Stefano Zambon

Accurate Sound Synthesis of 3D Object Collisions in Interactive Virtual Scenarios

Ph.D. Thesis

April 23, 2012

Università degli Studi di Verona
Dipartimento di Informatica

Advisor:
prof. Federico Fontana Series N°: **TD-11-12**

Università di Verona
Dipartimento di Informatica
Strada le Grazie 15, 37134 Verona
Italy

a mia madre

Abstract

This thesis investigates efficient algorithms for the synthesis of sounds produced by colliding objects, starting from a physical description of the problem. The objective of this investigation is to provide tools capable of increasing the accuracy of the synthetic auditory feedback in virtual environments through a physics-based approach, hence without the need of pre-recorded sounds.

Due to their versatility in dealing with complex geometries, Finite Element Methods (FEM) are chosen for the space-domain discretization of generic three-dimensional resonators. The resulting state-space representations are rearranged so as to decouple the normal modes in the corresponding equations, through the use of Modal Analysis/Synthesis techniques. Such techniques, in fact, conveniently lead to computationally efficient sound synthesis algorithms. The whole mathematical treatment develops until deriving such algorithms. Finally, implementation examples are provided which rely only on open-source software: this companion material guarantees the reproducibility of the results, and can be handled without much effort by most researchers having a background in sound processing.

The original results presented in this work include:

i efficient physics-based techniques that help implement real-time sound synthesis algorithms on common hardware;

ii a method for the efficient management of FEM data which, by working together with an expressive damping model, allows to pre-compute the information characterizing a resonating object and then to store it in a compact data structure;

iii a time-domain transformation of the state-space representation of second-order digital filters, allowing for the exact computation of dependent variables such as resonator velocity and energy, even when simple all-pole realizations are used;

iv an efficient multirate realization of a parallel bank of resonators, which is derived using a Quadrature-Mirror-Filters (QMF) subdivision. Compared to similar works previously proposed in the literature, this realization allows for the nonlinear feedback excitation of a multirate filter bank: the key idea is to perform an adaptive state change in the

resonator bank, by switching the sampling rate of the resonators from a common highest value, used while processing the initial transient of the signals at full bandwidth, to a set of lower values in ways to enable a multirate realization of the same bank during the steady state evolution of the signals.

Ringraziamenti

Se sono arrivato alla conclusione dei miei studi di dottorato, nonostante alcune difficoltà nel percorso, è grazie alle persone che ho avuto la fortuna di incontrare nel tragitto.

Per il mio revisore Federico Fontana non esisteranno mai ringraziamenti che riflettono quanto mi ha dato. Mi ha convinto ad iniziare il dottorato, mi ha convinto a continuarlo e finirlo con un inesauribile supporto morale oltre che scientifico ed una fiducia immensa.

Ho avuto la fortuna di poter iniziare ad occuparmi di audio digitale grazie a Davide Rocchesso, che mi ha avviato nel mondo della ricerca fin da giovanissimo e mi ha trasmesso l'entusiasmo per un argomento straordinario quale la sintesi a modelli fisici.

Grazie ai revisori di questa tesi per la pazienza nel processo di revisione e per gli esaurienti commenti su di essa. Un augurio rivolto ad entrambi: per uno dei "padri fondatori" dell'elaborazione audio in Italia, Giovanni De Poli, spero che possa vedere i frutti di decadi di lavoro anche nell'affermazione di giovani ricercatori come Stefano Papetti.

Balazs Bank e Gianpaolo Borin non hanno mai smesso di stupirmi per la loro competenza fenomenale, ed è d'obbligo ringraziarli per avermene trasmesso una parte. Oltre all'esperienza lavorativa, le preziosissime informazioni apprese nelle svariate chiacchierate informali non hanno veramente prezzo.

Un ringraziamento un po' amaro per i componenti del gruppo suoni del Dipartimento di Informatica dell'Università di Verona, che purtroppo ha chiuso di recente i battenti. Son convinto che il suo spirito vada avanti nel lavoro di quanti ho avuto modo di incontrare lì e che ora proseguono le loro attività nel mondo. In ordine cronologico, grazie a tutti per avere condiviso questa bella esperienza: Antonio de Sena, Carlo Drioli, Matthias Rath, Pietro Polotti, Stefano Papetti, Stefano Delle Monache, Delphine Devallez, Marco Civolani. Un augurio anche ai "cugini" del gruppo Sound and Music Computing di Padova, in particolare a Federico Avanzini, sperando che possano continuare sempre a mantenere l'alto livello di ricerca degli ultimi anni.

Mauro e Loriana Galanti di Viscount International sono da lodare per il loro sforzo nello sostenere la ricerca a livello di veri e propri mecenati, specie in un mondo imprenditoriale difficile come quello italiano. Grazie per permettermi di

svolgere uno dei lavori più belli che possa immaginare, e grazie anche a tutte le straordinarie persone con cui ho la fortuna di collaborare: Sandro Gabrielli, Maurizio Galanti, Alessandro Larcher, Carmine Cella, Marco Del Fiasco. Una menzione particolare per Eugenio "lo sciamano del suono" Giordani, per la sua dedizione e per il suo atteggiamento iper-critico ma sempre costruttivo.

Ringrazio Vesa Valimaki per l'ospitalità in due periodi diversi 2008 presso la Altoo University di Espoo (Finlandia), e Heidi-Maria Lethonen per la collaborazione e la disponibilità. Grazie anche a Jeremy Copperstock per avermi ospitato nello Shared Reality Lab alla McGill University (Montreal, Canada). Anche se purtroppo non sono stato in grado di finalizzare opportunamente il mio lavoro in tale sede, l'esperienza è stata senza dubbio formativa.

Acknowledgements

If, despite some difficulties along the way, I was able to conclude my doctoral studies, it is mainly for the people that I had the luck to meet in these years.

For my thesis advisor Federico Fontana there will never be appropriate words to thank him for what he gave me. He convinced me to start the Ph.D., he convinced me to continue and finish it with an inexhaustible moral and scientific support and an overwhelming trust.

I was able to start my investigation in digital audio processing thanks to Davide Rocchesso, who groom me in the world of research since I was very young and passed me the enthusiasm for an extraordinary topic such as physical-based sound synthesis.

I would like to thank the reviewers of this thesis for their patience and for the exhaustive comments that they gave me on this work. I hope that a "founding father" of digital audio in Italy, Giovanni De Poli, can see the results of many decades of work also in the affirmation of young researchers such as Stefano Papetti.

Both Balazs Bank and Gianpaolo Borin never ceased to amaze me for their amazing expertise, and I thank them for passing me part of it. What I learned even from informal conversations with you has really no price.

A slightly sad "thank you" goes to all the members of the Sound processing group at the University of Verona, which unfortunately has recently been shut down. Thanks to everyone with whom I shared this experience, and good luck with your current and future carriers: Antonio de Sena, Carlo Drioli, Matthias Rath, Pietro Polotti, Stefano Papetti, Stefano Delle Monache, Delphine Devallez, Marco Civolani. Best wishes also to the "cousins" of the Sound and Music Computing group in Padova, and especially to Federio Avanzini.

Thanks to Mauro and Loriana Galanti of Viscount International for letting me do one of the best jobs in the world, and for their passion in promoting innovation and research despite all the difficulties. I am grateful to work with many extraordinary people: Sandro Gabrielli, Maurizio Galanti, Alessandro Larcher, Carmine Cella, Marco Del Fiasco and, especially, Eugenio "the shaman of sound" Giordani.

I would like to thank Vesa Valimaki for the hosting me at Altoo University of Espoo in 2008, and Heidi-Maria Lethonen for her collaboration in the same period. I am grateful to Jeremy Copperstock for welcoming me at the Shared Reality Lab at McGill University in 2010 and giving me access to his excellent laboratory. Even

if unfortunately I was not able to properly finalize my work there, it was definitely an important learning experience.

Contents

1

Introduction

This thesis deals with the real-time synthesis of sound produced by the collision of generic three-dimensional objects. Auditory feedback in Virtual Reality (VR) environments is considered as one main application, although some of the results contained in this work can be used also for *interactive sonification* [59] or in the development of *virtual musical instruments* [123].

We put the focus on a class of non-speech, non-musical sounds that form an important part of the so-called *everyday* auditory phenomena. These phenomena have been widely studied inside Gaver's *ecological* psychoacoustics [48, 49], which claims that the human brain gathers information on the surrounding environment and objects also through auditory cues of perception.

According to Hahn et al. [57], integrating the visual display with auditory and haptic feedback can provide a higher sense of immersion in the virtual environment compared to rendering only images. These authors list three main problems that should be addressed by sound generation in VR applications. The first one is *sound modeling*, which is seen as the process of synthesizing sounds using a perceptually meaningful mapping between user's actions and synthesis parameters. Then, *synchronization* between sound and other modalities has to be considered, together with proper *sound rendering* techniques for positioning the sounds inside the environment, capable also of taking listening effects into account. Finally, the whole computational process is compared to the image-rendering pipeline used in computer graphics.

