REAL-TIME IMPLEMENTATION OF NON-LINEAR PHYSICAL MODELS WITH MODAL SYNTHESIS AND PERFORMANCE ANALYSIS

Alessandro CERIOLI (alexcerio92@gmail.com) (acerio19@student.aau.dk) [1],
Michele DUCCESCHI (michele.ducceschi@unibo.it) [2], and **Stefania SERAFIN** (sts@create.aau.dk) [1]

[1] **Aalborg University**, Copenhagen, Denmark
[2] **University of Bologna**, Bologna, Italy

ABSTRACT

Modal decomposition is a popular analysis approach involving the description of a target system via a bank of resonant oscillators called modes. Early sound synthesis frameworks successfully exploited this idea for the simulation of vibrating objects such as bars, plates and strings. While popular, modal synthesis is often applied to linear systems, since the modes become densely coupled in systems presenting distributed or multiple nonlinearities. In this work, the modal approach is used for the simulation of nonlinearly connected systems. When the nonlinearity is of cubic type, a suitable energy-stable modal update can be derived requiring the solution of a single linear system at each time step. A working plugin written in the C++ programming language is presented. Moreover, the performance of the plugin is analysed considering systems of different dimensions, defining the current limits for a real-time application of these models. The analysis also revealed a linear correlation between the number of modes which compose the systems and the CPU usage necessary for their real-time computation.

1. INTRODUCTION

The analysis of linear, time-invariant systems by superposition of modes is a longstanding idea, tracing back to the early works by Daniel Bernoulli on the vibrating string [1], and later formalised by Fourier [2]. Time-invariance and linearity allow to describe systems in terms of eigenfunctions and frequencies, called the *modes* of the system [3,4]. Such modal shapes and frequencies may be either determined experimentally, or starting from a suitable mathematical model. In the latter case, the model is in the form of a system of partial difference equations (PDEs), depending on material and geometric properties, type of the excitation, initial and boundary conditions [5]. Modal equations result after an appropriate projection is applied to the system of PDEs, yielding an eigenvalue problem, from which the modal frequencies and shapes are determined. The resulting modal equations depend exclusively on time, and output may be extracted as a suitable combination of the time-dependent modal coordinates. Usually, one is interested in computing a physical output at one or more points of the system, via a weighted sum resulting from an inverse projection.

Other than being a useful analysis tool, this approach lends itself naturally to the simulation of mechanical vibrations, and thus to sound synthesis via physics-based modelling. Modal synthesis began in earnest in the 1990s, when frameworks such as Mosaic [6] and Modalys [7] emerged. The early success of modal synthesis was partly due to the ease of implementation, and efficiency, of the modal structure: the orthogonality of the modes yields a bank of parallel damped oscillators. The inclusion of complicated loss profiles (necessary for realistic sound synthesis) is also trivial and inexpensive within the modal framework, as is the fine-tuning of the system's resonances, see e.g. [8].

In direct numerical simulation, such as finite differences, distributed nonlinearities can be resolved locally, and in some cases efficiently, via linearly-implicit schemes [9,10]. For the modal approach, the presence of nonlinearities, either lumped or distributed, may become problematic, since a coupling takes place between the modes of the associated linear system [11].

In this work, an extension of the modal approach, including nonlinearly coupled subsystems, is presented. In the work by Bilbao and Webb, using finite differences [12], simple objects such as bars and plates are connected nonlinearly using springs of cubic type. The same systems and connections are used here. The results are directly implemented in a working plugin, whose performance is tested.

The article is structured as follows. In Sec. 2, the continuous models are given, along with an energy analysis. Modal expansions are derived for the systems in isolation, as well as for two coupled subsystems. A suitable energy-passive finite difference scheme is offered. In Sec. 3 the C++ implementation is discussed, along with the plugin architecture, and in Sec. 4 the performance of the plugin is assessed, for increasing system sizes.

2. IMPLEMENTED MODELS

In this section, the mathematical formalism is introduced, along with notation. The modal approach will be developed starting from a system in isolation, and then extended to include nonlinearly connected objects.

