Chapter 8

Low-level models: resonators, interactions, surface textures

Federico Avanzini
Università di Padova – Department of Information Engineering
Padova, Italy
avanzini@dei.unipd.it

Matthias Rath, Davide Rocchesso and Laura Ottaviani
Università di Verona – Department of Computer Science
Verona, Italy
rath@sci.univr.it, davide.rocchesso@univr.it,
ottaviani@sci.univr.it

8.1 Introduction

"Traditional" techniques of sound synthesis (e.g. additive, subtractive, FM) are based on, and accessed through, signal theoretic parameters and tools. While deep research has established a close connection to conventional musical terms, such as pitch, timbre or loudness, newer psychoacoustic works [92] point out that the nature of everyday listening is rather different. From the *ecological* viewpoint, auditory perception delivers information about a listener's

surrounding, i.e. objects in this surrounding and their interactions, mostly without awareness of and beyond attributes of musical listening. The common use of wavetables, i.e. the playback of prerecorded sound files, can probably be seen as the standard reaction to the severe restrictions existing in former methods of sound generation. That approach is still signal-based, and not satisfactory in many contexts, such as in human-computer interaction or in interactive virtual environments, due to the static nature of the produced sounds.

In contrast, we use the term *acoustic modeling* to refer to the development of "sound objects" that incorporate a (possibly) complex responsive acoustic behavior, expressive in the sense of ecological hearing, rather than fixed isolated signals. Physically-based models offer a viable way to synthesize naturally behaving sounds from computational structures that can easily interact with the environment and respond to physical input parameters. Various modeling approaches can be used: Van den Doel et al. [238, 237] proposed modal synthesis [2] as an efficient yet accurate framework for describing the acoustic properties of objects. Contact forces are used to drive the modal synthesizer, under the assumption that the sound-producing phenomena are linear, thus being representable as source-filter systems. For non-interactive applications, it has been proposed to generate sound as a side effect of nonlinear finite element simulations [188]. In this way, sounds arising from complex nonlinear phenomena can be simulated, but the heavy computational load prevents the use of the method in interactive applications. Physical models are widely developed in the computer music community, especially using the waveguide simulation paradigm [223], but their main application has been the faithful simulation of existing musical instruments.

Although real sounds hereby serve as an orientation, realistic simulation is not necessarily the perfect goal: simplifications which preserve and possibly exaggerate certain acoustic aspects, while losing others considered less important, are often preferred. Besides being more effective in conveying certain information, such "cartoonifications" are often cheaper to implement, just like graphical icons are both clearer and easier to draw than photo-realistic pictures. Can the idea of audio cartoons suggest an approach to sound design, that fills the gap between simulation or arrangement of concrete sounds and abstract musical expression?

The design approach outlined in this chapter can be roughly referred to as *low-level* modeling. The basic physical mechanisms involved in sound generation are accounted for, including the description of resonating structures, as well as various interaction modalities, such as impact and continuous contact (friction). Section 8.2 describes an efficient structure for describing resonating

objects, based on the modal analysis/synthesis approach. Section 8.3 presents models for the non-linear interaction forces which arise during collision and friction between two modal resonators, and describes an efficient model for rendering texture of surfaces with variable degree of roughness. Section 8.4 discusses the implementation of these contact models as real-time modules, and addresses the problem of control. Finally, details about the synthesis algorithms are given in the appendix 8.A. This is especially meant to help the understanding of the numerical techniques used for developing real-time algorithms, and provides detailed equations and pseudo-code. Chapter 9 will make use of these sound models for developing *high-level* strategies for sound presentation.

8.2 Modal resonators

8.2.1 Continuous-time model

The simplest possible representation of a mechanical oscillating system is a second-order linear oscillator of the form

$$\ddot{x}^{(r)}(t) + g^{(r)}\dot{x}^{(r)}(t) + \left[\omega^{(r)}\right]^2 x^{(r)}(t) = \frac{1}{m^{(r)}} f_{ext}(t) \quad , \qquad (8.1)$$

where $x^{(r)}$ is the oscillator displacement and f_{ext} represents any external driving force, while the parameters $\omega^{(r)}$ and $g^{(r)}$ are the oscillator center frequency and damping coefficient, respectively. The parameter $1/m^{(r)}$ controls the "inertial" properties of the oscillator (note that $m^{(r)}$ has the dimension of a mass). Such a one-dimensional model provides a basic description of the resonator in terms of its pitch $\omega^{(r)}$ and quality factor $q^{(r)} = \omega^{(r)}/g^{(r)}$. The parameter $g^{(r)}$ relates to the decay properties of the impulse response of system (8.1): specifically, the relation $t_e = 2/g^{(r)}$ holds, where t_e is the $1/e$ decay time of the impulse response.

However, in most cases a single-mode oscillator is not enough to produce interesting and spectrally-rich sounds. A slightly more sophisticated model is obtained by parallel connection of N oscillators such as that of Eq. (8.1). By choosing a different center frequency $\omega_l^{(r)}$ ($l = 1 \dots N$) for each oscillator, it is possible to account for a set $\{\omega_l^{(r)}\}_{l=1}^{N}$ of partials of the resonator spectrum. A set of N decoupled modal resonators excited by the same external force can be described by means of a multi-variable generalization of Eq. (8.1). In matrix

form, this can be written as

$$\begin{bmatrix} \ddot{x}_1^{(r)}(t) \\ \vdots \\ \ddot{x}_N^{(r)}(t) \end{bmatrix} + \boldsymbol{G^{(r)}} \begin{bmatrix} \dot{x}_1^{(r)}(t) \\ \vdots \\ \dot{x}_N^{(r)}(t) \end{bmatrix} + \left[\boldsymbol{\Omega^{(r)}}\right]^2 \begin{bmatrix} x_1^{(r)}(t) \\ \vdots \\ x_N^{(r)}(t) \end{bmatrix} = \boldsymbol{m^{(r)}} f_{ext}(t) \quad , \tag{8.2}$$

where the matrices are given by

$$\begin{aligned} \boldsymbol{\Omega^{(r)}} &= \begin{bmatrix} \omega_1^{(r)} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \omega_N^{(r)} \end{bmatrix} \quad , \\ \boldsymbol{G^{(r)}} &= \begin{bmatrix} g_1^{(r)} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & g_N^{(r)} \end{bmatrix} \quad , \\ \boldsymbol{m^{(r)}} &= \begin{bmatrix} 1/m_1^{(r)} \\ \vdots \\ 1/m_N^{(r)} \end{bmatrix} \quad . \end{aligned} \tag{8.3}$$

When a distributed resonating object is modeled as a chain of N masses connected with springs and dampers, the resulting system is composed of N coupled equations [180]. However, the theory of modal analysis [2] shows that it is generally possible to find a transformation matrix $\boldsymbol{T} = \{t_{jl}\}_{j,l=1}^{N}$ which diagonalizes the system and turns it into a set of decoupled equations. The transformed variables $\{x_l^{(r)}\}_{l=1}^{N}$ are generally referred to as *modal displacements*. The displacement x_j and velocity v_j of the resonating object at a given point $j = 1 \ldots N$ are then given by

$$x_j = \sum_{l=1}^{N} t_{jl} x_l^{(r)} \quad \text{and} \quad \dot{x}_j = \sum_{l=1}^{N} t_{jl} \dot{x}_l^{(r)} \quad . \tag{8.4}$$

The modal description given by Eqs. (8.2), (8.4) provides a high degree of controllability. The damping coefficients $g_l^{(r)}$ control the decay times of each exponentially-decaying mode of the resonator. The frequencies $\omega_l^{(r)}$ can be chosen to reproduce spectra corresponding to various geometries of 1D, 2D and 3D resonators. As an example, the first N resonances of a cavity can be

mapped into the modal frequencies of the N oscillators, and morphing between different shapes can be obtained by designing appropriate trajectories for each of these resonances.

In the remainder of this chapter, the quantities $m_l^{(r)}$ are referred to as *modal masses*, while the quantities $1/m_l^{(r)}$ are referred to as *modal weights*. Note that by allowing the modal masses to vary for each oscillator, the matrix $\boldsymbol{m^{(r)}}$ can be generalized to give control on the amounts of energy provided to each oscillator (see section 8.2.2 below). This permits simulation of position-dependent interaction, in that different interaction points excite the resonator modes in different ways.

8.2.2 Position-dependent excitation

Figure 8.1 shows a membrane which is displaced from its rest position in such a way that only one single mode is set into vibration. The distance of each point of the membrane from the "rest plane" is proportional to the weighting factor $1/m^{(r)}$ of the mode at this position. Note that the intersections of the mode–shape with the rest plane (i.e., the *nodal lines*) remain fixed during the entire cycle of the modal vibration. Therefore, the modal weights at these positions are 0 (equivalently, the modal masses tend to infinity). Correspondingly, an external force applied at these node lines does not excite the mode at all.

In order for the resonator model (8.2) to account for such a situation, the weights $1/m_l^{(r)}$ must be made position-dependent. In other words, the $(N \times 1)$ matrix $\boldsymbol{m^{(r)}}$ must be generalized by defining a $(N \times N)$ matrix $\boldsymbol{M^{(r)}}$, whose element (l, j) is the modal weight of mode l at interaction point j.

There are several possible approaches to gain the position dependent weights. In the case of a finite one dimensional system of point masses with linear interaction forces, modal parameters are exactly found through standard matrix calculations. Most systems of interest of course do not fit these assumptions. In some cases the differential equations of distributed systems can be solved analytically, giving the modal parameters; this holds for several symmetric problems such as circular or rectangular membranes.

Alternatively, either accurate numerical simulations (e.g. waveguide mesh methods) or "real" physical measurements can be used. Impulse responses computed (or recorded) at various interaction points then form a basis for the extraction of modal parameters. The acoustic "robustness" of the modal description allows convincing approximations on the basis of microphone-recorded signals of e.g. an object struck at different points, despite all the involved inaccuracies: spatially distributed interaction, as well as wave dis-