The algorithms discussed in this work fall into the category of *physics-based* sound synthesis methods. Inside the research area of *Sound and Music Computing*, which is one of the most recent disciplines in the ACM Computing Classification System [47], the use of such techniques in the context of VR environments has been recently highlighted in an extensive survey [97] as one of the most interesting research topics.

Most of the techniques used for numerical sound synthesis have been developed initially for computer music applications. Many categorizations have been proposed for such techniques, e.g. in [103, 108, 119]. Hereafter, we will simply divide them into two classes: signal-based and physics-based. Techniques which are not explicitly targeted at mimicking acoustical phenomena, e.g. abstract algorithms such as Frequency Modulation (FM), will not be taken into account.

Signal-based models include all the approaches which consider pre-recorded digital audio signals as an essential part of their information content. The most common example is *Pulse Code Modulation* (PCM), which is basically a sample playback technique with possible filtering and/or envelope signal modulation [103]. The digital samples can also be arranged to form loops, that are used as oscillators in more complex synthesis environments. It is common to use the term wavetable synthesis for this case, although in some situations the distinction between the two methods can be hard to be marked. Due to its ease of implementation and generally good sound quality, sample playback is by far the most popular method used today for synthesizing auditory feedback in VR environments. Nevertheless, there are several limitations exhibited by this approach when used in interactive applications. Using a common analogy with graphics rendering [104], pre-recorded sound samples have the same restrictions as digital photos compared to three-dimensional computer graphic models. In order to synthesize sounds, signals must be first recorded from real environments or acquired by other audio generation means. Moreover, the same signals have few possibility to be interactively adapted to the user's input, thus largely failing to convey contextual feedback.

Aside of sample reproduction, *additive synthesis* methods (also called spectral methods) aim at reproducing a target signal through operations in the frequency domain. The most common approach devises a separation between sinusoidal and noisy components [105] within an analysis/resynthesis framework. While offering more flexibility compared to PCM techniques in terms of better time and frequency scaling, yet these algorithms start from a recorded signal, moreover it is hard to map user inputs into corresponding changes of the parameters. Furthermore, the analysis step can be hard to be applied to non-musical sounds, since in this case the sinusoidal components can be very close to each other in the frequency domain.

Physically-based sound synthesis models follow a radically different approach. Here, the sound signal is considered as the final result of a simulation of the object sound source. For this reason, they are often referred to as *source models*, i.e. the goal of the simulation is the mechanical and vibrational behaviour of a given object/instrument. Compared to signal-based techniques, they provide models that can directly reproduce the interaction between the user and a simulated object. This happens because the model parameters are inherently expressed in form of physical variables such as position, mass and velocity. Furthermore, there is no need to have a pre-recorded signal: a simple geometrical description of an object, together with its material properties, contains all the information that is needed by the model to synthesize sounds.

On the other hand, physical models do not generally achieve the same level of realism as signal-based methods do when the goal is to reproduce a specific sound. The main reason for this drawback is due to the limits on the complexity of the models that can be used for real-time simulations running on common hardware. While the theories behind acoustics and vibration are well known and consolidated in the respective sub-fields of physics [84], exceedingly high computational cost for numerical simulation. Thus, the challenge in developing efficient physics-based algorithms resides in finding the trade-off between accuracy, formal simplicity of the sounding objects and computational complexity. The next sec-

tion reviews some algorithms which have been become popular in Digital Signal Processing, for achieving a good compromise among these factors.

1.1 Sound Synthesis by Physical Modeling

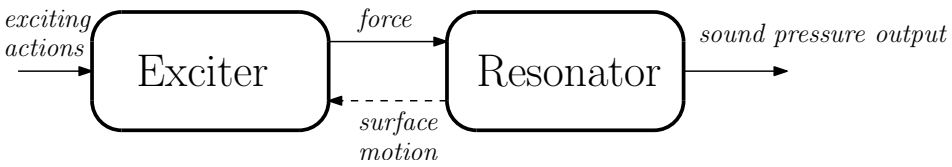

Fig. 1.1. Factorization of a physics-based sound synthesis system in exciting and resonating components, similar to the one presented in [40].

The history of physics-based sound synthesis dates back to 1971, when Hiller and Ruiz first used a numerical approximation of a Partial Differential Equation (PDE) to reproduce the sound of a vibrating string [60]. Since then, much work has been carried out with the goal of developing efficient algorithms furthermore suitable for real-time implementations. Usually, it is not sufficient to merely apply a numerical discretization scheme to a known physical equation. In most cases, the algorithms are partly rewritten in terms of digital filters and other building blocks that are typical in Digital Signal Processing. In this way the synthesis can be combined more efficiently with the rest of the audio system, moreover better perceptual optimizations can be used - we do not need to model what we cannot hear. Thus, investigation in this area involves a wide array of topics: physics, numerical analysis, digital sound processing, real-time programming, human-computer interaction, psychoacoustics.

A relevant part of the literature is devoted to the simulation of musical instruments. The research has advanced at a point of nurturing some commercial high-quality products [36, 83, 114, 115]. Another success story of this research is the contribution to the understanding of acoustics phenomena resulting from digital sound synthesis models. It is the case of e.g. the "phantom partials" in piano sounds, which have been proved to be related to longitudinal string vibration effects by Bank [13]. For a comprehensive treatment of physical-based instrument modeling using DSP techniques, the reader can refer to the excellent book by J.O. Smith [111] or to the review article [123]. On a different perspective, a recent book by Bilbao [18] approaches the subject from the viewpoint of traditional numerical analysis techniques (mostly finite differences methods) for solving complex acoustical systems, involving multidimensional wave propagation and nonlinear vibration.

The design process of algorithms for physics-based synthesis starts from a description of the original system in terms of functional blocks. One popular approach [40] divides an acoustical instrument into an *exciter* and a *resonator* block. The exciter is typically a nonlinear lumped system, which elicits the vibrational

phenomenon by injecting energy into the system. In the field of musical instruments, example of exciters are the hammer (piano), finger/plectrum (guitar), reeds (clarinet), etc. The exciter transmits an excitation force to the resonator block, producing a vibration on its surface which is the source of the variation of the acoustic pressure field at the base of the output sound. The resonator is generally modeled as a linear, spatially-distributed system, described in mathematical terms by a Partial Differential Equation (PDE). The coupling with the exciter block requires that the resonator transmits the information on the motion of its surface, which e.g. for impact-like excitations is given by the displacement and velocity at the contact position. More complex interconnections between different modeling blocks are discussed among others by Rabenstein et al. in [99].

The diverse techniques for physics-based sound synthesis differ mostly in the details of the resonator part. Here, a main distinction can be drawn between *white-box* and *black-box* approaches. White-box approaches start directly from the discretization of PDEs, exposing the internal details of the numerical simulation. Common examples are waveguide modeling [109] (for one-dimensional resonators such as strings and pipes), time-domain finite difference methods [18] and waveguide meshes [17] (for multidimensional resonators like membranes, plates and rooms), and some of the modal-based approaches discussed in the next paragraph.

Black-box techniques rely instead on a simpler mathematical description of the resonating system. The resonator is modeled as a lumped system, thus losing the information on the spatial properties of the interaction. In most cases, the parameters of the resonator block are derived using signal-analysis techniques like sinusoidal component extraction from a pre-recorded sound. It has become common to refer to such techniques as *Physically-Inspired* (or physically-informed) synthesis methods [33,35,123], since they are somehow a hybrid between signal and physics-based modeling. Another common simplification within these approaches is the absence of the feedback from the resonator to the exciter. Instead of being a non-linearly coupled system, the exciter is also modeled using signal-based methods, similarly to what happens in *source-filter* synthesis [103].

Physical modeling for the simulation of everyday sounds was first proposed by Van den Doel [124, 125], who concentrated his efforts on the modeling of colliding objects. Since then, much work has been done e.g. in improving non-linear contact models [7], using different kinds of excitations such as friction [8] and integration with haptics output [5]. A milestone in the definition of the research field has been the EU-funded project *The Sounding Object* [104], where simplified although highly-interactive multimodal rendering systems have been described using an analogy with cartoon-like sounds.

Some recent high-quality results in accurate modeling of generic, non-musical resonators has come from researchers working in the computer graphics community, where the use of physical models for interactive rendering has a long history. Examples include accurate sound-pressure radiation modeling [66, 87], nonlinear high-amplitude vibration simulation [29], precise contact modeling [138] and simulation of fracture phenomena [137]. They all follow a white-box approach, where every detail of the algorithm is derived from the physical equations describing the problem. Nevertheless, the computational cost of such techniques is often too high for real-time simulations on common hardware, and the models themselves are

presented in a way which is hard to understand from researchers with usual audio background.