2.1 Isolated Systems

Consider an isolated system, either a bar, a string or a plate, described by a PDE of the form

$$\left(m\partial_t^2 + \mathcal{L}\right) u(t,\mathbf{x}) = 0. \tag{1}$$

In this equation, $u(t,\mathbf{x})$ is the displacement of the vibrating object, depending on time t, as well as on the spatial coordinate $\mathbf{x} \in D = \Pi_{s=1}^{a}[0, L_s]$, where $a \in \{1, 2\}$, and where L_s is the side length. In practice, D is either a one- or a two-dimensional rectangular domain. Here, $\mathcal{L}$ is a linear, time-invariant spatial differential operator, obtained from a suitable physical model of the target object. For the systems considered here, one has

$$\mathcal{L} = \sum_{j=1}^{2} (-1)^j \alpha_j \Delta^j, \tag{2}$$

where Δ is the a-dimensional Laplacian, and where $\alpha_j \geq 0$ are coefficients. For the moment, we may assume that losses are not included in the system, such that in (1) energy remains constant. Finally, m specifies the mass per unit length[a], and the symbol ∂_t^p indicates the pth partial time derivative.

In order to fully specify the motion, (1) must be completed by suitable initial and boundary conditions. Since the problem is second-order in time, initial conditions are usually given as

$$u(0,\mathbf{x}) = u_0(\mathbf{x}), \quad \partial_t u(0,\mathbf{x}) = v_0(\mathbf{x}). \tag{3}$$

Boundary conditions may be harder to get to. As a useful guiding principle, an appropriate energy analysis leads in most cases to the correct expressions for the boundary conditions. For the two (either vector or scalar) square-integrable functions $\mathbf{f}$, $\mathbf{g}$, the inner product and norm are introduced as [13]

$$\langle \mathbf{f}, \mathbf{g} \rangle = \int_D \mathbf{f} \cdot \mathbf{g} \, \mathrm{d}\mathbf{x}, \quad \|\mathbf{f}\| = \sqrt{\langle \mathbf{f}, \mathbf{f} \rangle}, \tag{4}$$

where the dot operation is the Euclidian scalar product (reducing to the common multiplication when the functions are scalar). Taking the inner product of (1) with $\partial_t u$, after appropriate integration by parts, results in

$$\frac{d}{dt} H = B, \tag{5}$$

where H is the total energy of the system, and where B are boundary terms. Explicit expressions for the boundary terms are given as [1]

$$B = \oint_\Gamma \partial_t u \, (\alpha_1 \nabla u - \alpha_2 \nabla \Delta u) \cdot \mathrm{d}\mathbf{\Gamma} + \oint_\Gamma \alpha_2 \Delta u \nabla(\partial_t u) \cdot \mathrm{d}\mathbf{\Gamma},$$

where $\mathbf{\Gamma}$ is the outwardly oriented contour of the domain D. Among the possible combinations of boundary conditions, the simply-supported type lead to a closed-form solution for the modes, as will be seen shortly. Such conditions are given as

$$u = \Delta u = 0, \text{ along } \mathbf{\Gamma}. \tag{6}$$

Energy conservation is achieved under such choice. One therefore has

$$H(t) = H(t=0) \triangleq H_0, \tag{7}$$

where the form of the total energy depends exclusively on the initial conditions (3). The energy is expressed as the sum of kinetic and potential energy components, as

$$H = \frac{m \left\|\partial_t u\right\|^2}{2} + \frac{\alpha_1 \left\|\nabla u\right\|^2}{2} + \frac{\alpha_2 \left\|\Delta u\right\|^2}{2}. \tag{8}$$

From such energy conservation, it results that

$$\|\partial_t u\| \leq \sqrt{2H_0/m}, \tag{9}$$

and hence the norm of the velocity remains bounded over time.

2.1.1 Model Coefficients

Two subsystems are considered here: the string/bar, and the plate. For the string/bar, one has

$$a = 1, \, m = \rho A, \, \alpha_1 = T, \;\; \alpha_2 = EI, \tag{10}$$

whereas for the plate, one has

$$a = 2, \, m = \rho h, \, \alpha_1 = 0, \;\; \alpha_2 = D. \tag{11}$$

In the above, ρ is the volume density, A is the area of the cross section, E is Young's modulus, I is the area moment of inertia, h is the thickness, and D is the flexural rigidity.