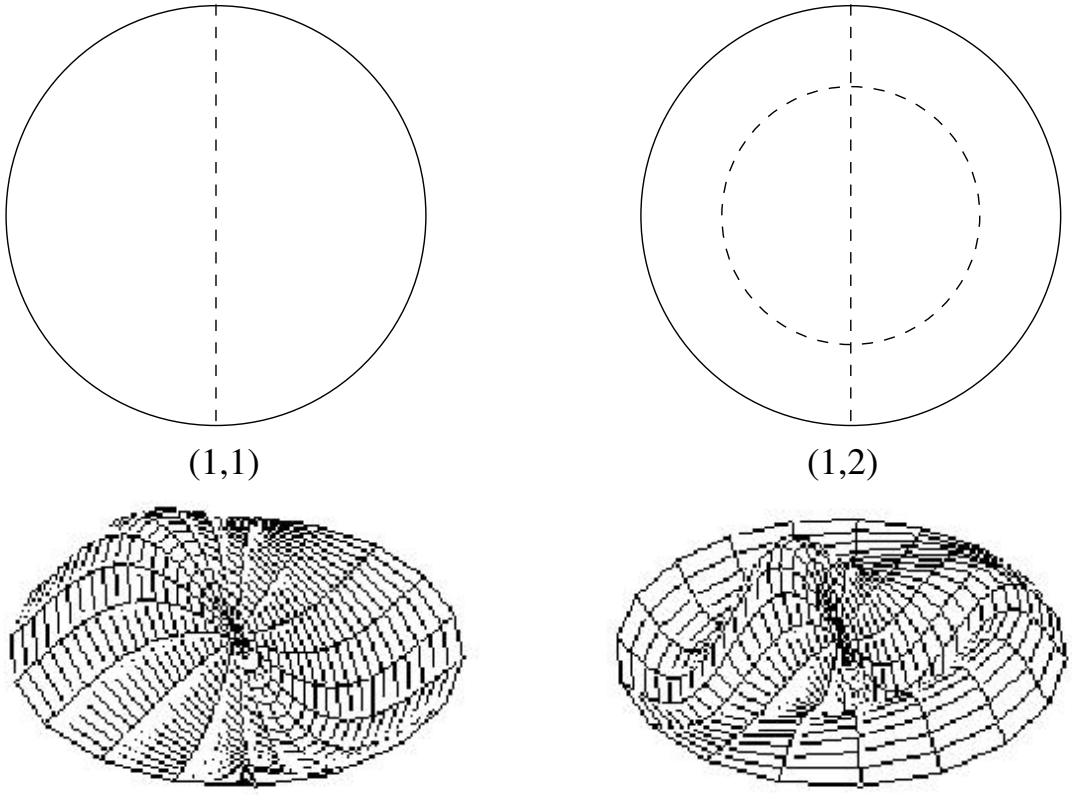


Figure 8.1: A circular membrane displaced from its rest position according to the spatial shape of mode(1,1) (left) and mode(1,2) (right). The frequencies of vibration along these axes are 1.593 and 2.917 times that of mode(0,1) (the "fundamental").

tribution in air, provide signals that are quite far from impulse/frequency responses at single points.

Qualitative observations on modal shapes can be effectively used in a context of cartoonification: for modes of higher frequencies the number of nodal lines increases and their spatial distance decreases accordingly. One consequence is that for higher modes even small inaccuracies in interaction or pickup position may result in strongly different modal weights, so that an element of randomization can here add "naturalness". In the case of vibrating strings, membranes, or clamped bars, the boundary is a nodal line for all the modes, and the higher modes gradually gain importance over the lower modes as the interaction point is shifted toward the boundary. This phenomenon can be well noticed for a drum: if the membrane is struck close to the rim, the excited sound becomes "sharper", as the energy distribution in the frequency spectrum is shifted upward ("rimshots"). For a clamped bar higher partials are dominant

near the fixed end, whereas lower frequencies are stronger for strokes close to the free vibrating boundary (noticeable in sound adjustments of electromechanical pianos).

Similar considerations apply to points of symmetry: some resonant modes, those with modal shapes antisymmetric to central axes, are not excited when the driving force is applied at the center of a round or square membrane. They consequently disappear "bottom–up" when approaching the center point. For 3D resonators, such as cavities, one can generally say that the most effective modal excitation is obtained at the boundary. For instance, in a rectangular room seven modes over eight have a nodal plane passing through the room center, while all the modes are excited at a corner.

8.2.3 Discrete-time equations

The continuous-time system (8.4) is discretized using the bilinear transformation[1], which is usually interpreted as a s-to-z mapping between the Laplace and the Z domains:

$$s = 2F_s \frac{1 - z^{-1}}{1 + z^{-1}} \quad . \tag{8.5}$$

The bilinear transformation is one appealing discretization technique for various reasons. First, its order of accuracy can be seen [146] to be two. Second, the transformation preserves the order of the system (e.g., the second-order equation (8.1) is turned into a second-order difference equation by the bilinear transformation). Finally, the transformation is stable, since the left-half s-plane is mapped by Eq. (8.5) into the unit z-circle. Consequently, the bilinear transformation provides a reasonable trade-off between accuracy and efficiency. On the other hand, some of its properties can be seen as drawbacks in this context. Noticeably, it introduces frequency warping [177], and it is an implicit method (which has some consequences on the resulting numerical equations, as discussed in appendix 8.A.1 below).

After applying transformation (8.5) to system (8.2), the resulting discrete-time system appears as a parallel filter bank of second-order low-pass resonant filters, each one accounting for one specific mode of the resonator. The output of the filter bank can be taken to be the weighted sum (8.4) of either the modal displacement or the modal velocities. Each of the filters can be accessed to its parameters of center-frequency $\omega_l^{(r)}$ and damping coefficient $g_l^{(r)}$ (or, equivalently, decay time t_{el}).

[1] Also known in the numerical analysis literature as the one-step *Adams-Moulton method* [146]

The complete numerical system takes the form

$$\begin{cases} \boldsymbol{x}_l^{(r)}(n) = \boldsymbol{A}_l^{(r)}\boldsymbol{x}_l^{(r)}(n-1) + \boldsymbol{b}_l^{(r)}[y(n) + y(n-1)] \\ \boldsymbol{x}_j(n) = \sum_{l=1}^{N} t_{jl}\boldsymbol{x}_l^{(r)}(n) \\ y(n) = f_{ext}(n) \end{cases} , \tag{8.6}$$

for $j = 1 \dots N$, where the vectors $\boldsymbol{x}_l^{(r)}$ and $\boldsymbol{x_j}$ are defined as $\boldsymbol{x}_l^{(r)} = \begin{bmatrix} x_l^{(r)} \\ \dot{x}_l^{(r)} \end{bmatrix}$, and $\boldsymbol{x_j} = \begin{bmatrix} x_j \\ v_j \end{bmatrix}$, respectively. The matrices $\boldsymbol{A}_l^{(r)}$ and $\boldsymbol{b}_l^{(r)}$ are found to be

$$\begin{aligned} \boldsymbol{A}_l^{(r)} &= \frac{1}{\Delta_l^{(r)}} \begin{bmatrix} \Delta_l^{(r)} - \left[\omega_l^{(r)}\right]^2/2 & F_s \\ -F_s\left[\omega_l^{(r)}\right]^2 & 2F_s^2 - \Delta_l^{(r)} \end{bmatrix} , \\ \boldsymbol{b}_l^{(r)} &= \frac{1}{m_l^{(r)}} \cdot \frac{1}{4\Delta^{(r)}} \begin{bmatrix} 1 \\ 2F_s \end{bmatrix} , \end{aligned} \tag{8.7}$$

where the quantity $\Delta_l^{(r)}$ is given by $\Delta_l^{(r)} = F_s^2 + F_s/t_{el} + \left[\omega_l^{(r)}\right]^2/4$.

Note that the modal weights $1/m_l^{(r)}$ only appear in the definition of $\boldsymbol{b}_l^{(r)}$, which controls the extent to which mode l is excited by the external force f_{ext}. Following the discussion in section 8.2.2 above, multiple excitation points can be modeled by attaching an additional index $j = 1 \dots N$ to the modal masses.

8.3 Interactions

8.3.1 Impact

In this section we construct a continuous-time impact model between two modal resonators, following the "cartoonification" approach that has informed the research activities of the SOb project: Figure 8.2 shows that the sound model is controlled through a small number of parameters, which are clearly related either to the resonating objects or to the interaction force. The precise meaning and role of the parameters depicted in Figure 8.2 will be explained in the remainder of this section.

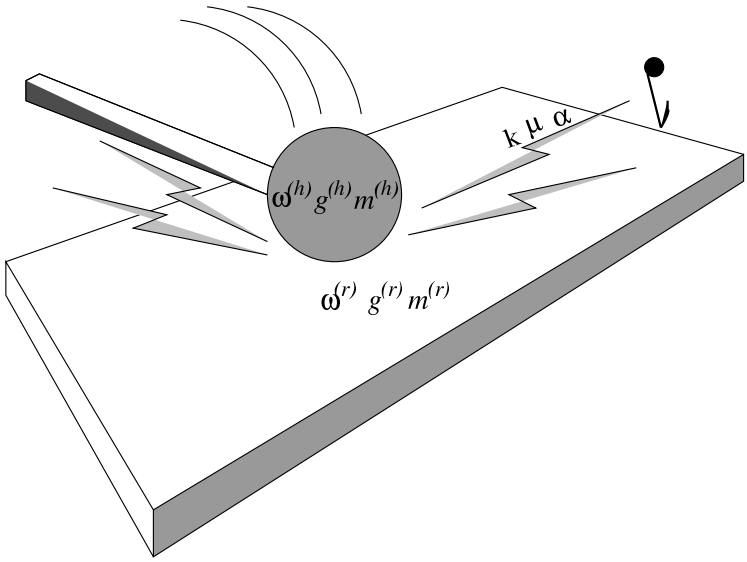


Figure 8.2: Cartoon impact between two modal resonators.

Impact models have been widely studied in musical acoustics, mainly in relation with hammer-string interaction in the piano. If the contact area between the two colliding objects is assumed to be small (ideally, a point), the simplest model [111] states a polynomial dependence of the contact force f on the hammer felt compression x:

$$f(x(t)) = \begin{cases} k[x(t)]^{\alpha} & x > 0 \\ 0 & x \leq 0 \end{cases} , \tag{8.8}$$

The compression x at the contact point is computed as the difference between hammer and string displacements. Therefore, the condition $x > 0$ states that there is actual felt compression, while the complementary condition $x \leq 0$ says that the two objects are not in contact. The parameter k is the force *stiffness*, and the exponent α depends on the local geometry around the contact area. As an example, in an ideal impact between two spherical object α takes the value 1.5. Typical experimental values in a piano hammer felt range from 1.5 to 3.5, with no definite trend from bass to treble.

More realistic models have to take into account the hysteresis effects involved in the interaction. As an example, it is known that the force-compression characteristic in a piano hammer exhibits a hysteretic behavior, such that loading and unloading of the hammer felt are not alike. In particular, the dynamic force-compression characteristic is strongly dependent on the hammer normal velocity before collision. In order to account for these phenomena, Stulov [230] proposed an improved model where the contact force possesses

history-dependent properties. The idea, which is taken from the general theory of mechanics of solids, is that the spring stiffness k in Eq. (8.8) has to be replaced by a time-dependent operator. Consequently, according to Stulov the contact force can be modeled as

$$f(x(t),t) = \begin{cases} k[1 - h_r(t)] * [x(t)^\alpha] & x > 0 \\ 0 & x \leq 0 \end{cases} , \qquad (8.9)$$

where $*$ stands for the continuous-time convolution operator, and $h_r(t) = \frac{\epsilon}{\tau} e^{-t/\tau}$ is a *relaxation function* that controls the "memory" of the material. In fact, by rewriting the convolution explicitly the Stulov force is seen to be:

$$f(x(t),t) = kx(t)^\alpha - \frac{\epsilon}{\tau} e^{-t/\tau} \int_0^t e^{\xi/\tau} x(\xi)^\alpha \, d\xi \quad \text{for} \quad x > 0 \quad . \quad (8.10)$$

The Stulov model has proved to be successful in fitting experimental data where a hammer strikes a massive surface, and force, acceleration, displacement signal are recorded. Borin and De Poli [23] showed that it can be implemented numerically without significant losses in accuracy, stability and efficiency with respect to the simpler model (8.8).