1.1.1 Modal Synthesis

Almost all the methods for synthesizing non-musical sound objects use some modal based numerical scheme. Using a term coined by Adrien [3] in the audio signal processing literature, a sound synthesis technique whose final implementation consists of a parallel bank of second-order digital resonators is conventionally referred to as *modal synthesis*. However, there are radical differences among the methods regarding the procedure used for computing the parameters of the resonators, i.e. the resonance frequencies, decays and amplitudes. When the synthesis engine simply consists of a parallel bank of many simple components, the complexity of the technique resides more in the computation of these micro-parameters than in the synthesis itself. From here on, we propose to categorize the various modal synthesis techniques depending on the nature of this computation:

1 Signal-based techniques With these approaches, all the parameters belonging to every resonator are computed from the analysis of a pre-recorded sound. Modal extraction techniques that have been used include, among others, Linear Predictive Coding [35], ARMA system identification made upon on heterodyne-modulated signals [70], or complex transform-based methods using Wavelet [39] or Gabor [107] representations.
It should be clear that, while being often improperly cited as physically-based, these techniques are mostly signal-based. For example, it is not possible to properly compute physical quantities such as displacement, force or velocity with these approaches. Moreover, we do not have any access to the spatial information about the resonator, since a lumped representation is instead used according to a black-box like approach.

2 Diagonalization of a linear system This traditional approach in *Modal Analysis* [58] will be chosen also in the following of this thesis. It starts from a system of Ordinary Differential Equations with second-order time derivatives, which comes out of the spatial discretization of a specific PDE describing the elastic motion of the object under study. Various numerical techniques can be used for the derivation of this system: common approaches are based on mass-spring network analogies [27, 104] and Finite Element Modeling [26, 66, 88]. Also finite differences or basically any other numerical discretization method that leads to a linear system, second-order in time, with symmetric, positive-definite matrices can be used, although they are less common in literature.
However such a discrete system of ODEs is derived, it is possible to decouple the equations into *normal modes* by diagonalizing the system matrix. This decoupling makes possible to solve N independent ODEs describing corresponding orthogonal normal modes, instead of a coupled system of order N. The diagonalizing matrix acts as a transformation between physical variables defined e.g. in Cartesian coordinates, and modal variables in the space spanned by the normal modes. As far as we are interested only in the value of few physical variables, like sound pressure or force and displacement at a single contact

point, the number of such transformations that need to be applied is often proportionally low compared to the order of the system. Therefore, modal analysis algorithms can provide a significant computational advantage over the direct solution of the initial linear system of equations.
The accuracy and flexibility of the final algorithms depend on the discretization technique used for the derivation of the coupled linear system. Finite Element Methods are proposed in this work, in Chapter 2, due to their ease to treat complex geometries compared to finite differences and mass-spring networks.

3 Spectral Methods Another white-box approach prescribes to use the analytical solutions of a PDE, typically in terms of Fourier (or related) series expansions. These techniques are called *spectral methods* in the literature of numerical analysis [25]. Compared to the above-mentioned diagonalization, spectral methods do not require to discretize the spatial domain. This happens because the continuous-time equations are solved entirely with an analytic procedure. In practical implementations these solutions need to be discretized, but their accuracy is much higher compared to that achieved by direct discretization of the equation itself[1]. The drawback of spectral methods is that they can be applied only when analytical solutions of the PDE can be obtained: this constraint typically implies strong assumptions on the domain geometry (e.g. rectangular or circular shapes), on the homogeneity of the material, and on the boundary conditions.
Spectral methods have been employed for sound synthesis of piano strings [14], rectangular reverberation plates [133] and circular membranes [6]. A comprehensive sound synthesis theory based on spectral methods is the *Functional Transformation Method* (FTM) [120], where similar examples of acoustic systems having simple geometry have been examined.

It is important to notice that, as in almost any categorization system, there are some exceptions which do not fall exactly in any of these three classes. In practical applications one might start from either white-box approach, and then modify some parameters using signal-based methods. This is the case of the damping parameters, that will be discussed in detail in the next chapter. Another example is the perceptual adjustment of physically-computed amplitudes in virtual musical instruments modeling [134, 135].Sometimes, the theory underpinning some methods is classified as white-box and then, in practical applications, resorts to signal-based modal extraction methods for the sake of simplicity [104, 125, 129].

Some software packages that employ modal synthesis techniques for sound synthesis of generic objects are nowadays available. Commercial products include *Modalys*, developed at the IRCAM [43], and the recent musical synthesizer *Cromaphone* by Applied Acoustic Systems [118]. Modalys employs both Finite Element Methods and spectral methods (e.g. for strings). Observing its functionalities, Cromaphone is probably based on spectral methods due to the simple geometries available (strings, rectangular plates, circular membranes). Open-source alternatives are the virtual reality environment *Phya* [80], which interfaces with a rigid body engine for graphics rendering, and the *Sound Design Toolkit (SDT)* [129],

[1] Due to their high accuracy, spectral methods are often said to have $\mathcal{O}(\infty)$ convergence rate compared to typical $\mathcal{O}(1), \mathcal{O}(2)$ orders of convergence of traditional methods.

developed mainly at the University of Verona and implementing many interaction primitives (e.g. rolling, crumpling) upon simple contact modeling. However, both of these open-source alternatives use a signal-driven approach for the modeling of the resonator, without any possibility to derive the modal parameters from the physical analysis but for trivial resonators.

1.2 Thesis Outline

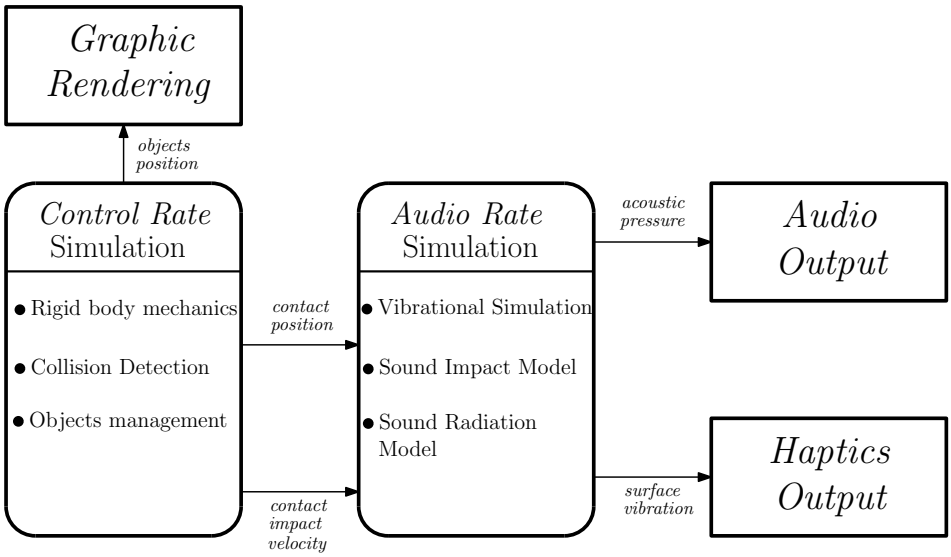

Fig. 1.2. High-level block diagram showing the modules of a physics-based engine for multimodal rendering.

In this thesis, a white-box framework for the real-time synthesis of generic three-dimensional objects is presented. The modeling process require only a geometrical and material description of the objects and do not rely on pre-recorded sound signals.

The sound synthesis algorithm is supposed to be part of a larger system for multimodal rendering in virtual reality environments. A generic block-diagram representation of this system, which mimics and extends the architecture of similar applications developed in the same context [76, 80, 125], is shown in Fig. 1.2. The rigid body motion of the objects is carried out using an engine available off-the-shelf, developed for computer graphics applications [19]. This simulation typically runs at a sampling rate between 30 and 200 Hz, as slowly as so-called user *control rates* operate in many audio systems. Contacts between different objects are detected using a collision detection algorithm [75], and information on contact position and impact velocity are transmitted to the sound synthesis engine.

The role of the sound synthesis block, which runs at a much higher sampling frequency (*audio rate*), e.g. 44100-48000 Hz, is to simulate the vibrational be-

haviour of the objects. This simulation implies the resolution of contacts at audio rate and the generation of the sound pressure signal, which can be sent to the audio output eventually preceded by some spatialization and/or postprocessing. The same vibrational information can also be used for haptic rendering, by properly sending surface forces to suitable interfaces.

The space-domain discretization of complex-shaped three dimensional objects is treated in Chapter 2 using the Finite Element Method. The exposition does not require much background in solid mechanics and/or FEM modeling. Moreover, only techniques which can be practically reproduced using open-source software are considered. The goal is to guarantee a certain degree of reproducibility of the research by people possessing an audio processing background. Most of the high-quality sound synthesis methods recently proposed by graphic researchers (e.g. [29,66]) instead require a strong background in advanced numerical methods, and are often impractical to implement without the use of expensive commercial software for the pre-processing stage. In the same chapter, novel efficient techniques for modeling the damping properties and for reducing the memory requirements in real-time contexts are proposed.

Chapter 3 deals with the time-domain discretization of the normal modes, using the well-known realization of the second-order digital resonator. As an original result, a run-time transformation between two common forms of resonators is presented, with applications for efficient computations of derived physical variables like velocity, instantaneous amplitude and residual energy.