2.2 Modal Decomposition

A modal decomposition is assumed for $u(t,\mathbf{x})$. This is

$$u(t,\mathbf{x}) = \mathbf{X}^\intercal(\mathbf{x})\mathbf{q}(t), \tag{12}$$

where both $\mathbf{X}$, $\mathbf{q}$ are column vectors of length N. In theory, N is infinite, though it will be truncated to an integer according to Nyquist-like requirements, as detailed below. Since simply-supported conditions over a rectangular domain are assumed, one has

$$X_n(\mathbf{x}) = \Pi_{s=1}^{a} \sin\left(\frac{n_s \pi x_s}{L_s}\right), \; n = 1, ..., N. \tag{13}$$

Inserting (12) into (1), multiplying on the left by $\mathbf{X}$, and integrating, gives

$$m\left(\int_D \mathbf{X}\,\mathbf{X}^\intercal \, \mathrm{d}\mathbf{x}\right)\ddot{\mathbf{q}} + \left(\int_D \mathbf{X}\mathcal{L}\mathbf{X}^\intercal \, \mathrm{d}\mathbf{x}\right)\mathbf{q} = 0. \tag{14}$$

Carrying out the integrations, one has

$$\int_D \mathbf{X}\,\mathbf{X}^\intercal \, \mathrm{d}\mathbf{x} = \left(\Pi_{s=1}^{a} \frac{L_s}{2}\right)\mathbf{I}, \tag{15a}$$

$$\int_D \mathbf{X}\mathcal{L}\mathbf{X}^\intercal \, \mathrm{d}\mathbf{x} = m\left(\Pi_{s=1}^{a} \frac{L_s}{2}\right)\mathbf{\Omega}_0^2. \tag{15b}$$

[1] In the case of the plate, as a cautionary note, it is remarked that the energy analysis presented here is valid only under simply-supported boundary conditions. For other types of boundary conditions, the energy analysis must be modified, see e.g. [13].

Here, $\mathbf{I}$ is the $N \times N$ identity matrix, and $\mathbf{\Omega}_0 = \text{diag}\,[\omega_n]$ is an $N \times N$ diagonal matrix whose diagonal elements are the resonant frequencies ω_n. Direct expressions for such frequencies are

$$\omega_n = \sqrt{\alpha_1 \sum_{s=1}^{a} \frac{n_s^2 \pi^2}{L_s^2} + \alpha_2 \left(\sum_{s=1}^{a} \frac{n_s^2 \pi^2}{L_s^2} \right)^2}. \tag{16}$$

In practice, the original PDE (1), after the modal projection (14), is now written as

$$\ddot{\mathbf{q}} + \mathbf{\Omega}_0^2 \mathbf{q} = 0. \tag{17}$$

It is seen that the modal projection, effectively reduced the original PDE to a set of parallel ordinary differential equations (ODEs). It is remarked that the total energy (8) can now be expressed as a sum over the energy of uncoupled modes, as

$$H = m \left(\Pi_{s=1}^{a} \frac{L_s}{2} \right) \sum_{n=1}^{N} \left(\frac{\dot{q}_n^2}{2} + \frac{\omega_n^2 q_n}{2} \right). \tag{18}$$

2.3 Coupled Subsystems

Two subsystems may be coupled as follows:

$$\left(m_u \partial_t^2 + \mathcal{L}_u \right) u(t, \mathbf{x}) = \delta(\mathbf{x} - \mathbf{x}_\eta) K(t) \eta(t), \tag{19a}$$
$$\left(m_w \partial_t^2 + \mathcal{L}_w \right) w(t, \mathbf{y}) = -\delta(\mathbf{y} - \mathbf{y}_\eta) K(t) \eta(t) \tag{19b}$$
$$\eta(t) = w(t, \mathbf{y}_\eta) - u(t, \mathbf{x}_\eta), \tag{19c}$$
$$K(t) = K_l + K_c \eta^2(t). \tag{19d}$$

In practice, the two subsystems are connected via a nonlinear connection, expressed as the sum of a linear and a cubic spring. The springs' stiffness constants are denoted K_l and K_c. The contact points are given as $\mathbf{x}_\eta$ and $\mathbf{y}_\eta$, respectively. It is remarked that such system is conservative, since the total energy is expressed as

$$H = H_u + H_w + H_\eta, \tag{20}$$

where

$$H_\eta = \eta^2 \left(\frac{K_l}{2} + \frac{K_c}{4} \eta^2 \right),$$

and where H_u, H_w are the energies of the isolated subsystems, with expressions analogous to (8).