Useful results on impact modeling are also found from studies in robotics. Physical modeling of contact events is indeed a relevant issue in dynamic simulations of robotic systems, when physical contact with the environment is required in order for the system to execute its assigned task (for example, handling of parts by an industrial manipulator during assembly tasks, or manipulator collisions with unknown objects when operating in an unstructured environment). Marhefka and Orin [167] provide a detailed discussion of a collision model that was originally proposed by Hunt and Crossley [122]. Under the hypothesis that the contact surface is small, Hunt and Crossley proposed the following form for the contact force f:

$$f(x(t),v(t)) = \begin{cases} kx(t)^\alpha + \lambda x(t)^\alpha \cdot v(t) = kx(t)^\alpha \left(1 + \mu v(t)\right) & x > 0 \\ 0 & x \leq 0 \end{cases} , \qquad (8.11)$$

where $v(t) = \dot{x}(t)$ is the compression velocity, and k and α are defined as above. The parameter λ is the force damping weight, and $\mu = \lambda/k$ is a mathematically convenient term which is called "viscoelastic characteristic" by Marhefka and Orin. Similarly to Eqs. (8.8) and (8.9), the value of the exponent α depends only on the local geometry around the contact surface. Note that the force model (8.11) includes both an elastic component kx^α and a dissipative term $\lambda x^\alpha v$. Moreover, the dissipative term depends on both x and v, and is zero for zero compression.

Marhefka and Orin have studied the following case: an idealized hammer, described as a lumped mass $m^{(h)}$, strikes a surface. The surface mass is assumed to be much greater than $m^{(h)}$, therefore the surface is assumed not to move during the collision. When the two objects collide, the hammer initial conditions are $x^{(h)}(0) = 0$ (hammer position) and $\dot{x}^{(h)}(0) = -v_{in}$ (hammer normal velocity before collision). Since the surface is assumed not to move, the hammer position and velocity relate to the compression and compression velocity through the equalities $x^{(h)}(t) = -x(t)$, $\dot{x}^{(h)}(t) = -v(t)$. The hammer trajectory is therefore described by the differential equation $m^{(h)}\ddot{x}^{(h)} = f(-x^{(h)}, -\dot{x}^{(h)})$. Then it is shown in [167] that

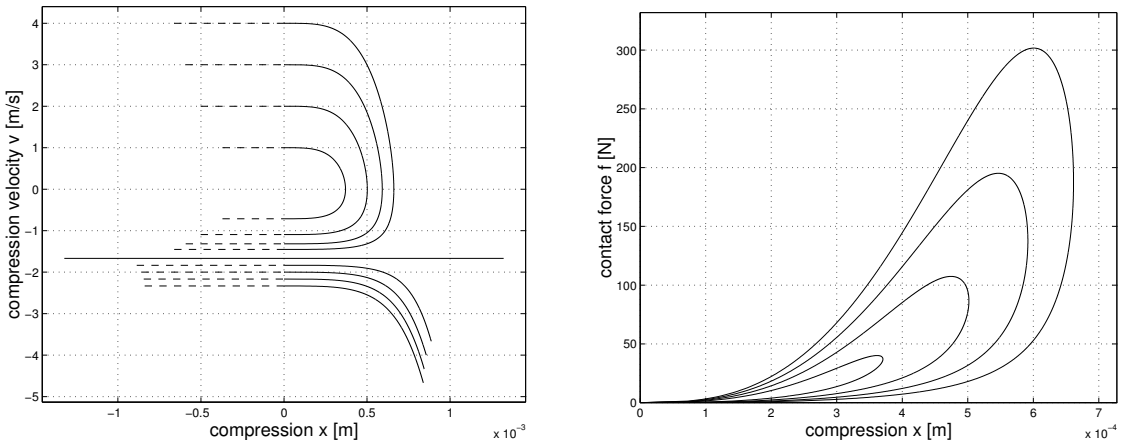
$$\frac{d(\dot{x}^{(h)})}{dx^{(h)}} = \frac{\dot{v}}{v} = \frac{(\Lambda v + K)\,[x]^{\alpha}}{v} \quad \Rightarrow \quad \int \frac{v\,dv}{(\Lambda v + K)} = \int [x]^{\alpha}\,dx \quad , \tag{8.12}$$

where two auxiliary parameters $\Lambda = -\lambda/m^{(h)}$ and $K = -k/m^{(h)}$ have been introduced for clarity. The integral in Eq. (8.12) can be computed explicitly and gives

$$x(v) = \left[\left(\frac{\alpha+1}{\Lambda^2}\right)\left(\Lambda(v - v_{in}) - K\log\left|\frac{K+\Lambda v}{K+\Lambda v_{in}}\right|\right)\right]^{\frac{1}{\alpha+1}} \quad . \tag{8.13}$$

Eq. (8.13) provides x as a function of v, and can therefore be exploited for plotting the phase portrait on the (x, v) plane. This is shown in Figure 8.3a.

Another remark by Marhefka and Orin is concerned with "stickiness" properties of the contact force f. From Eq. (8.11), it can be seen that f becomes inward (or sticky) if $v < v_{lim} := -1/\mu$. However, this limit velocity is never exceeded on a trajectory with initial conditions $x = 0$, $v = v_{in}$, as shown in the phase portrait of Figure 8.3a. The upper half of the plot depicts the trajectories of a hammer which strikes the surface with various normal velocities (trajectories are traveled in clockwise direction). Note that the output velocities after collision v_{out} are always smaller in magnitude than the corresponding v_{in}. Moreover, for increasing v_{in} the resulting v_{out} converges to the limit value v_{lim}. The horizontal line $v = v_{lim}$ corresponds to the trajectory where the elastic and dissipative terms cancel, and therefore the hammer travels from right to left with constant velocity. This horizontal line separates two regions of the phase space, and the lower region is never entered by the upper paths. The lower trajectories are entered for an initial compression $x < 0$ and initial *negative* compression velocity $v_{in} < v_{lim}$. If such conditions are imposed, then one of the lower trajectories is traveled from right to left: the hammer



(a) (b)

Figure 8.3: Collision of a hammer with a massive surface for various v_{in}'s; (a) phase portrait, (b) compression-force characteristics. Values for the hammer parameters are $m^{(h)} = 10^{-2}$ [Kg], $k = 1.5 \cdot 10^{11}$ [N/m$^{\alpha}$], $\mu = 0.6$ [s/m], $\alpha = 2.8$, $v_{in} = 1 \ldots 4$ [m/s].

bounces back from the surface, while its velocity decreases in magnitude, due to the dissipative term in the force f.

Figure 8.3b shows the compression-force characteristics during collision. Note that the dissipative term $\lambda x^{\alpha} v$ introduces hysteresis. In this respect the role of the dissipative term, in the Hunt and Crossley model, is very similar to that of the relaxation function in the Stulov model.

The Hunt and Crossley impact model (8.11) can be used as a coupling mechanism between two modal resonators (described in section 8.2). For clarity, the two objects are denoted with the superscripts (h) and (r), which stand for "hammer" and "resonator", respectively. The two objects interact through the impact contact force $f(x, v)$ given in Eq. (8.11). Assuming that the interaction occurs at point $l = 1 \ldots N^{(h)}$ of the hammer and point $m = 1 \ldots N^{(r)}$ of the resonator, the continuous-time equations of the coupled system are given

by:

$$\begin{cases} \ddot{x}_i^{(h)} + g_i^{(h)}\dot{x}_i^{(h)} + \left[\omega_i^{(h)}\right]^2 x_i^{(h)} = \dfrac{1}{m_{il}^{(h)}}(f_e^{(h)} + f) \quad , \quad (i = 1 \dots N^{(h)}) \\ \ddot{x}_j^{(r)} + g_j^{(r)}\dot{x}_j^{(r)} + \left[\omega_j^{(r)}\right]^2 x_j^{(r)} = \dfrac{1}{m_{jm}^{(r)}}(f_e^{(r)} - f) \quad , \quad (j = 1 \dots N^{(r)}) \\ x = \sum_{j=1}^{N^{(r)}} t_{mj}^{(r)} x_j^{(r)} - \sum_{i=1}^{N^{(h)}} t_{li}^{(h)} x_i^{(h)} \\ v = \sum_{j=1}^{N^{(r)}} t_{mj}^{(r)} \dot{x}_j^{(r)} - \sum_{i=1}^{N^{(h)}} t_{li}^{(h)} \dot{x}_i^{(h)} \\ f(x,v) = \begin{cases} kx(t)^\alpha + \lambda x(t)^\alpha \cdot v(t) & x > 0 \\ 0 & x \leq 0 \end{cases} \text{ (impact force)} \end{cases} , \tag{8.14}$$

where $x_i^{(h)}$ and $x_j^{(r)}$ are the modal displacements of the hammer and the resonator, respectively. The terms $f_e^{(h)}, f_e^{(r)}$ represent external forces, while the integers $N^{(h)}$ and $N^{(r)}$ are the number of modes for the hammer and the resonator, respectively. As a special case, one or both the objects can be a "hammer", i.e. an inertial mass described with one mode, zero spring constant and zero internal damping. As another special case, one object can be a "rigid wall", i.e. a modal object with an ideally infinite mass.

8.3.2 Friction

The continuous-time friction model presented in this section follows the same "cartoon" approach adopted for the impact model: Figure 8.4 shows that the sound model is controlled through a small number of parameters, which are clearly related either to the resonating objects or to the interaction force. The precise meaning and role of the parameters depicted in Figure 8.4 will be explained in the remainder of this section. Given the non-linear nature of friction, interacting structures with few resonances are able to produce complex and rich sonorities. This is an important issue from a computational viewpoint, since efficient models can be developed that provide realistic simulations of contacting objects. It is necessary to bear in mind, however, that when looking for accurate reproduction of friction phenomena ≪ *[...] there are many different mechanisms. To construct a general friction model from physical first principles is simply not possible [...]* ≫ [189].

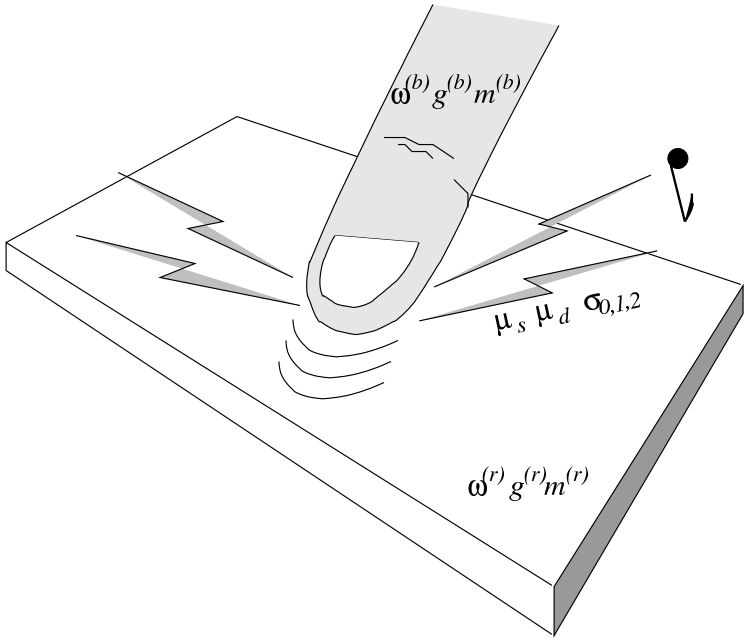


Figure 8.4: Cartoon friction between two modal resonators.