Chapter 4 briefly discuss some variants that can be used for lumped nonlinear contact modeling between two objects. The computational aspects of the whole system are instead described in Chapter 5. The core is a *multirate* signal processing framework for modal synthesis, based on a Quadrature-Mirror-Phase (QMF) subdivision. Its originality resides in the ability to handle nonlinear feedback within a multirate signal processing architecture. In order to do so, an adaptive system is described which uses a high fixed sample rate during the contact phase and then switches at run-time, without audible artifacts, to an efficient multirate implementation which is active for the free evolution of the system. Since the average duration of the contacts is usually very short (less than 10 milliseconds), the overall computational cost is significantly reduced.

Finally, Appendix A describes some software applications, working both offline and in real-time, that have been developed as a addendum to this thesis. They consist of tools for computing modal resonators parameters using an open source software for FEM computations, a graphical application for the analysis of the resulting resonators, a real-time application where a rigid-body simulator is connected with a real-time audio synthesis engine, and a fast realization of the second-order parallel bank using vector and parallel computing techniques.

2

Physical Modeling of 3D Resonators

This chapter focuses on the modeling of the resonator block through the generic physics-based procedure that has been described previously (see Fig. 1.1. In summary, the problem can be formulated as to compute the displacement of the vibrations in the resonant object by known forces acting on its surface. Once the dynamical displacement of the surface is known, we have all the information needed to interface the resonator with the exciter with an appropriate contact model. From here, we can produce the output sound by modeling the radiation properties of the object.

In particular, the modeling of generic resonant objects can be simplified by making the following assumptions:

In order to simplify the modeling for the specific case of generic sounding objects, we will make the following assumptions:

- Only shell objects are considered, i.e. we will limit our investigation to the object surface motion.
- Deformations and stress are small enough, such that the law of elasticity will be linear. Nonlinearities in object motion depend on nonlinear elastic properties (quite rare, and not leading to vibrations bringing important audible effects) or geometric properties (affecting high-amplitude deformations). The latter are more relevant in terms of their consequences to the sound, although their accurate reproduction increases significantly both the complexity and the computational cost of the model.
- The material is isotropic, i.e. the stress-deformation relationships will be independent of the direction considered. In nature, metals and other stiff materials (ceramic, glass) are usually isotropic with good approximation. Other materials, like wood, can be highly anisotropic - this fact must be taken into account for instance when accurately modeling soundboards, like those onboard good pianos [54].

There are many choices that can be made while selecting among different mathematical tools, concerning both the continuous-time equations and the numerical discretization algorithms. Here we describe in detail the modeling approach that is at the origin of this thesis work. Although focusing on one of the many possible derivations capable of figuring out a generic resonator model, the proposed

solution satisfies most of the requirements previously introduced, that are needed for a successful inclusion of the model in an effective virtual reality environment. Specifically, the geometry and material characterizing an object are sufficient for providing a complete description of a generic resonator, apart from the fine-tuning of the decay parameters.

Moreover, the specific type of finite element approach has the valuable property of being sufficiently simple to interface with, and to be supported by many software packages. One of the reasons why FEM are rarely used for physics-based sound synthesis is that the amount of code that needs to be written to implement them can be enormous. Another difficulty comes from the fact that researchers in digital audio usually miss a solid background in solid mechanics, or finite element techniques. Therefore, a sufficiently simple method furthermore widely supported by available software packages is fundamental to improve the reproducibility of the research.

2.1 Modeling of Thin Shell Structures

In this work, we focus our attention on objects that can fall under the category of *thin shell structures*, i.e. shells in which the thickness is relatively small compared to the radius of curvature. The motivations behind this choice are basically of computational nature. In fact, material points on a shell can be described using only two coordinates instead of three, thus leading to a significant reduction of the computational domain. Moreover, we are only interested in the motion on the surface of the object. Thus, avoiding to compute anything in the inside is an additional advantage. For these reasons, along with the fact that shells exhibit good properties as structural elements, the subject has been widely studied, mainly in various fields of mechanical engineering [113, 140]. Other objects such as beams and plates can be treated as a particular case of shells of zero curvature [113].

By choosing only shell-like bodies we are making a quite hard restriction on the kind of simulated objects. However, probably this is not a critical problem, since most fully-solid objects usually show a strong damping behavior [58] and thus their sound-production mechanism is not very interesting from a perceptual point of view. Even more, a full three-dimensional treatment would be much more complicated and will eventually cause numerical problems for shell-like objects (see [139], chap. 6 and [140], chap. 8).

The motion of a shell is typically defined by its undeformed shape (also called rest shape or neutral surface) and by a set of material parameters that define how it deforms under applied forces. Locations on the neutral surface can be defined by two-dimensional curvilinear coordinates ξ_1 and ξ_2. Thus, three-dimensional Cartesian coordinates of the points on the surface can be described by a vector field $\mathbf{x}_0(\xi_1, \xi_2)$. When forces are applied, the object deforms and the new shape is described by a deformed vector field $\mathbf{x}(\xi_1, \xi_2)$. Usually, the deformation is specified by the *displacement* vector field $\mathbf{u} = \mathbf{x} - \mathbf{x}_0$ (see Fig. 2.1).

From the displacement $\mathbf{u}$ the elastic strain ε is computed in terms of the spatial variations of $\mathbf{u}$. Under the hypothesis of "small" displacements, a linear relationship between strain and displacement can be considered. This assumption is needed

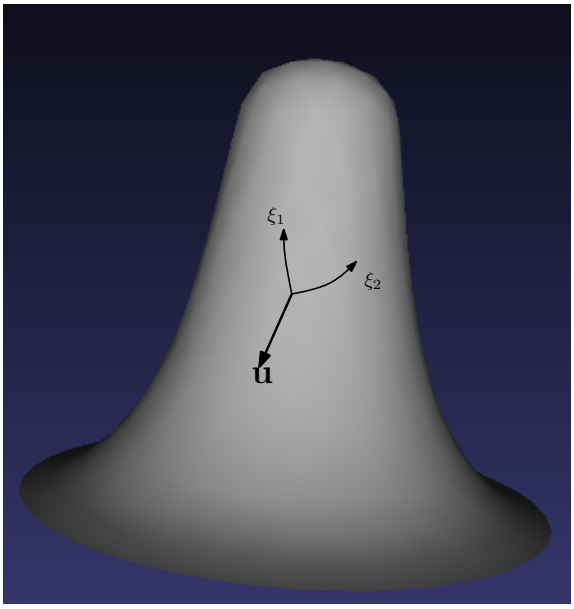

Fig. 2.1. Two-dimensional curvilinear coordinates ξ_1, ξ_2 on a generic shell surface immersed in a 3-dimensional space. The displacement function $\mathbf{u}(\xi_1, \xi_2)$ is computed on this coordinate system.

in order to derive a linear description of the system. It may be limiting if high-amplitude excitations have to be considered. Another physical hypothesis that we will consider is linear elastic behavior, i.e. Hooke's law applies. This results in a linear dependence of the stress σ from the strain ε. In our case of interest, this assumption is not as much critical as the neglect of geometrical nonlinearities [84].

Partial differential equations describing the motion of shells are usually derived by applying *Hamilton's principle* after the imposition of further simplifications. Popular choices are Rayleigh's *membrane approximation* and *bending approximation*, which correspond to neglect respectively out-of-plane and in-plane strains. They can be both seen as a particular case of Love's equations, which are commonly used in many engineering situations [113]. Other types of shell equations are discussed in reference textbooks as [73,113]. Unfortunately, analytical solutions to these equations can not be derived, apart from very simplified cases (such as those using Rayleigh's approximations) and only for trivial domains (e.g., beams, spheres, etc.).

Therefore, numerical approximations of the equations are needed. While it is possible to discretize directly the equations of shell vibration, for example applying finite differences [116], one common approach in engineering is to use the *explicit* Finite Element Method (FEM) [139, 140]. With this method, the domain is discretized into a finite number of disjoint elements, and a set of ordinary differential equations (ODEs) is derived for each element. The process is summarized in the following section.

2.1.1 Finite Element Modeling

Finite Element Methods (FEM) are a class of numerical discretization techniques that are particularly suited to handle problems where the domain exhibits a complex, irregular geometry, often coupled with varying boundary conditions. They have arisen as *de facto* standard in fields such as structural mechanics and can be used for a large variety of physical problems, from heat conduction to fluid dynamics.

FEM are used to find an approximation for a continuos function that satisfies some equilibrium expressions, typically derived from a variational principle. Many reference numerical analysis books (see for example [98]) present the Finite Element Method in a way that is similar to other traditional solving techniques such as finite differences:

1. A partial differential equation is given in both analytical and continuous-time domains, together with initial and boundary conditions
2. The equation is rewritten in its *weak* formulation, which can be summarized very briefly as a version of the problem where the solution is no longer needed to satisfy the initial continuity properties. In this way, we can approximate the solution with a set of separate functions defined on a compact support (finite elements), so that the overall result can admit e.g. discontinuities in the first derivative along the element borders.
3. The weak formulation of the problem is solved by discretization of the spatial domain and the Galerkin method is appliedis applied to derive a system of (usually linear) equations describing the value of the solution in each element.

