X.

ration $\nu = 0.3$, tension 600N. The damping coefficients have been selected in order to cover 8 frequency bands, so that the modes belonging to the same band share the same damping coefficient. Also, the damping coefficients are set by means of a typical decay time, called T_{60}, which is the time needed to decay by 60 dB. The relation between the time and the loss coefficients is

$$c = 6\frac{\ln(10)}{T_{60}} \tag{19}$$

The coefficients for the bands shown in Figure 2 have been selected as per Table 2.

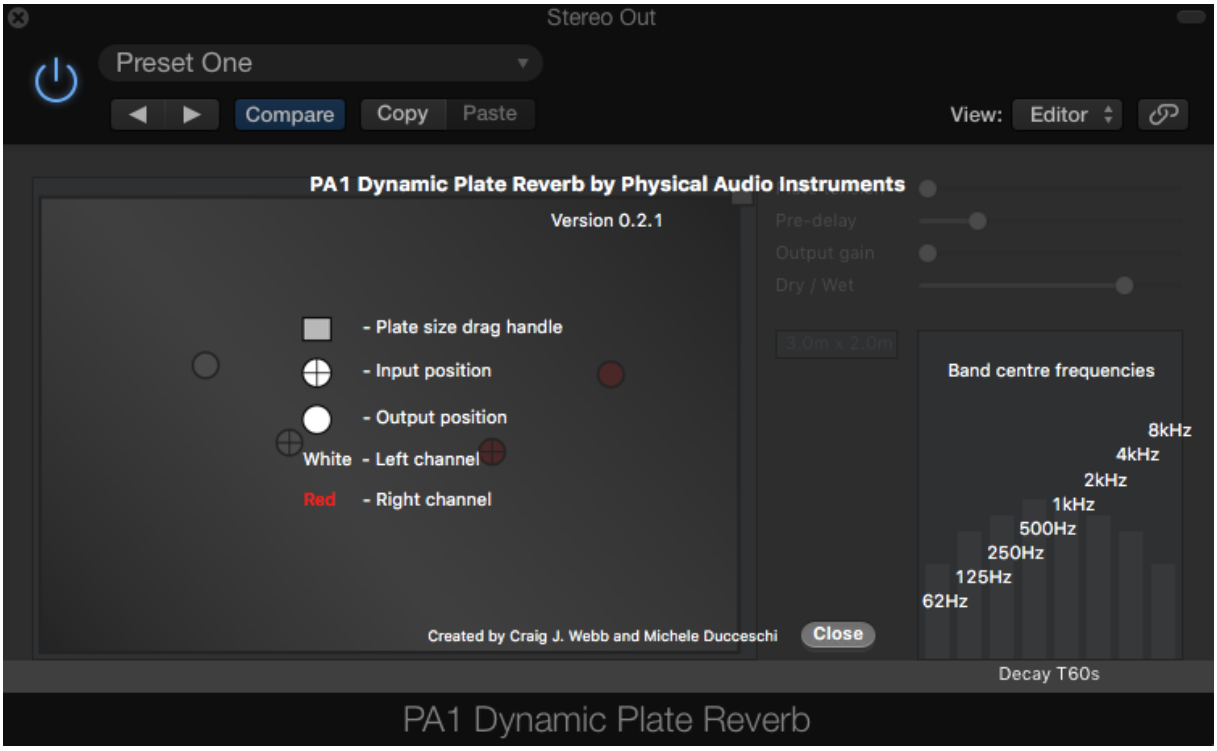

Figure 2: Information sheet, as displayed in Logic Pro X.

Hz	62	125	250	500	1000	2000	4000	8000
sec	8	7	8	6	5	6	3	2

Table 2: Frequency bands (top row) and T_{60} (bottom row) for the case under study.

Figure 3 presents the spectrograms of the input audio (i.e. with the plug-in bypassed) and two "wet" cases, with static and dynamic outputs. The input audio is a recording of female vocalist performing a sustained note. It is difficult to judge the quality of the sound by just looking at the spectrogram. Notice, however, that the phase movement is evident in Figure 3(c). The interested reader may find this and more examples at the following link: www.physicalaudio.co.uk.