The friction models adopted in the literature of physical modeling of bowed string instruments are typically referred to as *kinetic* or *static* models. Although the two definitions may seem contradictory (kinetic vs. static) at a first glance, they actually refer to the same modeling approach: given a fixed bow pressure, the friction force f is assumed to be a function of the relative velocity only (kinetic models), and the dependence is derived under stationary conditions (static models). An example of parametrization of the steady velocity friction force is given by the following equation:

$$f(v) = \text{sgn}(v) \left[f_c + (f_s - f_c) e^{-(v/v_s)^2} \right] \quad , \tag{8.15}$$

where f_c, f_s are the Coulomb force and the stiction force, respectively, while v_s is usually referred to as Stribeck velocity. Figure 8.5 provides a plot of this function. Note in particular that static models are able to account for the so-called Stribeck effect, i.e. the dip in the force at low velocities. The stiction force is always higher than the Coulomb force, and the term $e^{-(v/v_s)^2}$ parametrizes the slope of the dip in the friction force as the relative velocity v increases.

In recent years, a new class of friction models has been developed and exploited for automatic control applications, where small displacements and velocities are involved, and friction modeling and compensation is a very im-

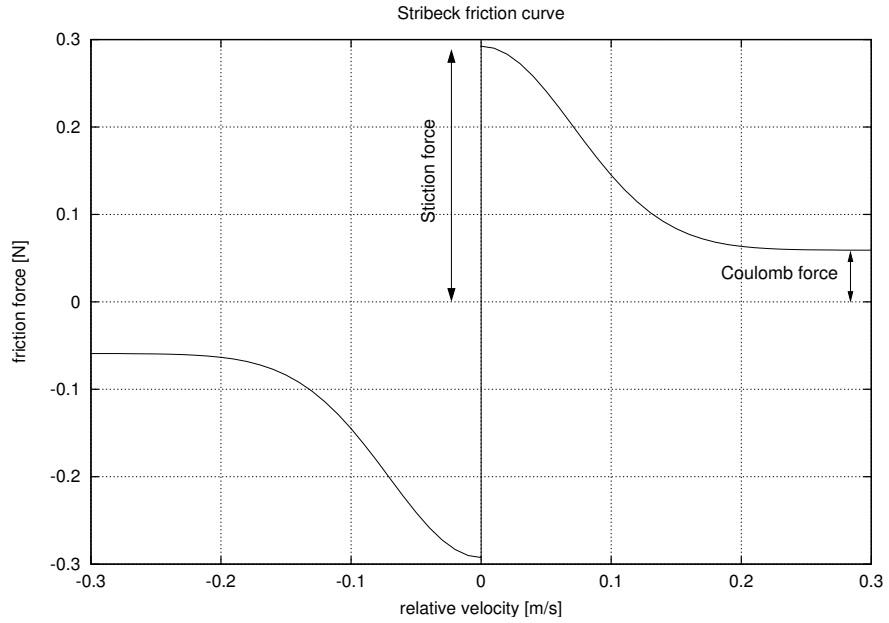


Figure 8.5: The static friction model (8.15), computed with parameters values $\mu_d = 0.197, \mu_s = 0.975, v_s = 0.1, f_N = 0.3$.

portant issue. These are usually referred to as *dynamic* models, since the dependence of friction on the relative sliding velocity is modeled using a differential equation. Being more refined, these models are able to account for more subtle phenomena, one of which is presliding behavior, i.e. the gradual increase of the friction force for very small displacement values. Static and dynamic friction models exhibit the same behavior at high or stationary velocities, but dynamic models provide more accurate simulation of transients [7], which is particularly important for realistic sound synthesis. The difference between static and dynamic models is qualitatively similar to what occurs in reed instrument modeling: it has been shown that dynamic models of the single reed mechanism offer superior sound quality and are capable to reproduce various oscillation regimes found in experiments with real instruments [12].

The first step toward dynamic modeling was proposed by Dahl (see [189] for a review), and was based on the stress-strain curve of classic solid mechanics. This has been later improved by the so-called *LuGre* model[2] which provides a more detailed description of frictional effects [62]. Specifically, friction is interpreted as the result of a multitude of micro-contacts (bristles), as shown in Figure 8.6a. The LuGre model describes this interaction as a single-

[2]The name derives from LUnd and GREnoble, and refers to the two research groups that have developed the model.

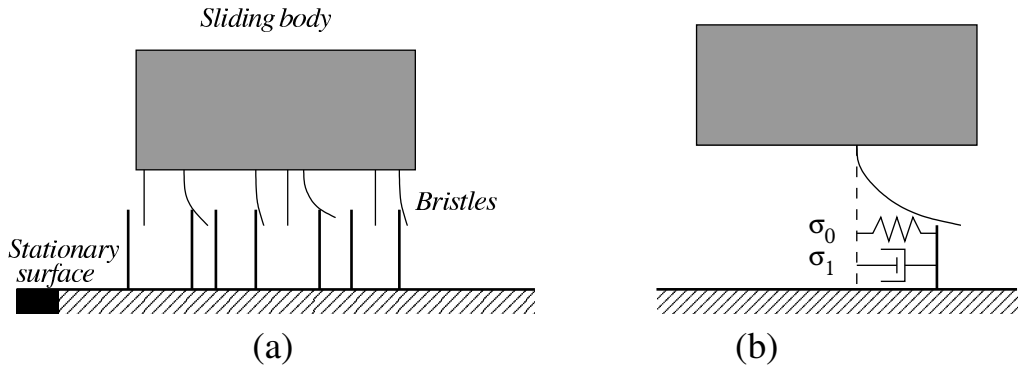


Figure 8.6: The bristle interpretation (a) and the LuGre single-state averaged model (b).

state system which represents the average bristle behavior (as in Figure 8.6b).

One drawback of the LuGre model is that it exhibits drift for arbitrarily small external forces, which is not physically consistent. This effect has been explained in [66] by observing that LuGre does not allow purely elastic regime for small displacements: therefore, a class of *elasto-plastic* models has been proposed in [66], where the drawbacks of LuGre are overcome. These models have been applied in [117] to haptic rendering applications. An alternative extension of LuGre has been proposed in [231], which incorporates hysteresis with nonlocal memory in the non-linear friction force. The elasto-plastic models are going to be used in the remainder of this section, and consequently demand a more detailed description.

The pair of equations

$$\begin{aligned} \dot{z}(v,z) &= v\left[1-\alpha(z,v)\frac{z}{z_{ss}(v)}\right] \quad , \\ f(z,\dot{z},v,w) &= \sigma_0 z+\sigma_1\dot{z}+\sigma_2 v+\sigma_3 w \quad , \end{aligned} \tag{8.16}$$

summarizes the elasto-plastic modeling approach. The first equation in (8.16) defines the averaged bristle behavior as a first-order system: z and $\dot{z}$ can be interpreted as the mean bristle displacement and velocity, respectively, while v is the relative velocity. The second equation in (8.16) states that the friction force f results from the sum of three components: an elastic term $\sigma_0 z$, an internal dissipation term $\sigma_1\dot{z}$, and a viscosity term $\sigma_2 v$ which appears in lubricated systems.[3] A fourth component $\sigma_3 w$ is added here to equation (8.16), which is

[3] As explained in [189], the viscosity term needs not to be linear and may be a more complicated

not part of the original formulation by Dupont et al. [66]. The term $w(t)$ is a pseudo-random function of time which introduces noise in the force signal, and is therefore related to surface roughness (see also section 8.3.3 below).

The auxiliary functions α and z_{ss} can be parametrized in various ways. Here we follow [62] by defining z_{ss} as

$$z_{ss}(v) = \frac{\text{sgn}(v)}{\sigma_0}\left[f_c + (f_s - f_c)e^{-(v/v_s)^2}\right] \quad , \tag{8.17}$$

where f_c, f_s, and v_s are defined as above (see Eq. 8.15), and the subscript ss in z_{ss} stands for "steady-state". As far as α is concerned, we follow [66] by defining it as

$$\alpha(v,z) = \begin{cases} \left.\begin{array}{ll} 0 & |z| < z_{ba} \\ \alpha_m(v,z) & z_{ba} < |z| < z_{ss}(v) \\ 1 & |z| > z_{ss}(v) \end{array}\right\} & \text{if } \text{sgn}(v) = \text{sgn}(z) \\ 0 & \text{if } \text{sgn}(v) \neq \text{sgn}(z) \end{cases} . \tag{8.18}$$

The function $\alpha_m(v,z)$, which describes the transition between elastic and plastic behavior, is parametrized as

$$\alpha_m(v,z) = \frac{1}{2}\left[1 + \sin\left(\pi \frac{z - \frac{1}{2}(z_{ss}(v) + z_{ba})}{z_{ss}(v) - z_{ba}}\right)\right] \quad . \tag{8.19}$$

Therefore the parameter z_{ba} defines the point where α starts to take non-zero values, and is termed *breakaway displacement*.

It is now time to try to make out some sense from these equations. Suppose that a constant relative velocity v is applied, starting from zero conditions.

1. As far as z remains small ($z < z_{ba}$), then $\alpha = 0$ and the first equation in (8.16) states that $\dot{z} = v$. This describes*presliding elastic* displacement: the (mean) bristle deflection rate equals the relative velocity and the bristle is still anchored to the contact surface.

2. When z exceeds z_{ba}, the mixed *elastic-plastic* regime is entered, where $|\dot{z}| < |v|$.

3. After the transient mixed regime, the first-order equation in (8.16) converges to the equilibrium $\dot{z} = 0$, and steady-state is reached with purely

function of the relative velocity.

plastic bristle displacement. Note that $\dot{z} = 0$ means $z = z_{ss}$. It is now clear why z_{ss} (z at steady-state) has been given this name.

Therefore, the steady-state friction force is $f(v) = \sigma_0 z_{ss}(v)$. In other words, at steady-state the elasto-plastic model converges to the kinetic model (8.15). What interests us is the complex transient that takes place *before* steady-state, which is going to provide our friction sounds with rich and veridical dynamic variations.

Using the elasto-plastic model as the coupling mechanism between two modal resonators, the resulting system is

$$\begin{cases} \ddot{x}_i^{(b)} + g_i^{(b)}\dot{x}_i^{(b)} + \left[\omega_i^{(b)}\right]^2 x_i^{(b)} = \dfrac{1}{m_{il}^{(b)}}(f_e^{(b)} + f) \quad , \quad (i = 1 \dots N^{(b)}) \\ \ddot{x}_j^{(r)} + g_j^{(r)}\dot{x}_j^{(r)} + \left[\omega_j^{(r)}\right]^2 x_j^{(r)} = \dfrac{1}{m_{jm}^{(r)}}(f_e^{(r)} - f) \quad , \quad (j = 1 \dots N^{(r)}) \\ v = \sum_{j=1}^{N^{(r)}} t_{mj}^{(r)}\dot{x}_j^{(r)} - \sum_{i=1}^{N^{(b)}} t_{li}^{(b)}\dot{x}_i^{(b)} \\ \dot{z}(v,z) = v\left[1 - \alpha(z,v)\dfrac{z}{z_{ss}(v)}\right] \\ f = \sigma_0 z + \sigma_1\dot{z} + \sigma_2 v + \sigma_3 w \quad \text{(friction force)} \end{cases} , \tag{8.20}$$

where for clarity the two objects have been denoted with the superscripts (b) and (r), which stand for "bow" and "resonator", respectively. The x variables are the modal displacements, while z is the mean bristle displacement. The terms $f_e^{(b)}, f_e^{(r)}$ represent external forces, while the integers $N^{(b)}$ and $N^{(r)}$ are the number of modes for the bow and the resonator, respectively. The relative velocity v has been defined assuming that the interaction occurs at point $l = 1 \dots N^{(b)}$ of the bow and point $m = 1 \dots N^{(r)}$ of the resonator. Note that this system has one degree of freedom (z) more than the impact model given in Eq. (8.14).