In order to reduce this system to a modal form, one writes

$$u(t, \mathbf{x}) = \mathbf{X}^\intercal(\mathbf{x}) \mathbf{q}^u(t),\ w(t, \mathbf{y}) = \mathbf{Y}^\intercal(\mathbf{y}) \mathbf{q}^w(t). \tag{21}$$

Then, expansions analogous to (12) and (13) are assumed valid for $\mathbf{X}$, $\mathbf{Y}$. Doing a modal projection over $\mathbf{X}$ in (19a), and over $\mathbf{Y}$ in (19b), results in

$$\ddot{\mathbf{q}} + \left(\mathbf{\Omega}_0^2 + \mathbf{\Omega}_\eta^2 \right) \mathbf{q} = 0. \tag{22}$$

This is a generalisation of (17), including the effects of coupling. In the above, one has

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}^u \\ \mathbf{q}^w \end{bmatrix}, \mathbf{\Omega}_0 = \begin{bmatrix} \mathbf{\Omega}_0^u & \mathbf{0} \\ \mathbf{0} & \mathbf{\Omega}_0^w \end{bmatrix}. \tag{23}$$

The constant frequencies $\mathbf{\Omega}_0^u$, $\mathbf{\Omega}_0^u$ are the resonant frequencies of the two subsystems in isolation, given by (16). The state-dependent frequencies are expressed as

$$\mathbf{\Omega}_\eta^2 = \mathbf{N} \left(K_l + K_c \eta^2 \right) \begin{bmatrix} -\mathbf{X}(\mathbf{x}_\eta) \\ \mathbf{Y}(\mathbf{y}_\eta) \end{bmatrix} \begin{bmatrix} -\mathbf{X}(\mathbf{x}_\eta) \\ \mathbf{Y}(\mathbf{y}_\eta) \end{bmatrix}^\intercal \tag{24}$$

where the normalisation matrix is

$$\mathbf{N} = \begin{pmatrix} \frac{1}{m_u} \left(\Pi_{s=1}^{a} \frac{2}{L_s^u} \right) \mathbf{I}^u & \mathbf{0} \\ \mathbf{0} & \frac{1}{m_w} \left(\Pi_{s=1}^{a} \frac{2}{L_s^w} \right) \mathbf{I}^w \end{pmatrix} \tag{25}$$

2.4 Loss, Forcing and Output

The modal scheme (22) can be modified so to include loss and forcing, trivially, as

$$\ddot{\mathbf{q}} + \mathbf{C} \dot{\mathbf{q}} + \left(\mathbf{\Omega}_0^2 + \mathbf{\Omega}_\eta^2 \right) \mathbf{q} = \mathbf{r} f(t). \tag{26}$$

Here, $\mathbf{C}$ is a symmetric, positive-definite matrix, and $\mathbf{r}$ is a matrix of input weights. These may be computed directly by projecting the input onto the modes, but one may of course assign different input weights for sound synthesis purposes. Output is extracted simply as

$$g^u(t) = \sum_{j=1}^{N^u} \beta_j^u q_j^u(t),\ g^w(t) = \sum_{j=1}^{N^w} \beta_j^w q_j^w(t), \tag{27}$$

and again, the weights β_j may be computed by projecting the output point onto the modes, though one is free to choose any combination of weights. Multiple outputs are obtained by simply computing multiple sums with different sets of weights.