This approach, although being coherent with the majority of other discretization techniques for partial differential equations, has the drawback that the PDE has to be defined first on the continuos domain. From a practical point of view, this can pose serious drawbacks since the spatial discretization of a generic surface is not trivial, and typically involves methods such as Delaunay triangularization [62] or other mesh-generation techniques. Most of the software packages [65,117] which perform FEM computations usually assume the spatial domain to be given in some geometric mesh format. Therefore, we would like to have a mathematical formulation where the equations can be directly derived from a pre-existing subdivision of the geometrical domain. Fig. 2.2 shows an example triangularization of a smooth shell-like surface.

FEM pioneer O.C. Zienkiewicz developed a mathematical formulation of the Finite Element Method for elasticity problems, often called “direct approach” or “explicit finite element method” [139], where the physical equations are derived directly on the nodes of the mesh. Therefore, this technique is particularly suitable for practical software implementations where the geometry of the domain has already been meshed into the desired elements. In the following, a brief overview of this approach for elastostatic problems is presented and it is subsequently shown how it is possible to simulate the dynamical properties of the object.

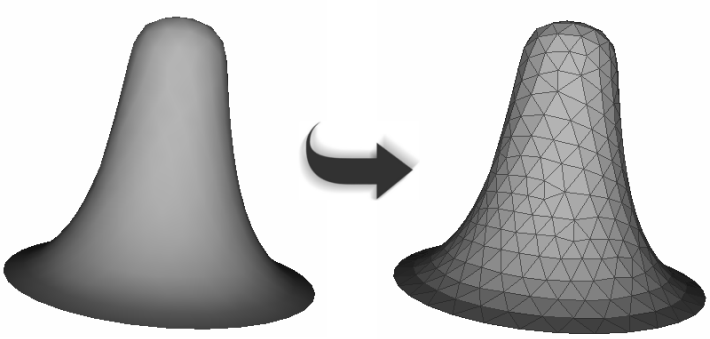

Fig. 2.2. Example of surface subdivision into triangular elements (*triangularization*).

Elastostatics Problems

In the direct approach to the Finite Element Method the continuum, which corresponds to the object surface in our case, is divided into a finite number of elements joined at discrete node points. Taking a look at a two-dimensional example (Fig. 2.3), the displacement $\mathbf{u}$ at any point within the element e can be approximated as a column vector, $\tilde{\mathbf{u}}$:

$$\mathbf{u} \simeq \tilde{\mathbf{u}} = \sum_k \mathbf{N}_k \mathbf{u}_k^e = \mathbf{N}\mathbf{u}^e, \tag{2.1}$$

in which $\mathbf{u}^e$ represents a listing of nodal displacements for a particular element and the components of $\mathbf{N}$ are prescribed functions of position.

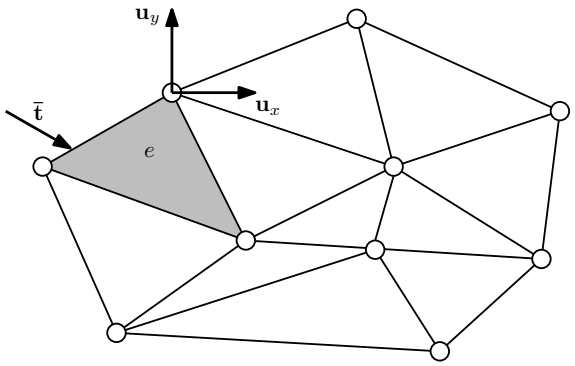

Fig. 2.3. Triangular finite elements on a planar (2D) domain. The components of the displacement $\mathbf{u}$ are shown for an example node, together with an external force $\bar{\mathbf{t}}$ acting on the element border.

The functions $\mathbf{N}_k$ are usually called *shape functions* (or interpolation functions) and their value must be 1 at node k and 0 at all other nodes (Kronecker Delta property). One of the three linear shape functions defined on a planar triangular

element is shown in Fig. 2.4. The elements of the vector $\tilde{\mathbf{u}}$ are usually named *degree of freedoms* (DoF) in standard vibration analysis [58].

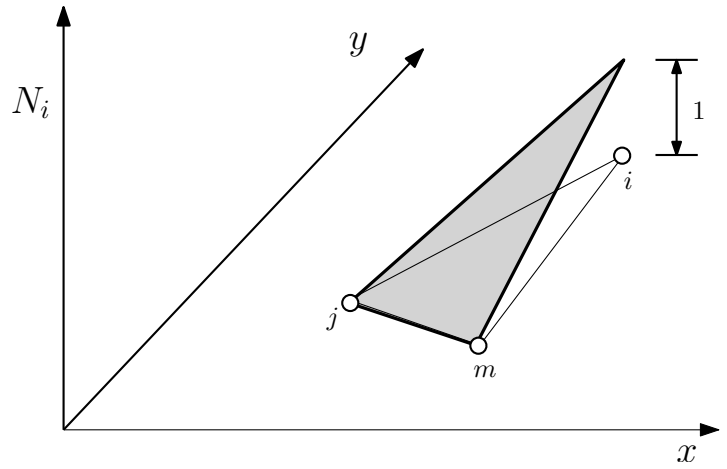

Fig. 2.4. Shape function for a triangular element. It has unitary value on the i-th node and null value on the other nodes.

With the small deformation assumption, strains at any point inside the element are then determined as

$$\varepsilon \simeq \tilde{\varepsilon} = \mathbf{S}\mathbf{u}, \tag{2.2}$$

where $\mathbf{S}$ is a suitable linear operator. Finally, assuming general linear elastic behavior (i.e., Hooke's law applies), the relationship between stresses and strains is also linear and, neglecting initial strains and stresses, can be expressed as

$$\sigma = \mathbf{D}\varepsilon \tag{2.3}$$

where $\mathbf{D}$ is an elasticity matrix, symmetric in the isotropic case, which contains the appropriate material properties, i.e. Young's modulus and Poisson's ratio.

The variational principle that is used to derive equilibrium conditions is formulated in terms of the total potential energy $\mathit{\Pi}$, which is a functional given by the expression

$$\mathit{\Pi} = \mathit{\Lambda} - W \tag{2.4}$$

where $\mathit{\Lambda}$ is the potential strain energy, that can be computed as an integral over the element volume V^e:

$$\mathit{\Lambda} = \int_{V^e} \varepsilon^T \sigma dV \tag{2.5}$$

and W is the total work done by external loads on the object:

$$(\mathbf{u}^e)^T \mathbf{r} + \int_{V^e} \mathbf{u}^T \mathbf{b} dV + \int_{A^e} \mathbf{u}^T \bar{\mathbf{t}} dA. \tag{2.6}$$

In the last expression, $\mathbf{r}$ is a vector of loads concentrated at node points, $\mathbf{b}$ are distributed body forces acting on a unit volume of material and $\bar{\mathbf{t}}$ is a distributed external loading acting on a unit area of the element.

Substituting Eq. (2.5) and (2.6) in Eq. (2.4) and imposing the condition that the total potential energy functional must be stationary for variations of admissible displacements, it is finally possible to derive the equations of equilibrium for the node displacements $\mathbf{u}^e$:

$$\mathbf{K}^e\mathbf{u}^e = \mathbf{f}^e \tag{2.7}$$

where $\mathbf{K}^e$ is denoted as the element *stiffness matrix* and can be computed as

$$\mathbf{K}^e = \int_{V^e} \mathbf{B}^T\mathbf{D}\mathbf{B}, \qquad \mathbf{B} = \mathbf{S}\mathbf{N} \tag{2.8}$$

and, similarly, the element force vector $\mathbf{f}^e$ is expressed as

$$\mathbf{f}^e = \mathbf{r} + \int_{V^e} \mathbf{N}^T\mathbf{b}dV + \int_{A^e} \mathbf{N}^T\bar{\mathbf{t}}dA. \tag{2.9}$$

Clearly, the computation of both the element stiffness matrix and force vector requires the use of numerical integration techniques, apart from the case of concentrated loads $\mathbf{r}$, which luckily is the most useful in our situation as will be discussed in Sec. V.

The same reasoning that led to Eq. (2.7) can be applied to the whole structure, thus leading to the equation for the displacement of the nodes in every element:

$$\mathbf{K}\mathbf{u}^g = \mathbf{f}. \tag{2.10}$$

In this case, the stiffness matrix $\mathbf{K}$ is assembled from the contributions of each single element and is generally sparse and banded. Therefore, efficient numeric techniques can be applied for the numerical solution of Eq. (2.10) [56]. Direct banded solvers employing Cholesky factorization can be applied on current entry-level hardware for problems size up to around 50,000 elements. Above this limit, the main bottleneck is usually found in memory requirements and iterative solvers have to be used, such as the *biconjugate gradient method* (BiCG) or the *generalized minimal residual method* (GMRES), eventually preceded by a preconditioning stage on the system matrix.

The resolution of the mesh is usually defined by a parameter h, which is the largest radius of an element in the mesh. In audio applications, it is important to choose h sufficiently small to properly simulate the smallest wavelength of interest. As a rule of thumb [140], h can be chosen around 5-10 times the smallest wavelength that we need to simulate. This poses obviously an upper limit on the highest audio frequency for which the results are significant. However, due to generally dispersive wave propagation and complex geometries in solid mechanics, it is hard to determine beforehand the value of h given a maximum frequency of interest f_c, so the parameter might be adjusted according to the results if high-frequency simulations are needed.