8.3.3 Surface texture

Many of the contact sounds we are interested in cannot be convincingly rendered by only using deterministic models. As an example, rolling sounds result from random sequences of micro-impacts between two resonating ob-

jects, which are determined by the profile of the contacting surface[4]. Friction modeling also requires the knowledge of some surface texture, in order to synthesize noisy sliding sounds (see Eq. (8.16) and the $\sigma_3 w$ term). In the remainder of this section we therefore address the problem of surface texture rendering by means of fractal processes. Fractal processes are widely used in computer graphics, since they provide surfaces and textures that look natural to a human eye [193]. Since in physics-based modeling there is direct translation of geometric surface properties into force signals and, consequently, into sound, it seems natural to follow the same fractal approach to surface modeling.

Fractals are generally defined [116] as scale-invariant geometric. They are *self-similar* if the rescaling is isotropic or uniform in all directions, *self-affine* if the rescaling is anisotropic or dependent on the direction, as *statistically self-similar* if they are the union of statistically rescaled copies of themselves.

More formally, a one-dimensional fractal process can be defined as a generalization of the definition of standard Brownian motion. As reported in [202], the stochastic process $x = \{x(t), t \geq 0\}$ is *standard Brownian motion* if

1. the stochastic process x has independent increments;
2. the property

$$x(t) - x(s) \sim \mathcal{N}(0, t-s) \quad \text{for} \quad 0 \leq s < t$$

holds. That is, the increment $x(t) - x(s)$ is normally distributed with mean 0 and variance equal to $(t - s)$;

3. $x(0) = 0$.

The definition of standard Brownian motion can be generalized to the definition of *fractal process*, if the increment $x(t) - x(s)$ is normally distributed with mean 0 and variance proportional to $(t - s)^{2H}$. The parameter H is called *Hurst exponent*, and characterizes the scaling behaviour of fractal processes: if $x = \{x(t), t \geq 0\}$ is a fractal process with Hurst exponent H, then, for any real $a > 0$, it obeys the scaling relation

$$x(t) \stackrel{\mathcal{P}}{=} a^{-H} x(at) \quad , \tag{8.21}$$

where $\stackrel{\mathcal{P}}{=}$ denotes equality in a statistical sense. This is the formal definition for *statistical self-similarity*. The $1/f$ family of statistically self-similar processes, also known as $1/f$ *noise*, is defined as having power spectral density

[4]Rolling sound design is addressed in detail in chapter 9.

$S_x(\omega)$ proportional to $1/\omega^\beta$ for some spectral parameter β related to the Hurst exponent H by $\beta = 2H + 1$. For $\beta = 0$ the definition corresponds to white noise, for $\beta = 2$ Brownian noise is obtained, and for $\beta = 1$ the resulting noise is referred to as pink noise.

The parameter β also relates to the *fractal dimension* [256]. The fractal dimension of a function is related to the roughness of its plot and is exploited in computer graphics to control the perceived roughness [193]. For $1/f$ processes, it is inversely proportional to the Hurst exponent H. Larger values of H correspond to lower values of the fractal dimension and H is proportional to β. Therefore, by increasing β, we will achieve a redistribution of power from high to low frequencies, with an overall smoothing of the waveform.

The problem of generating $1/f$ noise has been treated extensively. One of the most common approaches amounts to properly filtering a white noise source in order to obtain a $1/f$ spectrum. We follow here this approach, and use a model reported in [210] and [55]. The shaping filter is a cascade of N first-order filters, each with a real zero-pole pair. The overall transfer function $H(s)$ in the Laplace domain is the following:

$$H(s) = A\frac{\prod_{i=1}^{N}(s - s_{0i})}{\prod_{i=1}^{N}(s - s_{pi})} \quad , \tag{8.22}$$

where A is a suitable constant.

The fractal noise generator is obtained by properly setting the poles and the zeros of the filters in the cascade [210]. Specifically, the pole and zero frequencies, f_{pi} and f_{0i}, can be computed as functions of the spectral slope β with the following formulas:

$$\begin{aligned} f_{pi} &= -\frac{s_{pi}}{2\pi} = f_{p(i-1)}10^{\frac{1}{h}} \quad , \\ f_{0i} &= -\frac{s_{0i}}{2\pi} = f_{pi}10^{\frac{\beta}{2h}} \quad , \end{aligned} \tag{8.23}$$

where f_{p1} is the lowest pole frequency of the filter. Therefore, the lowest limit of the frequency band for the approximation is f_{p1} and the range width is expressed in decades. The density h (density of the poles per frequency decade) can be used to control the error between the target spectrum and the approximated spectrum obtained by white noise filtering. The dependence of the error with respect to filter pole density is discussed in [55]. Figure 8.7 shows a $1/f^\beta$ spectrum obtained using the filter (8.22), with two different h values.

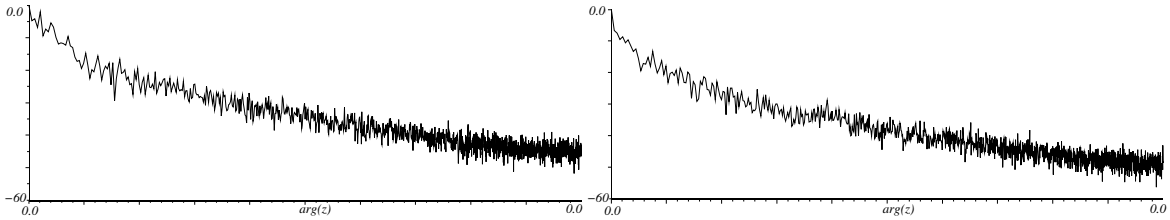


Figure 8.7: Magnitude spectrum of generated fractal noise with $\beta = 1.81$, $h = 2$ (left) and $h = 6$ (right)

The transfer function in the discrete-time domain can be computed with the Impulse Invariant method [177]. It is known that this corresponds to mapping poles and zeros of the transfer function $H(s)$ to poles and zeros of the transfer function $H(z)$ in the discrete-time domain by making the following substitution:

$$s - s_x \rightarrow 1 - e^{s_x T_s} z^{-1} \quad , \tag{8.24}$$

where T_s is the sampling period and s_x stands for a pole s_{pi} or a zero s_{0i}. The following discrete transfer function is then obtained:

$$H(z) = A' \frac{\prod_{i=1}^{N} 1 - e^{s_{0i} T} z^{-1}}{\prod_{i=1}^{N} 1 - e^{s_{pi} T} z^{-1}} \quad , \tag{8.25}$$

where A' is a normalizing constant. In conclusion, the $1/f^{\beta}$ spectrum is approximated by a cascade of first-order filters, each one with the following discrete transfer function:

$$H^{(i)}(z) = \frac{1 + b_i z^{-1}}{1 + a_i z^{-1}} \quad , \quad \text{with} \quad \begin{cases} a_i = e^{-2\pi f_{pi} T}, & b_i = e^{-2\pi f_{0i} T} \\ f_{pi} = f_{p(i-1)} 10^{\frac{1}{h}}, & f_{0i} = f_{pi} 10^{\frac{\beta}{2h}} \end{cases} . \tag{8.26}$$

8.4 Implementation and control

As part of the SOb project activities, the low-level sound models described so far have been implemented as real-time modules written in C language for `pd`[5], the open source real-time synthesis environment developed by Miller

[5] `http://www.pure-data.org/doc/`

Puckette and widely used in the computer music community. The modules have been collected into the `interaction_modal` package, downloadable from the Sounding Object web site [6].

When opening the `interaction_modal` folder, one finds a few subdirectories that reflect the object-oriented structure of the plugins:

`resonators`**:** contains the implementation of resonators described as a bank of modal oscillators, each discretized with the bilinear transformation. External forces can be applied at specified interaction points, each point being described by a set of numbers that weight each mode at that point. Displacement or velocity are returned as outputs from the modal object;

`interactors`**:** for impact and friction interactions, a function computes the forces to be applied to two interacting resonators using the non-linear equations discussed in the previous section. Details about numerical issues are discussed in appendix 8.A below;

`sound_modules`**:** contains a subdirectory for each plugin implemented, where the structures and functions required by `pd` are provided. Here, the external appearance (default parameter values, inlets and outlets) of the plugins is also defined.

One critical issue in physically-based sound modeling is parameter estimation and control. Interaction between the user and the audio objects relies mainly upon a small subset of the control parameters. These are the external forces acting on each of the two objects (which have the same direction as the interaction force). In the case of friction, a third high-level parameter is the normal force f_N between the two objects. The remaining parameters belong to a lower level control layer, as they are less likely to be touched by the user and have to be tuned at the sound design level.

Such low-level parameters can be grouped into two subsets, depending on whether they are related to the resonators' internal properties or to the interaction mechanism. Each mode of the two resonating objects is tuned according to its center frequency and decay time. Additionally, the modal gain (inversely proportional to the modal mass) can be set for each resonator mode, and controls the extent to which the mode can be excited during the interaction. The implementation allows position dependent interaction by giving the option to choose any number of interaction points. A different set of modal gains can be set for each point.

[6] `http://www.soundobject.org`

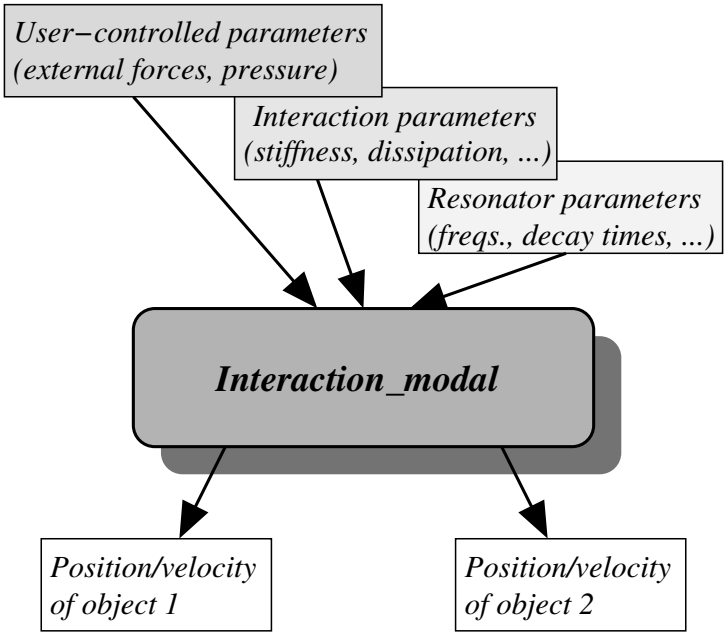


Figure 8.8: Implementation of the interaction models as `pd` plugins: schematic representation of the sound modules.