2.5 Finite Difference Schemes

Integration of (26) may be performed via a suitable finite difference scheme. To that end, time is discretised by means of a sample rate $f_s = 1/k$, where k is the time step. In practice, the solution is evaluated at the time steps kn, where $n \in \mathbb{N}$. Approximations to the time derivatives, and to the identity, are given here as

$$\frac{dq}{dt} \to \delta q^n \triangleq \frac{q^{n+1} - q^{n-1}}{2k}, \tag{28a}$$
$$\frac{d^2 q}{dt^2} \to \delta^{(2)} q^n \triangleq \frac{q^{n+1} - 2q^n + q^{n-1}}{k^2}, \tag{28b}$$
$$1q \to \mu q^n \triangleq \frac{q^{n+1} + q^{n-1}}{2}. \tag{28c}$$

A finite difference scheme, approximating (26), is then given as

$$\delta^{(2)} \mathbf{q}^n + \mathbf{C}\, \delta \mathbf{q}^n + \mathbf{\Omega}_0^2\, \mathbf{q}^n + \mathbf{\Omega}_\eta^2\, \mu \mathbf{q}^n = \mathbf{r} f^n. \tag{29}$$

The particular form shown here is energy-passive, with a form of the numerical energy discretising (20), plus dissipation (not shown here for brevity). This discrete energy is also non-negative, whenever the time step satisfies [13]

$$k < 2/\omega_n,\ \forall\, n. \tag{30}$$

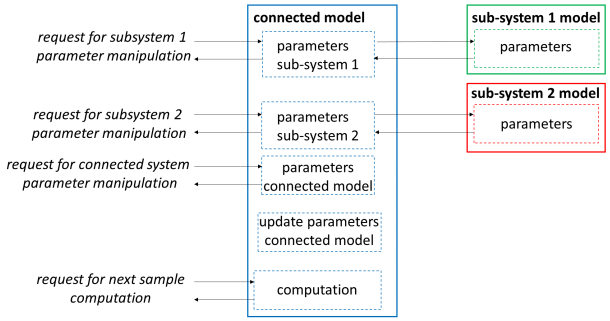

Figure 1. The structure of a connected model class is divided in five sections: one for the manipulation of the first subsystem's parameters, one for the manipulation of the second subsystem's parameters, one for the connected system's parameters, one for the update methods and one for the computation of the next samples.

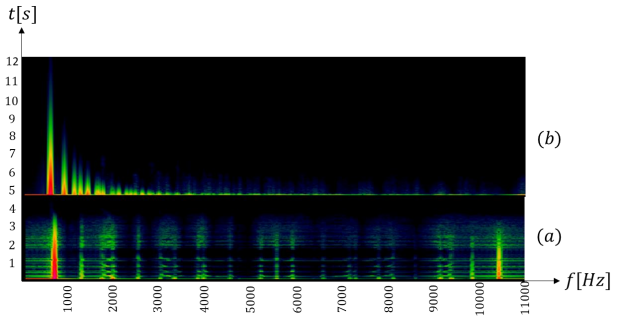

Figure 2. Here is reported the spectrogram of the string-plate model considering the plugin's default parameters and the note C5. In (a) both the excitation the output are applied to the string sub-system; in (b) they are instead applied to the plate sub-system.

Here, ω_n is the nth characteristic frequency of either one of the two subsystems, given in (16). In particular, this condition fixes N^u, N^w, the largest amount of modes available for the two subsystems, for a given value of sample rate. It is remarked that the state-dependent nonlinearity is here approximated implicitly, such that the stability condition above arises solely as a consequence of the discretisation of the linear part. It is noted also that scheme (29) is in fact *linearly-implicit*, in that the solution of a single linear system is required at each time step, making the update efficient and suited for real-time synthesis.

3. C++ MODEL IMPLEMENTATION

In this section, the implementation of the model in plugin form is discussed. The first step for both the creation of the connected models and the same implementation in a real-time system is the development of the sub-systems model. The model implemented in Matlab programming language is used as a landmark for the successive development of the plugin in C++. As a result, firstly a complete model is developed, or rather a class which contained all the typical attributes of the sub-systems and the portion of code related to the processing of the next samples as well. Successively, for the implementation of the connected models, the classes used were exactly the same, but without the portions of code dedicated to the computation of the model. As a matter of fact, this part is implemented inside the class related to the connected model and consequently the sub-system class can be mostly considered as an auxiliary class where the updated parameters are stored, see figure 1.