7 Conclusions

This work presented the development of a software plug-in for simulating plate reverberation. Unlike standard software implementations based upon impulse responses, the user has control over a set of physical parameters that define the system. Particularly attractive features include frequency-dependent decay times over eight octave bands, the selection of plate size, and dynamic movement of the input forcing and output pickup positions. These allow far greater experimental control over the resulting effect, allowing the user to tune the reverberation in a

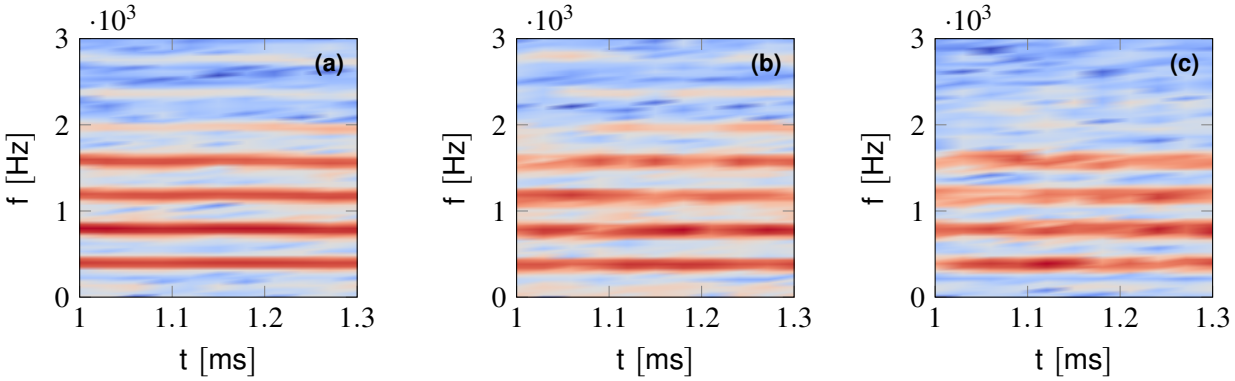

Figure 3: Spectrograms of output files. (a): dry sound (bypassed plugin). (b): wet sound (static). (c) wet sound (dynamic output). The plate parameters are detailed in Section 6. The input transducer was placed at $[0.52L_x, 0.53L_y]$ and the output pickup was placed at $[0.47Lx, 0.62L_y]$. For the dynamic case, the pickup was given a velocity of 5m/s, and was set to scatter off at equal angles at the edges.

creative manner.

Acknowledgements

Dr Ducceschi's research was supported by the Royal Society and the British Academy, through a Newton International Fellowship.

References

[1] Abbey Road Studios. WAVES [Computer Software], 2015.

[2] S. Bilbao. "A Digital Plate Reverberation Algorithm". *Journal of the Audio Engineering Society*, 55(3):135–144, 2007.

[3] S. Bilbao. *Numerical Sound synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*. Wiley, Chichester UK, 2009.

[4] S. Bilbao, K. Arcas and A. Chaigne. "A Physical Model of Plate Reverberation". In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Toulouse, France, 2006.

[5] M. Ducceschi and C. Touzé. "Modal approach for nonlinear vibrations of damped impacted plates: application to sound synthesis of gongs and cymbals". *Journal of Sound and Vibration*, 334:313–331, 2015.

[6] B. Hamilton and A. Torin. "Finite Difference Schemes on Hexagonal Grids for Thin Linear Plates with Finite Volume Boundaries". In *Proceedings of the 17th International Conference on Digital Audio Effects (DAFx)*, Erlangen, Germany, 2014.

[7] L. Meirovitch. *Fundamentals of Vibrations*. Waveland PR Inc, Long Grove, USA, 2010.

[8] Universal Audio. EMT® 140 Classic Plate Reverberator Plug-In [Computer Software], 2015.