A second subset of low-level parameters relates to the interaction force specification, as given in Eqs. (8.11) and (8.16). In certain cases typical parameter values can be found from the literature. Alternatively, they can also be found from analysis of real signals. Parameter estimation techniques are the subject of many studies in automatic control, an extensive discussion of such issues is provided in [7].

This hierarchy for the control parameters is depicted in Figure 8.8, where a schematic representation of the `pd` sound modules is provided. The remainder of this section addresses the phenomenological role of the low-level control parameters.

8.4.1 Controlling the impact model

The impact model has been tested in order to assess its ability to convey perceptually relevant information to a listener. A study on materials [10] has shown that the decay time is the most salient cue for material perception. This is very much in accordance with previous results [137]; however, the physical model used here is advantageous over using a signal-based sound model, in that more realistic attack transients are obtained. The decay times t_{ej} of the resonator modes can therefore be used to "tune" the perceived material of the

resonator in a collision with a hammer. See also chapter 4 in this book for more detailed discussion on material perception from recorded and synthesized sounds.

A study on hammer hardness [11] has shown that the contact time t_0 (i.e. the time after which the hammer separates from the resonator) can be controlled using the physical parameters. This is a relevant result, since t_0 has a major role in defining the spectral characteristics of the initial transient. Qualitatively, a short t_0 corresponds to an impulse-like transient with a rich spectrum, and thus provides a bright attack. Similarly, a long t_0 corresponds to a smoother transient with little energy in the high frequency region. Therefore t_0 influences the spectral centroid of the attack transient, and it is known that this acoustic parameter determines to a large extent the perceived quality of the impact [82]. See also an earlier chapter in this book for a detailed discussion on impact sounds and the perceptual role of the spectral centroid of the attack transient.

An equation has been derived in [11], which relates t_0 to the physical parameters of the model:

$$t_0 = \left(\frac{m^{(h)}}{k}\right)^{\frac{1}{\alpha+1}} \cdot \left(\frac{\mu^2}{\alpha+1}\right)^{\frac{\alpha}{\alpha+1}} \cdot \int_{v_{out}}^{v_{in}} \frac{dv}{(1+\mu v)\left[-\mu(v-v_{in}) + \log\left|\frac{1+\mu v}{1+\mu v_{in}}\right|\right]^{\frac{\alpha}{\alpha+1}}} \quad . \tag{8.27}$$

This equation states that the contact time t_0 depends only on v_{in} and two parameters, i.e. the viscoelastic characteristic μ and the ratio $m^{(h)}/k$. Specifically, the ratio $m^{(h)}/k$ is found to be the most relevant parameter in controlling contact time and consequently the perceived hardness of the impact. Numerical simulations have shown excellent accordance between contact times computed using Eq. (8.28) and those observed in the simulations. Figure 8.9 shows an example of soft and hard impacts, obtained by varying m_h/k.

Due to the physical description of the contact force, realistic effects can be obtained from the model by properly adjusting the physical parameters. Figure 8.10a shows an example output from the model, in which the impact occurs when the resonator is already oscillating: the interaction, and consequently the contact force profile, differs from the case when the resonator is not in motion before collision. This effect can not be simulated using pre-stored contact force profiles (as e.g. in [237]). Figure 8.10b shows an example of "hard collision", obtained by giving a very high value to the stiffness k, while the other model

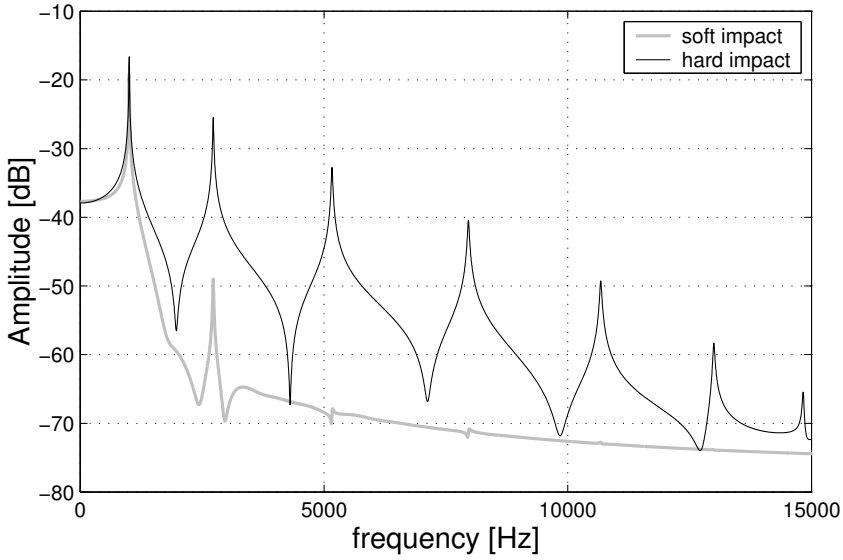


Figure 8.9: Sound spectra obtained when hitting a resonator with a soft mallet (low m_h/k) and with a hard hammer (high m_h/k).

parameters have the same values as in Figure 8.10a. It can be noticed that several micro-collisions take place during a single impact. This is qualitatively in accordance with the remarks about hard collisions by van den Doel et al. [237].

8.4.2 Controlling the friction model [7]

Similarly to impact, the phenomenological role of the low-level physical parameters of the friction model has been studied. The description given in Table 8.1 can be a helpful starting point for the sound designer.

The triple $(\sigma_0, \sigma_1, \sigma_2)$ (see Eq. (8.16)) define the bristle stiffness, the bristle internal dissipation, and the viscous friction, and therefore affects the characteristics of signal transients as well as the ease in establishing stick-slip motion. The triple (f_c, f_s, v_s) (see Eq. (8.17)) specifies the shape of the steady state Stribeck curve. Specifically, the Coulomb force and the stiction force are related to the normal force through the equations $f_s = \mu_s f_N$ and $f_c = \mu_d f_N$, where μ_s and μ_d are the static and dynamic friction coefficients[8]. Finally, the breakaway displacement z_{ba} (see equation (8.18)) is also influenced by the normal force. In order for the function $\alpha(v, z)$ to be well defined, the inequality

[7] section co-authored with Stefania Serafin

[8] It must be noted that treating f_N as a control parameter is a simplifying assumption, since oscillatory normal force components always accompany the friction force in real systems [5].

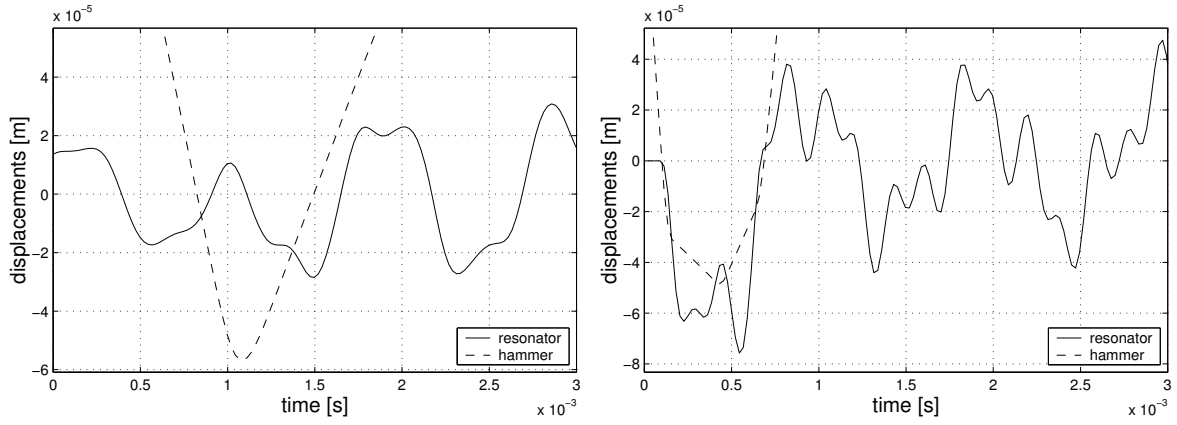


Figure 8.10: Numerical simulations; (a) impact on an oscillating resonator; (b) micro-impacts in a hard collision. Intersections between the solid and the dashed lines denote start/release of contact.

Symbol	Physical Description	Phenomenological Description
σ_0	bristle stiffness	affects the evolution of mode lock-in
σ_1	bristle dissipation	affects the sound bandwidth
σ_2	viscous friction	affects the speed of timbre evolution and pitch
σ_3	noise coefficient	affects the perceived surface roughness
μ_d	dynamic friction coeff.	high values reduce the sound bandwidth
μ_s	static friction coeff.	affects the smoothness of sound attack
v_s	Stribeck velocity	affects the smoothness of sound attack
f_N	normal force	high values give rougher and louder sounds

Table 8.1: A phenomenological guide to the friction model parameters.

$z_{ba} < z_{ss}(v)\ \forall v$ must hold. Since $\min_v z_{ss}(v) = f_c/\sigma_0$, a suitable mapping between f_N and z_{ba} is

$$z_{ba} = cf_c/\sigma_0 = c\mu_d f_N/\sigma_0 \quad , \quad \text{with} \quad c < 1 \quad . \tag{8.28}$$

By exploiting the above indications on the phenomenological role of the low-level parameters, and their relation to user-controlled parameters, simple interactive applications have been designed which use a standard mouse as the controlling device. Namely, x- and y-coordinates of the pointer are linked to the external force $f_e^{(b)}$ and the normal force f_N, respectively. The applications

have been designed using the OpenGL-based `Gem`[9] external graphical library of `pd`.

Braking effects Different kinds of vibrations and sonorities develop within wheel brakes: in the case of rotating wheels slipping sideways across the rails, the friction forces acting at the wheel rim excite transverse vibrations. In the simulation depicted in Figure 8.11a, a wheel driven by the external force $f_e^{(b)}$ rolls on a circular track (a detailed description of rolling sound design is given in the next chapter). When a positive normal force is applied, the wheel is blocked from rolling and the friction model is triggered. Neat stick-slip is established only at sufficiently low velocities, and brake squeals are produced in the final stage of deceleration. The resulting effect convincingly mimics real brake noise.