3.1 Plugin Description

The plugin allows to select the three models previously described, offered as presets. In the plugin the user can manipulate different parameters (see figure 2) such as the damping; it is possible to choose both the model and the excitation type.

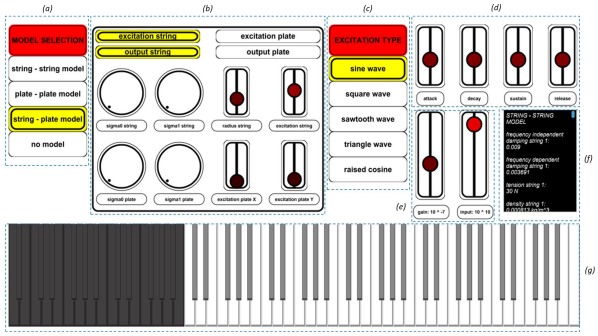

Figure 3. In (a) the model selection menu, in (b) the model panel, in (c) the excitation type menu, in (d) the ADSR panel, in (e) the sliders to control the output gain and the input intensity respectively, in (f) a box where the parameters of the models are shown and finally in (g) the keyboard.

The plugin is composed of different sections (see figure 3):

- Excitation Type Selection Menu: There are five different types of excitation available: the sine waveform, the square waveform, the sawtooth waveform, the triangle waveform and the raised cosine. The first four types of excitations mentioned are pure digital sound processing components, synthesized with wavetable synthesis [14]. This allows to use the model not as an instrument, but as a resonator. The raised cosine [13] has a stricter physical meaning of an impulse and it uses the system as an instrument. As a result, in this case the pitch is given by the same system, for example by changing appropriate parameters. In order to tune the string, the following expression case be used:

$$\omega_n = \sqrt{\frac{T}{\rho A}\left(\frac{n\pi}{L}\right)^2 + \frac{EI}{\rho A}\left(\frac{n\pi}{L}\right)^4}$$

Considering that $\omega_n = 2\pi f_n$ and that for the fundamental frequency $n = 1$, we can adjust the tension

of the string and to fix all the other parameters to obtained the desired pitch:

$$T = \rho A \left(\frac{L}{\pi}\right)^2 (2\pi f_0)^2 - EI \left(\frac{\pi}{L}\right)^2 \qquad (31)$$

An analogue procedure can be used to tune the plate, but considering the ω_n value associated to the plate with $n_x = 1$ and $n_y = 1$.

- ADSR Menu: when the waveforms such as the sine wave, the square wave, the sawtooth wave and the triangle wave are used as input, they could be sustained potentially for an infinite amount of time. In these cases, the application of an ADSR (Attack-Decay-Sustain-Release) provides the user the possibility to use the physical model as a resonator. The plugin presents a panel containing four sliders: one for the attack time, one for the decay time, one for the sustain time and one for the release time.
- Gain slider: The gain slider is useful in order to scale the output of the models, since different models can return different levels of amplitude.
- Input slider: The input slider allows to calibrate the intensity of the input. Therefore, in the case of a waveform, the input intensity represents the amplitude; similarly, for the raised cosine function, it represents the peak value.
- Parameters display: The parameters display is a text box where all the values of the parameters are shown for all the presets proposed. A lot of parameters of the model actually affect the pitch; consequently, they cannot be changed arbitrarily by the user. Nevertheless, in the parameters display all the values can be monitored.
- Model selection menu: the model selection menu is useful to choose the preset model to play. It is composed of four options: "string-string model", "plate-plate model", "string-plate model", "no model" (this further option allows to directly play the input excitation).
- String-string model panel: this panel is visible when the "string-string model" option is selected in the model selection menu. It allows to manipulate the following parameters of the string-string model: the frequency dependent damping, the frequency independent damping and the radius of the two string, as well as where the excitation is applied and from where the output is extrapolated.
- Plate-plate model: this panel is visible when the "plate-plate model" option is selected in the model selection menu. It allows to manipulate the following parameters of the plate-plate model: the frequency dependent damping, the frequency independent damping of the two plates as well as where the excitation is applied and from where the output is extrapolated.