Thin Shell Elements

Although quite simple, the simple matrix assembly procedure previously shown for elastic planar elements can easily be extended to the treatment of two-dimensional

plates which show bending moments. In both literature [140] and FEM software packages the most common example are the thin-plate (Kirchoff) and thick-plate (Reissner-Mindlin) approximations.

A shell can be thought essentially as a structure that can be derived from a plate by molding the flat surface into a curved one. However, shells support external loads in a way which is radically different from flat plates. It is thus not trivial to directly apply the explicit FEM formulation to a general curved shell structure. Here, one common approximation [140] is of a physical, rather than mathematical, nature. It is assumed that the behavior of a continuously curved surface can be adequately represented by the behavior of a surface built up of small flat elements. In this way, displacements and forces acting on a node can be derived from the formulation used for thin plate elements.

As a first step in the definition of elastic problems for shell elements, both displacements and forces are derived in a *local* coordinate system, which varies from element to element. Referring to Fig. 2.5 and dropping for simplicity of notation the apex e used in Eq. (2.1), the vector of the nodal displacements for a single node i is given as

$$\mathbf{u}_i^e = [u_i \; v_i \; w_i \; \theta_{xi} \; \theta_{yi} \; \theta_{zi}]^T \tag{2.11}$$

and the appropriate nodal 'forces' are combined in the vector

$$\mathbf{f}_i^e = [F_{xi} \; F_{yi} \; F_{zi} \; M_{xi} \; M_{yi} \; M_{zi}]^T . \tag{2.12}$$

It has to be noticed that, starting from the formulation of flat elements, the rotation around the z-axis θ_z does not enter as a parameter into the deformations of the strain. However, it is often inserted into the expression together with a fictitious force moment M_z, as this can be useful for improving the condition number of the stiffness matrix in case of coplanar elements [140]. This technique is often cited as *drill stabilization* in some FEM shell element solvers [65].

With such a choice of element and nodal variables, single blocks of the stiffness matrix $\mathbf{K}$ can be assembled. It is then necessary to transform these single blocks from *local* to *global* coordinates if we want to express equilibrium equations in presence of external loading forces. This transformation is usually done by applying a matrix of *direction cosines* to the nodal displacements and forces for each element. Typically, it is not necessary to explicit any local coordinate information while working on common FEM software packages : both the problem formulation and the solution are expressed only in global coordinates, and the task of converting from and to local coordinates is performed by the solver.

Dynamic Systems

In the previous discussion of the finite element method, a static analysis of the object was considered, which is summarized by the equilibrium equation (2.10). The goal of our analysis is to find the *dynamical* behavior of the object, in order to compute the displacement at every time instant. This can be accomplished by adding a kinetic energy term to the potential Π and by applying Hamilton's principle, which leads to the follow system of Ordinary Differential Equations (ODEs):

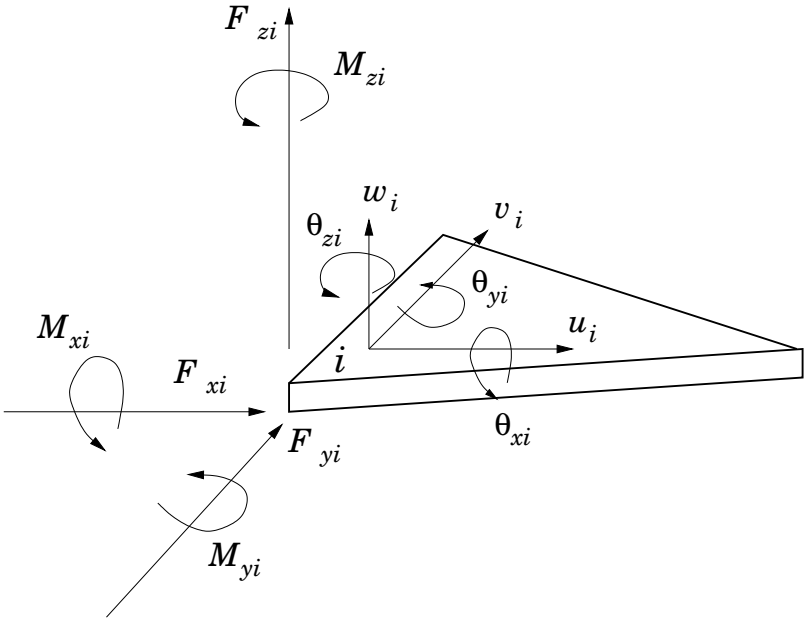

Fig. 2.5. Forces and strains on a flat shell-element node.

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f} \tag{2.13}$$

where we have renamed $\mathbf{u}^g$ in $\mathbf{u}$ for simplicity of notation and we have introduced a *mass matrix* $\mathbf{M}$ and a *damping matrix* $\mathbf{C}$. The mass matrix for a single element is defined again by an integral over the element volume:

$$\mathbf{M}^e = \int_{V^e} \rho \mathbf{N}^T \mathbf{N} dV. \tag{2.14}$$

This system of ODEs completely describes the linear motion of the surface. It is important to notice that both matrices $\mathbf{M}$ and $\mathbf{K}$ are simmetric and positive definite. This is fundamental in the diagonalizing procedure that leads to normal modes and that will be described in the next section. The nature of the damping matrix $\mathbf{C}$ will be discussed with more detail in a further section. For now, it may be convenient to assume that it is a linear combination of the mass and stiffness matrices, although the extraction of normal modes can work even with more complex formulations of $\mathbf{C}$.

2.2 Modal Decomposition

Modal analysis [58] is a standard tool used for the computation of the solution of the linear equation of a multiple DoF system as the one described by Eq. (2.13). It has widely been used in many of the *modal synthesis* approaches proposed for real-time sound simulation [26, 27, 66, 88, 100, 104, 124, 125] and it is sometimes employed in computer graphics for the simulations of deformable bodies [52,67,85].

The main idea behind modal decomposition is that the linear system of Eq. (2.13) can be diagonalized into a set of decoupled ODEs. Each ODE obtained in this way describes the motion of a *normal mode* of the original system and its solution can be computed in a numerically efficient way.

2.2.1 Reduction to Modal Coordinates

Given the mass, damping and stiffness matrices $\mathbf{M}$, $\mathbf{C}$ and $\mathbf{K}$ of Eq. (2.13), there exists a matrix $\mathbf{\Phi}$ and a diagonal matrix $\mathbf{\Omega}$ such that

$$\mathbf{\Phi\Omega} = \mathbf{M}^{-1}\mathbf{K\Phi}.$$

The columns of $\mathbf{\Phi}$ are the generalized eigenvectors of $\mathbf{M}$ and $\mathbf{K}$, and the diagonal elements of $\mathbf{\Omega}$ are the generalized eigenvalues. Because $\mathbf{M}$ and $\mathbf{K}$ are normally symmetric positive definite, they can be simultaneously diagonalized by $\mathbf{\Phi}$:

$$\mathbf{\Phi}^T\mathbf{M\Phi} = \tilde{\mathbf{M}}$$
$$\mathbf{\Phi}^T\mathbf{K\Phi} = \tilde{\mathbf{K}}$$

where $\tilde{\mathbf{M}}$ and $\tilde{\mathbf{K}}$ are diagonal. When $\mathbf{C}$ is a linear combination of $\tilde{\mathbf{M}}$ and $\tilde{\mathbf{K}}$, it can also be diagonalized by $\mathbf{\Phi}$ into the diagonal matrix $\tilde{\mathbf{C}}$. Substituting into Eq. (2.13) and using the transformation to *modal coordinates* $\mathbf{y}$ given by

$$\mathbf{u} = \mathbf{\Phi y} \tag{2.15}$$
$$\mathbf{f} = \mathbf{\Phi g} \tag{2.16}$$

yields to the following system of ODEs:

$$\tilde{\mathbf{M}}\ddot{\mathbf{y}} + \tilde{\mathbf{C}}\dot{\mathbf{y}} + \tilde{\mathbf{K}}\mathbf{y} = \mathbf{g}. \tag{2.17}$$

In the form of Eq. (2.17), the system equations are linearly independent and each equation describes a *normal mode* of the object being modeled. The columns of $\mathbf{\Phi} = [\mathbf{\Phi_1}, \ldots, \mathbf{\Phi_N}]$ are the *mode shapes* and the elements of $\mathbf{\Omega} = [\omega_1, \ldots, \omega_N]$ are the mode frequencies. Obviously, if the original systems was defined by N DoFs (the number of elements of $\mathbf{u}$), the diagonalizing procedure will lead to a system of N normal modes.

The equations (2.16) convert physical variables defined in the original world-system coordinates $\mathbf{u}, \mathbf{f}$ into equivalent *modal variables* defined in the modal coordinate system $\mathbf{y}, \mathbf{g}$. This scalar projection can be thought as a similar operation to the process that converts signals from the time to frequency domain (and viceversa) in the context of discrete functional transforms such as the Discrete Fourier Transform.