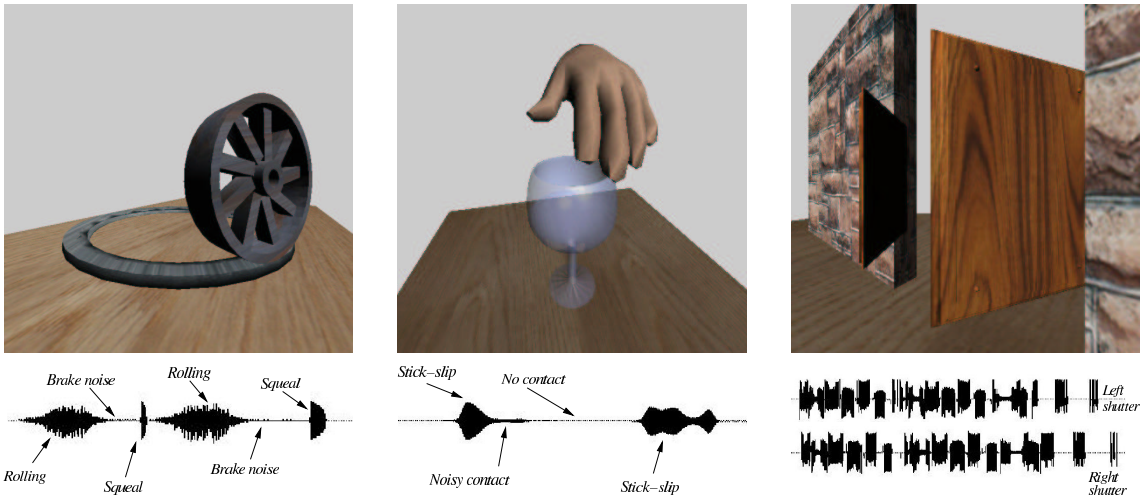
Wineglass rubbing An excitation mechanism analogous to wheel-brake interaction appears when a wineglass is rubbed around its rim with a moist finger. In this case sound radiates at one of the natural frequencies of the glass and its harmonics. By properly adjusting the modal frequencies and the decay times of the modal object which acts as the resonator, a distinctive glassy character can be obtained. In the example depicted in Figure 8.11b, the rubbing finger is controlled through mouse input. Interestingly, setting the glass into resonance is not a trivial task and requires some practice and careful control, just as in the real world.

Door squeaks Another everyday sound is the squeak produced by the hinges of a swinging door. In this situation, different combinations of transient and continuous sliding produce many squeaks which create a broad range of sonic responses. The example depicted in Figure 8.11c uses two exciter-resonator pairs, one for each of the shutters. In this case the modal frequencies of the objects have been chosen by hand and hear tuning on the basis of recorded sounds. The results are especially convincing in reproducing complex transient and *glissando* effects which are typically found in real door squeaks.

8.4.3 Implementation of the fractal noise generator patch

At the time of writing this book chapter the fractal noise generator discussed in section 8.3.3 has not been integrated within the friction model yet.

[9] `http://gem.iem.at/`



(a) (b) (c)

Figure 8.11: Interactive applications; (a) a wheel which rolls and slides on a circular track; (b) a moisty finger rubbing a crystal glass; (c) a swinging door, each of the two shutters is linked to a friction module.

Having done this would (will) allow to control information on surface roughness by controlling the fractal dimension of the noisy component $\sigma_3 w$ in Eq. (8.16).

The fractal noise generator has been implemented independently as a `pd` patch. For convenience in implementation, the shaping filters (8.26) are rewritten as a cascade of *biquads*. Therefore, the cascade is made of $N/2$ second-order filters, each one with the following transfer function (calculated from Eq. (8.26)):

$$\begin{aligned} H^{(i)}(z) = H^{(j)}H^{(j-1)}(z) &= \frac{(1+b_j z^{-1})(1+b_{j-1}z^{-1})}{(1+a_j z^{-1})(1+a_{j-1}z^{-1})} \\ &= \frac{1+(b_j+b_{j-1})z^{-1}+(b_j b_{j-1})z^{-2}}{1+(a_j+a_{j-1})z^{-1}+(a_j a_{j-1})z^{-2}} \\ &\quad \text{with } j = 2\cdot i,\ i = 1\ldots N/2. \end{aligned} \quad , \tag{8.29}$$

The `pd` patch of the fractal noise generator has been developed with a modular approach, for future exploitation in physics-based sound design. The most relevant parameter accessible to the user is β, which defines the target $1/f^\beta$ spectrum. The number of poles of the filtering cascade can be also set, as well as the frequency of the first pole: these parameters controls the accuracy of the $1/f^\beta$ approximation. A snapshot of the patch is given in Figure 8.12.

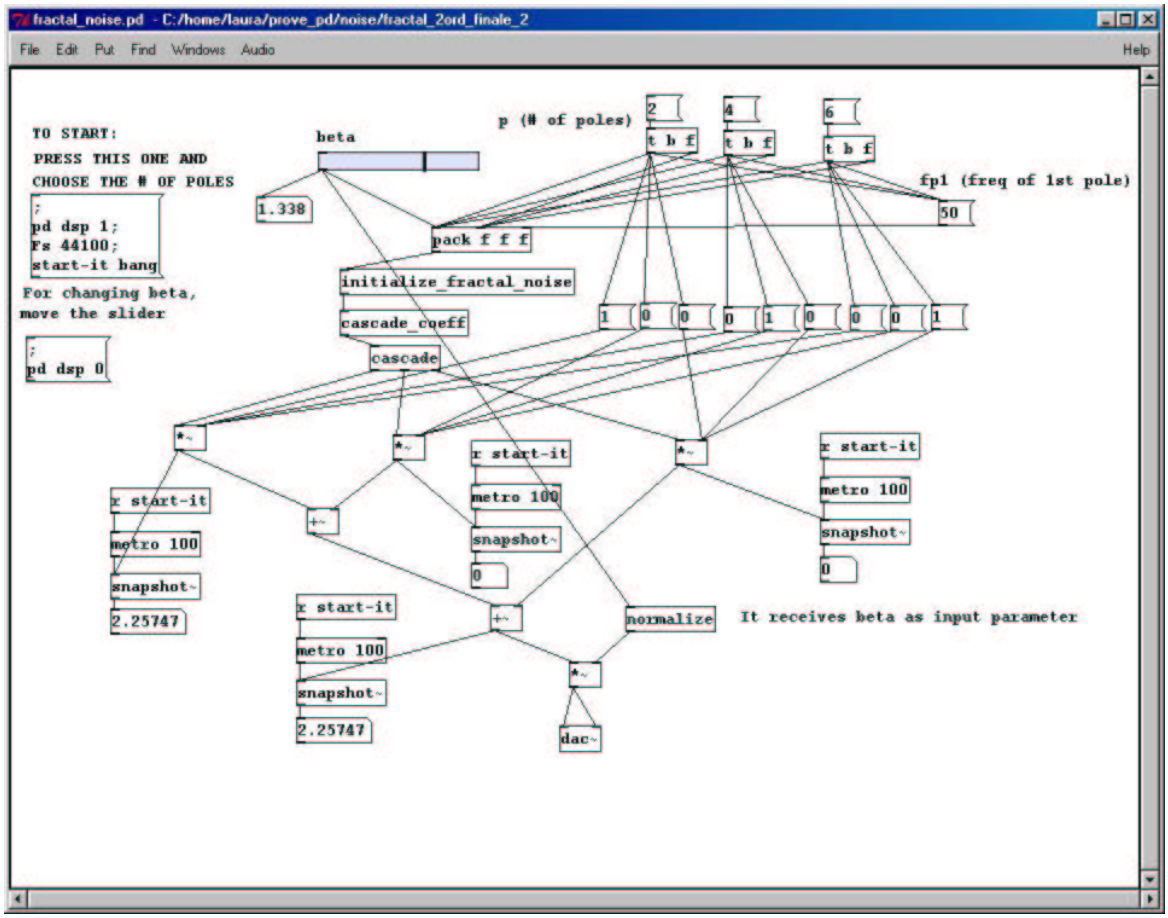


Figure 8.12: `pd` patch of the fractal noise generator.

8.A Appendix – Numerical issues

We have shown that the low-level interaction models, impact and friction, are represented through some non-linear coupling between the resonating objects. When the continuous-time systems are discretized and turned into numerical algorithms, the non-linear terms introduce computational problems that require to be solved adequately. This is the topic of this appendix.

8.A.1 Discretization

As already discussed in section 8.2.3, the discrete-time equations for the modal resonator are obtained by using the bilinear transformation. Recalling Eq. (8.6), the "resonator" object is then described in the discrete-time domain

by the system

$$\begin{cases} \boldsymbol{x}_j^{(r)}(n) = \boldsymbol{A}_j^{(r)}\boldsymbol{x}_j^{(r)}(n-1) + \boldsymbol{b}_j^{(r)}[y(n) + y(n-1)] \\ \boldsymbol{x}^{(r)}(n) = \sum_{j=1}^{N^{(r)}} t_{lj}\boldsymbol{x}_j^{(r)}(n) \\ y(n) = f_e^{(r)}(n) - f(n) \end{cases} , \tag{8.30}$$

for $j = 1 \dots N^{(r)}$, where the matrices are given in Eq. (8.8) and $f(n)$ is either the impact force (8.11) or the friction force (8.16). An identical system is written for the "hammer" and the "bow" objects. The state $\boldsymbol{x}^{(r)}(n)$ has been defined assuming that interaction occurs at point j of the resonator.

But how is the interaction force $f(n)$ computed at each time step?

Impact As for the impact force, the missing equations are simply

$$\begin{cases} \begin{bmatrix} x(n) \\ v(n) \end{bmatrix} = \boldsymbol{x}^{(r)}(n) - \boldsymbol{x}^{(h)}(n) \\ f(n) = f(x(n), v(n)) \end{cases} , \tag{8.31}$$

where f is given in Eq. (8.11). It can be seen that at each time step n the variables $[x(n), v(n)]$ and $f(n)$ have instantaneous mutual dependence. That is, a delay-free non-computable loop has been created in the discrete-time equations and, since a non-linear term is involved in the computation, it is not trivial to solve the loop. This is a known problem in numerical simulations of non-linear dynamic systems. An accurate and efficient solution, called K method, has been recently proposed in [24] and will be adopted here. First, the instantaneous contribution of $f(n)$ in the computation of vector $[x(n), v(n)]$ can be isolated as follows:

$$\begin{aligned} \begin{bmatrix} x(n) \\ v(n) \end{bmatrix} &= \begin{bmatrix} \tilde{x}(n) \\ \tilde{v}(n) \end{bmatrix} + \boldsymbol{K} f(n) \quad \text{with} \\ \boldsymbol{K} &= -\left(\sum_{i=1}^{N^{(h)}} t_{mi}\boldsymbol{b}_i^{(h)} + \sum_{j=1}^{N^{(r)}} t_{lj}\boldsymbol{b}_j^{(r)} \right) , \end{aligned} \tag{8.32}$$

where $[\tilde{x}(n), \tilde{v}(n)]$ is a computable vector (i.e. it is a linear combination of past values of $\boldsymbol{x}_j^{(r)}, \boldsymbol{x}_i^{(h)}$ and y). Second, substituting the expression

(8.32) in the non-linear contact force equation, and applying the implicit function theorem, $f(n)$ can be found as a function of $[\tilde{x}(n), \tilde{v}(n)]$ only:

$$f(n) = f\left(\left[\begin{array}{c} \tilde{x}(n) \\ \tilde{v}(n) \end{array}\right] + \boldsymbol{K} f(n)\right) \quad \overset{\text{K method}}{\longmapsto} \quad f(n) = h(\tilde{x}(n), \tilde{v}(n)) \quad . \tag{8.33}$$

Summarizing, if the map $f(n) = h(\tilde{x}(n), \tilde{v}(n))$ is known, then the delay-free loop in the computation can be removed by rewriting the algorithm as

```
for    n = 1... samplelength
       Assign    f(n) = 0
       Compute   x_i^(h)(n)  (i = 1...N^(h)),
             and x_j^(r)(n)  (j = 1...N^(r))
       Compute   x̃(n), ṽ(n),
             and f(n) = h(x̃(n), ṽ(n))
       Update    x_i^(h)(n) = x_i^(h)(n) + b_i^(h) f(n)  (i = 1...N^(h))
       Update    x_j^(r)(n) = x_j^(r)(n) - b_j^(r) f(n)  (j = 1...N^(r))
end
```