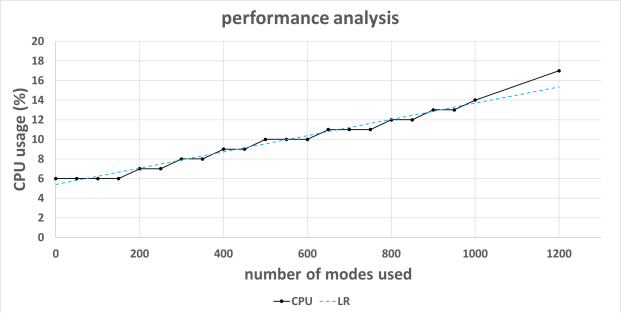

Figure 4. The current chart shows the variation of the CPU usage percentage over the increasing dimensions (in terms of number of modes used) of the system. In blue the linear regression is reported. Notice the linearity of the relationship between the CPU usage and the number of modes used by the system.

- String-plate model: this panel is visible when the "string-plate model" option is selected in the model selection menu. It allows to manipulate the following parameters of the string-plate model: the frequency dependent damping, the frequency independent damping of both the string and the plate, the radius of the string and also where the excitation is applied and from where the output is extrapolated.
- Keyboard: The keyboard component is a simple graphical representation of the notes currently active.

4. PERFORMANCE ANALYSIS

In order to establish which are the optimal conditions of the software and which are the limits, an analysis of the plugin's performance is conducted. Such analysis is mostly related to the global dimension of the system in terms of modes, which should be respected to have an effective result. The test is conducted on a laptop with installed the operating system "Microsoft Windows 10 Home", using a processor "Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz". The performance of the plugin is monitored by means of the task manager in the performance section, where the current usage of CPU of the computer is constantly displayed in terms of percentage. Without using the plugin, the amount of CPU used is around 6%. This is due to the working of other parallel processes of the system and could be considered as the offset percentage of our analysis. Consequently, the CPU usage is registered considering systems of different dimensions. The chart in Figure 4 clearly shows the linearity of such relation between CPU usage and number of modes used. Consequently, a more accurate procedure based on covariance and Pearson's correlation coefficient is carried out [15]. The covariance between the CPU usage values and the number of modes used is expressed as $\sigma_{C,M} = E(C \cdot M) - E(C) \cdot E(M) = 6165.909 - 9.818182 \cdot 531.8182 = 944.4215$. Here C is the variable regarding the CPU usage values, M is the variable related to the number of modes used, $E(C)$ is the

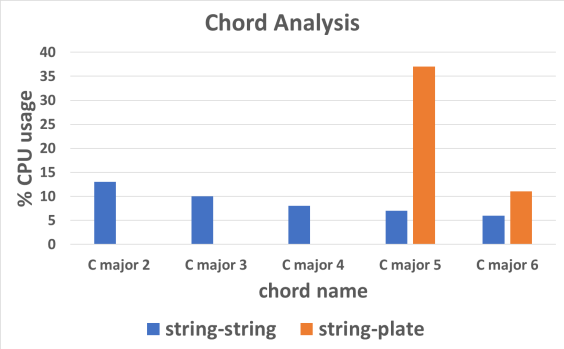

Figure 5. The current bar chart shows how the performance of a chord is affected by the register (for higher ones the computation required is lower, while conversely for lower ones it is required a higher level of CPU usage) and by the instrument. The CPU usage of the model composed of two stiff strings is reported with blue bars, while for the model composed of a string and a plate the bars are orange. It is easy to visualize how the first one is still suited for a real-time application even for more than one note played simultaneously. Conversely, the second one quickly "explodes" for lower registers and consequently it is supposed to be played monophonically.

mean of CPU usage samples, $E(M)$ is the mean of the number of modes used samples and finally $E(C \cdot M)$ is the mean of the product between the samples of CPU usage and the samples of number of modes. Consequently, the Pearson's correlation coefficient can be easily found as $\rho_{C,M} = \frac{\sigma_{C,M}}{\sigma_C \cdot \sigma_M} = \frac{944.4215}{337.5491 \cdot 2.970271} = 0.941961$ (a value close to 1.0 further confirms the hypothesis of a linear relation). The linear regression line [15] can be seen in Figure 4.