The ODE describing the motion of the k-th normal mode $y_k(t)$ can be explicitly written as:

$$\ddot{y}_k + R(\omega_k)\dot{y}_k + \omega_k^2 y_k = \frac{1}{m} g_k(t), \tag{2.18}$$

where $R(\omega_k)$ is a damping term that depends on the form of the damping matrix $\mathbf{C}$, m is the total object scalar mass and $g_k(t)$ is the modal force signal acting on the k-th normal mode.

This is the well-known equation of the damped harmonic oscillator, with an external forcing term. An analytic expression for the free evolution of the system, also referred as the *steady-state* solution, is obtained by using the Dirac impulse function δt as the forcing term. The actual solution depends on the particular form of the damping term $R(\omega_k)$. We consider here only the case of *underdamped* vibration [84], where the condition $R(\omega_k)^2/4 < \omega_k^2$ holds. Other cases of damping, such as *critically damping* ($R(\omega_k)^2/4 = \omega_k^2$) or *overdamping* ($R(\omega_k)^2/4 > \omega_k^2$) are not considered because they lead to non-oscillatory behaviour which has very few applications for the simulation of acoustic properties of the object.

In the underdamped case, the free solution of the motion of the k-th normal mode can be written in terms of an exponentially decaying sinusoidal oscillator:

$$\begin{aligned} y_{\delta,k}(t) &= A_k e^{-t/\tau_k} \sin(\tilde{\omega_k} t) \\ A_k &= \frac{1}{m\tilde{\omega_k}} \\ \tau_k &= \frac{1}{R(\omega_k)} \\ \tilde{\omega_k} &= \sqrt{\omega_k^2 - R(\omega_k)^2} \simeq \omega_k \end{aligned} \tag{2.19}$$

where A_k is the oscillator initial amplitude, τ_k the oscillator time-constant (often referred to also as *decay time*) and $\tilde{\omega_k}$ is the oscillation frequency which, for typical values of the damping term, can be approximated very well by the modal frequency ω_k.

Modal Weights

Usually, we do not need to know every element of the displacement vector $\mathbf{u}$. For example, we might only need to know the displacement at a single node u_n, with $1 \leq n \leq N$. In this case, the projection described by Eq. (2.16) is simplified into a scalar product with the vector $\mathbf{\Phi}_n$, i.e. with the n-th column of the modal shape matrix $\mathbf{\Phi}$. In the following, we will refer to any vector which can be used to derive a physical variable from a scalar product of the modal coordinates as an *output weights* vector $\mathbf{w}_{\text{out}} = [w_{\text{out},1}, \ldots, w_{\text{out,N}}]$. More complex output weights vector can be defined to get the average displacement on a surface area defined by one or more elements, or the near-field sound pressure generated by the vibration of the object.

In a similar way, we will refer to a vector which can be used to convert a continuous-time force signal $F(t)$, having a time-constant spatial distribution on the object, as an *input weights* vector $\mathbf{w}_{\text{in}} = [w_{\text{in},1}, \ldots, w_{\text{in,N}}]$. Here, the assumption on the force can be quite strong in many contexts. For example, it is not possible to express with such a formalism a situation where the spatial distribution of the force is varying over the time, as it happens during the contact between two colliding objects. However, as it will remarked in the following chapter, many simplified models of impact forces can be implemented which do not require time-varying contact surfaces.

If we take into account the contribute of all the normal modes, we can express the following Input/Output relation between a force signal F_n acting on the n-th

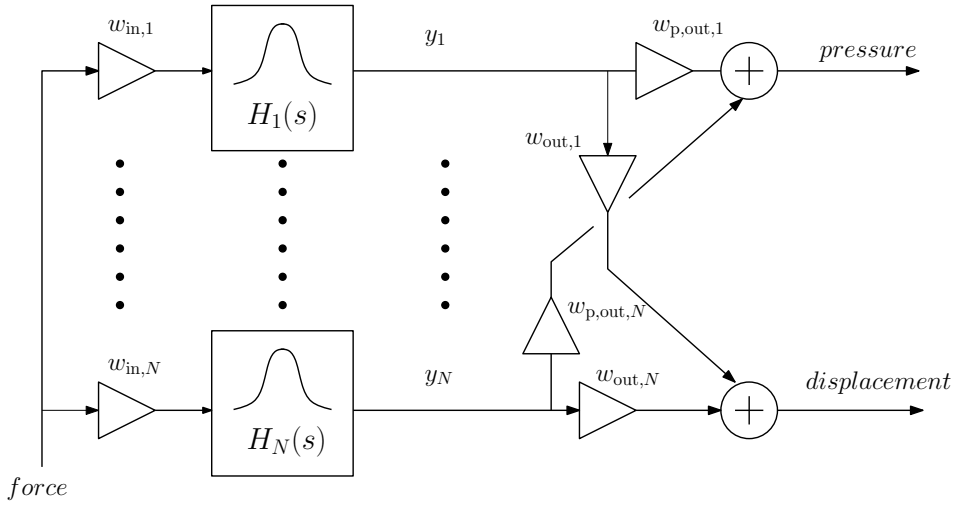

Fig. 2.6. Block diagram representing the interaction between a localized force signal and the output displacement and pressure signals. The scalar products with the input and output weights act as a change of coordinates between various the physical coordinates and the continuous-time normal modes $H_n(s)$.

node and the displacement u_m of the m-the node:

$$\begin{aligned} u_m(t) &= \sum_{k=1}^{\tilde{N}} w_{\text{out,k}} y_k(t) \\ y_k(t) &= y_{\delta,k} * w_{\text{in,k}} F_n(t) \end{aligned} \tag{2.20}$$

This equation can be interpreted in a signal-processing context as a parallel bank of continuous-time resonators, leading to the block-diagram of Fig. 2.6 where the resonator blocks are written in term of their Laplace transforms. The formalism used for input/output weights as transformers between the physical and modal variables is very similar to already proposed methods in the sound synthesis literature, e.g. for musical sound synthesis [9, 14] or modal synthesis derived from mass-spring networks [104].

So far, the modeling equations have been presented in the continuous-time domain. In order to get a concrete numerical realization, Eq. (2.20) must be discretized in the time variable. The common way to do so is by using a second-order digital resonator, which is a cheap computational structure which can approximate very well the behaviour of the continuous-time resonator defined by Eq.(2.18). The details of the most common discretization techniques, along with some properties of these numerical resonators, are discussed more extensively in the next chapter.

Modal weights can be easily derived from the modal shapes matrix Φ if we are interested in knowing the exact displacement of a single node for a single direction in the Cartesian space. This means that, in order to get just the modal displacement of a single node, we need to employ a set of three different output weights. If, as it is usual, we need to know the displacement along a contact surface, the number of weights can rapidly increase and both the computational

and memory costs can largely overcome those of the discrete-time integration. This is one of the reasons why modal synthesis is not as popular for graphics simulations, where it is necessary to update each node coordinate at each time step. In audio, however, we are usually interested in a small subset of the nodes displacements, typically those around the contact point with another colliding object. Moreover, with some simplified contact models e.g. the one that we will be presented in Chapter 4, only the displacement along the normal to the surface is needed.

With the finite element formulation, it is possible to easily compute the average normal displacement within one or more elements by exploiting the relationship between nodal values and displacement given by Eq. (2.1). The average normal displacement over a triangular element e can be computed as:

$$\bar{\hat{\mathbf{u}}} = \frac{1}{A^e} \int_{A^e} \sum_{k=0}^{2} \mathbf{N}_k \mathbf{u}_k \cdot dA = \frac{1}{3} \sum_{k=0}^{2} \mathbf{u}_k \cdot \hat{\mathbf{n}^e}, \tag{2.21}$$

that is we just need to compute a scalar product between the nodal displacement vectors $\mathbf{u}_k$ and the normal to the triangular element $\hat{\mathbf{n}^e}$, since the integral of the basis functions over the element surface is equal to $A^e/3$ in the linear element case. If we substitute in the previous equation the effect of the diagonalizing matrix $\mathbf{\Phi}$, we can express the m-th output weight corresponding to the average modal displacement of the k-th element:

$$w_{\mathrm{m,k}} = \frac{1}{3} \sum_{i=0}^{2} \Phi m, k, i \cdot \hat{\mathbf{n}^e}, \tag{2.22}$$

where $\Phi m, k, i$ is the value of the modal shape matrix $\mathbf{\Phi}$ for the k-th element and m-th node along the i-th Cartesian direction. Due to the linearity of Eq. (2.22), it is easy to compute weights for the average normal displacement over a set of contiguous elements, by a simple average of the value in each element weighted by the triangle's area.

Input weights and normal-averaged input weights are computed in the same manner as output weights. The only difference, coming from Eq. (2.18), resides in a scaling with the total object mass. This means that we do not need to separately precompute and store in memory an input weights vector for real-time simulations.

2.3 Damping Effects

From a theoretical point of view, the damping matrix $\mathbf{C}$ can be calculated by assembling contributions from element damping parameters, using for example results from linear viscoelastic theory without memory [89]. Useful results can be obtained with other modal-based approaches where the geometry of the problem is very simple. For example, in the case of flexible unidimensional strings, viscoelastic theory easily leads to a compact formulation of the damping matrix [120].