Friction The numerical implementation for frictional contact (8.16) is slightly more complicated, because of the additional degree of freedom z. The dynamic equation for $\dot{z}$ is again discretized using the bilinear transformation. Since this is a first order equation, discretization by the trapezoid rule is straightforward:

$$\begin{aligned} z(n) = &\quad z(n-1) + \int_{(n-1)T_s}^{nT_s} \dot{z}(\tau) d\tau \quad \Rightarrow \\ z(n) \approx &\quad z(n-1) + \frac{T_s}{2}\dot{z}(n-1) + \frac{T_s}{2}\dot{z}(n) \quad . \end{aligned} \tag{8.34}$$

Therefore, the missing equations in the coupled numerical system are

$$\left\{\begin{array}{l} \left[\begin{array}{c} x(n) \\ v(n) \end{array}\right] = \boldsymbol{x}^{(r)}(n) - \boldsymbol{x}^{(b)}(n) \\ z(n) = z(n-1) + \frac{T_s}{2}\dot{z}(n-1) + \frac{T_s}{2}\dot{z}(n) \\ \dot{z}(n) = \dot{z}(v(n), z(n)) \\ f(n) = f(z(n), \dot{z}(n), v(n), w(n)) \end{array}\right. , \tag{8.35}$$

where $\dot{z}(v,z)$ and $f(z,\dot{z},v,w)$ are given in Eq. (8.16). Again, it can be seen that at each time step n the variables $[v(n),z(n)]$ and $\dot{z}(n)$ have instantaneous mutual dependence. Again, the K method [24] will be adopted in order to solve this problem. To this end, the instantaneous contribution of $\dot{z}(n)$ in the computation of vector $[x(n),v(n)]$ must be isolated so that the K method can be applied on the non-linear function $\dot{z}(v,z)$:

$$\left[\begin{array}{c} v(n) \\ z(n) \end{array}\right] = \left[\begin{array}{c} \tilde{v}(n) \\ \tilde{z}(n) \end{array}\right] + \boldsymbol{K}\dot{z}(n) \quad , \tag{8.36}$$

then

$$\dot{z}(n) = \dot{z}\left(\left[\begin{array}{c} \tilde{v}(n) \\ \tilde{z}(n) \end{array}\right] + \boldsymbol{K}\dot{z}(n)\right) \quad \stackrel{\text{K method}}{\longmapsto} \quad \dot{z}(n) = h(\tilde{v}(n),\tilde{z}(n)) \quad . \tag{8.37}$$

where $\tilde{v}$ and $\tilde{z}$ are –as above– computable quantities. From Eq. (8.34), the element $\boldsymbol{K}(2)$ is easily found as $\boldsymbol{K}(2) = T_s/2$, while $\tilde{z}(n) = z(n-1) + T_s/2 \cdot \dot{z}(n-1)$. Finding $\boldsymbol{K}(1)$ is less straightforward, since the friction force itself depends explicitly upon v. Recalling that

$$v(n) = \sum_{j=1}^{N^{(r)}} t_{lj}\dot{x}_j^{(r)}(n) - \sum_{j=1}^{N^{(b)}} t_{mi}\dot{x}_i^{(b)}(n) \quad , \tag{8.38}$$

and substituting here the discrete-time equations (8.30) for $\dot{x}_j^{(r)}(n)$ and $\dot{x}_i^{(b)}(n)$, a little algebra leads to the result

$$\begin{aligned} v(n) = \quad & \frac{1}{1+\sigma_2 b}\left\{\sum_{j=1}^{N^{(r)}} t_{lj}\left\{\dot{\tilde{x}}_j^{(r)}(n) + \boldsymbol{b}_j^{(r)}(2)[f_e^{(r)}(n) - \sigma_0\tilde{z}(n)]\right\} - \right. \\ & \left. - \sum_{i=1}^{N^b} t_{mi}\left\{\dot{\tilde{x}}_i^{(b)}(n) + \boldsymbol{b}_i^{(b)}(2)[f_e^{(b)}(n) + \sigma_0\tilde{z}(n)]\right\}\right\} - \\ & - \frac{b}{1+\sigma_2 b}\left(\sigma_0\frac{T_s}{2} + \sigma_1\right)\dot{z}(n) = \\ = \quad & \tilde{v}(n) + \boldsymbol{K}(1)\dot{z}(n) \quad , \end{aligned} \tag{8.39}$$

where the quantities $\dot{\tilde{x}}^{(r,b)}$ are the "computable" part of the modal velocities (i.e., they are computed from modal resonator equations (8.30)

with $f(n) = 0$), and the term b is defined as

$$b = \left[\sum_{i=1}^{N^{(b)}} t_{mi} \boldsymbol{b}_i^{(b)}(2) + \sum_{j=1}^{N^{(r)}} t_{lj} \boldsymbol{b}_j^{(r)}(2) \right] \quad . \tag{8.40}$$

The quantities $\tilde{v}$ and $\boldsymbol{K}(1)$ are defined in Eq. (8.39) in an obvious way. Having determined the $\boldsymbol{K}$ matrix, the K method can be applied and the algorithm can be rewritten as

```
for     n = 1... samplelength
        Assign      f(n) = 0
        Compute     x_i^(b)(n)  (i = 1...N^(b)),
              and x_j^(r)(n)  (j = 1...N^(r))
        Compute     ṽ(n), z̃(n), and ż(n) = h(ṽ(n), z̃(n))
        Compute     v(n) = ṽ(n) + K(1)ż(n),
              z(n) = z̃(n) + K(2)ż(n), and f(n)
        Update      x_i^(b)(n) = x_i^(b)(n) + b_i^(b) f(n)  (i = 1...N^(b))
        Update      x_j^(r)(n) = x_j^(r)(n) − b_j^(r) f(n)  (j = 1...N^(r))
end
```

8.A.2 The Newton-Raphson algorithm

Two choices are available for efficient numerical implementation of the K method. The first choice amounts to pre-computing the new non-linear function h off-line and storing it in a look-up table. One drawback is that when the control parameters (and thus the $\boldsymbol{K}$ matrix) are varied over time, the function h needs to be re-computed at each update of $\boldsymbol{K}$. In such cases, an alternative and more convenient approach amounts to finding h iteratively at each time step, using the Newton-Raphson method. This latter approach is adopted here. Since most of the computational load in the numerical system comes from the non-linear function evaluation, the speed of convergence (i.e. the number of iterations) of the Newton-Raphson algorithm has a major role in determining the efficiency of the simulations.

Using the Newton-Raphson method for computing h means that at each

time step n the value $h(n)$ is found by searching a local zero of the function

$$g(h) = \begin{cases} f\left(\begin{bmatrix} \tilde{x} \\ \tilde{v} \end{bmatrix} + \boldsymbol{K}h\right) - h & \text{(impact)} \\ \dot{z}\left(\begin{bmatrix} \tilde{v} \\ \tilde{z} \end{bmatrix} + \boldsymbol{K}h\right) - h & \text{(friction)} \end{cases} \quad . \tag{8.41}$$

The Newton-Raphson algorithm operates the search in this way:

$h_0 = h(n-1)$
$k = 1$
`while (err <  Errmax)`
`      Compute` $g(h_k)$ `from Eq. (8.41)`
`      Compute` h_{k+1} `as` $h_{k+1} = h_k - \frac{g(h_k)}{g'(h_k)}$
`      Compute` `err = abs`$(h_{k+1} - h_k)$
$k = k + 1$
`end`
$h(n) = h_k$

Therefore, not only the function $g(h)$ but also its derivative $g'(h)$ has to be evaluated at each iteration. As for the impact, this is found as a composite derivative:

$$\frac{dg}{dh} = \frac{\partial f}{\partial x}\boldsymbol{K}(1) + \frac{\partial f}{\partial v}\boldsymbol{K}(2) - 1 \quad , \tag{8.42}$$

where (recalling Eq. (8.11))

$$\begin{aligned} \frac{\partial f}{\partial x} &= \alpha x^{\alpha-1}\left[k + \lambda v\right] \quad , \\ \frac{\partial f}{\partial v} &= \lambda x^{\alpha} \quad . \end{aligned} \tag{8.43}$$

As for friction, the computation of $g'(h)$ is slightly lengthier. Again, it is done in successive steps as a composite derivative. First step:

$$\frac{dg}{dh} = \frac{\partial \dot{z}}{\partial v}\boldsymbol{K}(1) + \frac{\partial \dot{z}}{\partial z}\boldsymbol{K}(2) - 1 \quad . \tag{8.44}$$

Second step (recalling Eq. (8.16)):

$$\begin{aligned} \frac{\partial \dot{z}}{\partial v} &= 1 - z \left[\frac{(\alpha + v \cdot \partial\alpha/\partial v) z_{ss} - \alpha \cdot v \cdot dz_{ss}/dv}{z_{ss}^2} \right] \quad , \\ \frac{\partial \dot{z}}{\partial z} &= -\frac{v}{z_{ss}} \left[z \frac{\partial \alpha}{\partial z} + \alpha \right] \quad . \end{aligned} \tag{8.45}$$

Third step (recalling Eqs. (8.18, 8.19)):

$$\frac{\partial \alpha}{\partial v} = \begin{cases} \frac{\pi}{2} \cos\left(\pi \frac{z - \frac{z_{ss}+z_{ba}}{2}}{z_{ss} - z_{ba}} \right) \frac{\frac{dz_{ss}}{dv}(z_{ba} - z)}{(z_{ss} - z_{ba})^2} & , \quad (z_{ba} < |z| < z_{ss}) \,\& \\ & \quad (\text{sgn}(v) = \text{sgn}(z)) \\ 0 \quad , & \quad \text{elsewhere} \end{cases} \tag{8.46}$$

$$\frac{\partial \alpha}{\partial z} = \begin{cases} \frac{\pi}{2} \cos\left(\pi \frac{z - \frac{z_{ss}+z_{ba}}{2}}{z_{ss} - z_{ba}} \right) \frac{1}{z_{ss} - z_{ba}} & , \quad (z_{ba} < |z| < z_{ss}) \,\& \\ & \quad (\text{sgn}(v) = \text{sgn}(z)) \\ 0 \quad , & \quad \text{elsewhere} \end{cases} . \tag{8.47}$$

Last step (recalling Eq. (8.17)):

$$\frac{dz_{ss}}{dv} = -\text{sgn}(v) \frac{2v}{\sigma_0 v_s^2} (f_s - f_c) e^{-(v/v_s)^2} \quad . \tag{8.48}$$

Computing these terms from the last step to the first step, the derivative $g'(h)$ can be obtained.

In order to develop a real-time model, it is essential that the number of iterations for the Newton-Raphson algorithm remains small in a large region of the parameter space. To this end, analysis on the simulations has to be performed, where model parameters are varied over a large range. Such analysis shows that in every conditions the algorithms exhibit a high speed of convergence. More precisely, in the case of the impact the number of iterations is observed to be never higher than four, even when the Newton-Raphson algorithm is given extremely low tolerance errors (Errmax$\sim 10^{-13}$). As for friction, the number of iterations remains smaller than seven.