4.1 Analysis of Chords and Multiples Notes

The current analysis is particularly useful to study the behaviour of the system in case of a chord and consequently with more than one note active at the same time. Obviously, the number of total number of modes is additive, considering the number of modes used by the single notes, separately. This property makes very easy to understand if a given chord is computationally sustainable in real-time or not and the total amount of CPU usage necessary. In this analysis, the chord C Major in different registers were arbitrarily chosen and it is found how the performance is strongly affected by both the register and the type of model used (as can be seen in figure 5). It is easy to notice that for lower registers, a higher amount of CPU usage is required and that the plate is much heavier from a computational point of view than a string.

5. CONCLUSIONS

The plugin developed in the current work demonstrates how it is possible to implement software for real-time synthesis of audio using models based on physical laws and numerical solutions. In fact, it is noticed how the increase of the dimension of the system (in terms of number of modes used) brings to a linear increment of the CPU usage, at least inside the range of values where the tests were carried out (between 0 and 1200 modes). Moreover, nonlinear elements were successfully introduced in the connection between the different sub-systems of the connected models. An objective for future work is the creation of a real-time plugin representing a model composed of a higher number of subsystems.

6. ACKNOWLEDGEMENTS

This work is supported by the European Research Council (ERC), under the European Union's Horizon 2020 research and innovation programme, grant 2020-StG-950084-NEMUS

7. REFERENCES

[1] D. Bernoulli, "Réflexions et éclaircissemens sur les nouvelles vibrations des cordes," *Mémoires de l'Academie Royale de Berlin*, vol. 9, pp. 147–195, 1753.

[2] J. B. Fourier, "Théorie du mouvement de la chaleur dans les corps solides," *Mémoires de l'Academie Royale des Sciences de l'Institut de France*, vol. 4, pp. 185–556, 1819.

[3] S. Bilbao, "Modal synthesis," https://ccrma.stanford.edu/~bilbao/booktop/node14.html, 2006.

[4] K. Van Den Doel and D. K. Pai, "Modal synthesis for vibrating objects," *Audio Anectodes. AK Peter, Natick, MA*, pp. 1–8, 2003.

[5] Z. Ren, H. Yeh, and M. C. Lin, "Example-guided physically based modal sound synthesis," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 1, pp. 1–16, 2013.

[6] J. D. Morrison and J.-M. Adrien, "Mosaic: A framework for modal synthesis," *Computer Music Journal*, vol. 17, no. 1, pp. 45–56, 1993.

[7] IRCAM, "Modalys," https://support.ircam.fr/docs/Modalys/3.4.1/co/publication-web.html, 2014.

[8] M. Ducceschi and C. Webb, "Plate reverberation: Towards the development of a real-time plug-in for the working musician," in *Proceedings of the International Conference on Acoustics (ICA 2016)*, Buenos Aires, Argentina, September 2016.

[9] S. Bilbao, C. Desvages, M. Ducceschi, B. Hamilton, R. Harrison-Harsley, A. Torin, and C. Webb, "Physical modeling, algorithms, and sound synthesis: The ness project," *Computer Music Journal*, vol. 43, no. 2-3, pp. 15–30, 2020.

[10] M. Ducceschi and S. Bilbao, "Non-iterative, conservative schemes for geometrically exact nonlinear string vibration," in *Proceedings of the 23rd International Conference on Acoustics (ICA19)*, Aachen, Germany, September 2019.

[11] 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*, vol. 334, pp. 313–331, 2015.

[12] C. J. Webb and S. Bilbao, "On the limits of real-time physical modelling synthesis with a modular environment," in *Proceedings of the International Conference on Digital Audio Effects (DAFx15)*, Trondheim, Norway, December 2015.

[13] S. Bilbao, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*, 1st ed. Wiley, 2009.

[14] T. H. Park, *Introduction to digital signal processing computer musically speaking*. World Scientific Publishing Company, 2010.

[15] E. L. Piazza, *Probabilità e statistica (Probability and Statistics)*. Esculapio, 2014.