However, for complex shapes this process is quite complicated also because, in general, the theory of damping is not as well known as the theory of elastic behavior. Moreover, damping matrices obtained in this way are usually not symmetric and positive definite, and so modal analysis of the system requires the use

of complex-valued mode shapes. It is therefore common in practical engineering the use of Rayleigh's damping assumption or *proportional damping* [58], which express $\mathbf{C}$ as a linear combination of $\mathbf{M}$ and $\mathbf{K}$:

$$\mathbf{C} = \alpha\mathbf{M} + \beta\mathbf{K}. \tag{2.23}$$

The definition of the damping parameters has a fundamental role in defining the perceptual qualities of the produced sound [71, 104]. Due to the nature of the Rayleigh damping approximation, the parameters α, β have no strict relation to the physical properties of the object considered. It is thus common to manually adjust these parameters in order to perceptually control the decay times distribution of the object. With the Rayleigh damping approximation, the damping term $R(\omega_k)$ in Eq. (2.18) assumes the linear form $R(\omega_k) = \alpha + \beta\omega_k$. Thus, the decay times follow a hyperbolic distribution over the frequency:

$$\tau_k = \frac{1}{\alpha + \beta\omega_k} \tag{2.24}$$

It might be better to control the damping properties via a more meaningful set of parameters from the sound design point of view. As it has been suggested in [9], one practical solution is to give the user the controls over the following parameters:

1. The decay time T_0 at DC frequency, expressed as T_{60}, i.e. the time necessary to decrease the amplitude by 60dB [110]
2. The frequency $f_{T/2}$ (in Hertz) at which the decay time is half of T_0.

With this choice, it is trivial to compute the proportional damping parameters from the user controls:

$$\begin{aligned} \alpha &= \frac{1}{\log(1000)T_0} \\ \beta &= \alpha\frac{1}{2\pi f_{T/2}} \end{aligned} \tag{2.25}$$

It has to be noted that, with the proportional damping approximation, damping parameters do not modify the modal shapes matrix $\mathbf{\Phi}$ computed by the diagonalizing procedure. In fact, the diagonalizability of the damping matrix $\mathbf{C}$ by the modal matrix $\mathbf{\Phi}$ does not depend on the particular damping parameters. In practice, this means that we can compute the modal shapes for the elastic problem in absence of damping, and then impose a decay-time distribution such as Eq.(2.24)

If there is a need to model a more complex relationship between mode frequencies and decay times, a generalization of the proportional damping model can be adopted. Caughey and Kelly [28] proposed a series representation of the damping matrix $\mathbf{C}$:

$$\mathbf{C} = \mathbf{M}\sum_n \alpha_n \left[\mathbf{M}^{-1}\mathbf{K}\right]^n \tag{2.26}$$

and they showed that such a representation is the necessary and sufficient condition for existence of classical normal modes for systems without any repeated roots. In terms of practical implementation, generalized proportional damping leads to a polynomial distribution of decay rates:

$$R(\omega_k) = \sum_{n=0}^{N_d} \alpha_n \omega_k^n. \tag{2.27}$$

As it will shown in the next paragraph, even a second-order polynomial ($N_d = 2$) can give a better approximation than proportional damping for matching some measured decay distributions. However, increasing further the order of the polynomial has a serious drawback in the control of the curve, because we need to specify more parameters and the oscillating behaviour of higher-order polynomial is often undesirable.

Another generalization of proportional damping in terms of analytical functions is given by Adhikari [1, 2]. Even in this case the diagonalizability of the damping matrix is assured, while having more flexibility in modeling the damping behaviour over frequency. Implementing such an approach to derive more accurate decay curves is a topic of future research.

2.3.1 Estimation of Damping Parameters

In the previous paragraph, we have used a simple damping model whose parameters needed to be manually adjusted by the user in order to achieve the desired decay properties. It is also possible to estimate such parameters from a recorded sound of a given object, for example if we want to match the decay properties of a certain material.

The topic of modal decay estimation for audio systems is widely covered by the literature. For a detailed review of the most common techniques, in the context of audio reverberation and musical sound synthesis, see [69]. Here, we will present a procedure which has been used for the estimation of piano strings decay parameters [134, 136]. Nevertheless, the generality of the damping model that has been assumed makes possible to easily use such techniques for a general class of modal objects.

We assume a target sound represented as a mono discrete-time signal $x(n)$, with sampling frequency f_s. The goal of the procedure is to estimate the decay parameters α_n in Eq. (2.27), where we also consider the proportional damping case when $N_d = 1$. The algorithm can be summarized in two main computational steps. In the first part, salient modal frequencies are estimated on the signal and their decay times are computed. Then, once we have this discrete map from frequencies to decay times, we can fit the chosen decay law using standard regression techniques.

In the detail, the algorithm can be decomposed in the following steps:

1. A set of relevant modal frequencies ω_k, $1 \leq k \leq N_{\text{modes}}$ is estimated with some traditional peak-picking algorithm on the Short-Time Fourier Transform (STFT) of the signal (see for example [141], chap. 10 for a reference technique used in additive synthesis). Since the purpose here is not to extract the whole modal data, but just the overall damping behaviour, the peak-picking procedure need not to be very accurate. Here we just selected the highest-magnitude spectral peaks in a STFT frame centered shortly after the start of the signal, with a preprocessing median-filtering on the spectral envelope to reduce noise and false peaks [132].

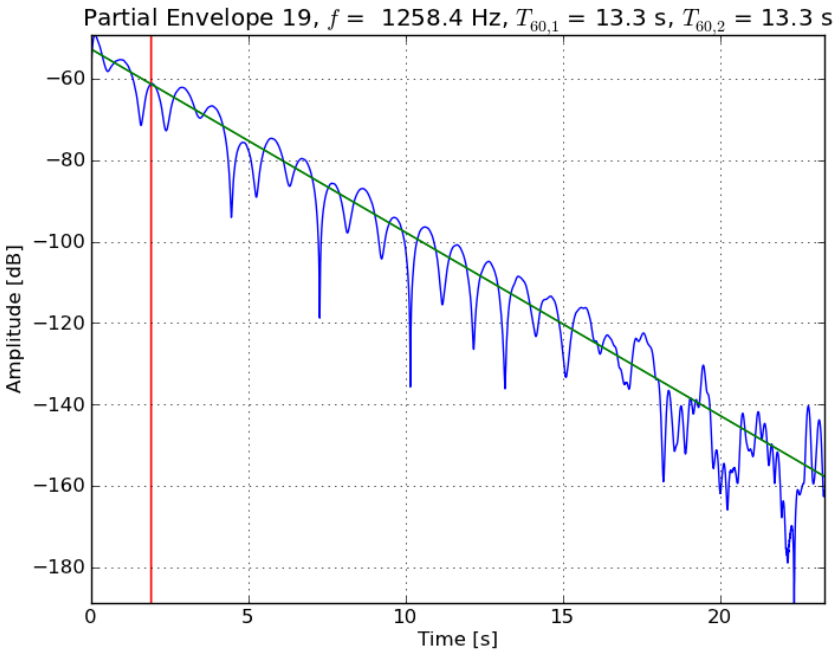

Fig. 2.7. Example of modal envelope extraction from a recorded sound. The original signal had a lenght of 30 sec., which has been appropriately cut by the noise level estimation procedure. The fitted straight-line envelope is shown on the top of the log-envelope of the heterodyne modulated signal.

2. For each modal frequency ω_k, the modal amplitude envelope is computed by means of *heterodyne* filtering. A similar procedure is sometimes referred in literature as frequency zooming [70], and it employs the following steps:
   - Shift the frequency of interest around DC, by modulation with a complex-valued oscillator:
   $$\tilde{x}(n) = x(n)\, e^{-j\frac{2\pi\omega_k n}{f_s}}$$
   - The complex signal $\tilde{x}(n)$ can then be lowpass filtered and downsampled according to the desired analysis bandwidth. The ratio of downsampling (referred as *zooming factor* in [70]) is usually between 50 and 100. The log-magnitude of the resulting signal is referred to as the *modal envelope* (or partial envelope in musical contexts) and can be seen in Fig. 2.7. Note that the envelope of a single sinusoidal mode should be a straight line in this logarithmic scale. However, especially for objects with complex geometry and high stiffness, it is common to observe in a single frequency window the combined effect of multiple modes, which give rise to beatings in the amplitude envelope.
   - The last part of the signal is cut with a noise floor level estimation similar to the one proposed in [44]. This is necessary because, otherwise, the estimation would be very poor especially for high-frequency modes which

decay shortly after the attack. First, the Root Mean Square (RMS) of the signal is taken in small (5-10 ms) windows and a straight-line dB envelope is fitted on the last part of this data. Then, a rough least-square estimate of a linear envelope is taken on the attack part of the signal, and the effective data length is computed from the intersection of these two lines.

3. A linear envelope is fitted on the signal obtained at the previous step. The author has found that, in presence of beatings and/or noisy data, using a *robust* regression estimate such as Least Absolute Deviation Regression (LADR) [74] gives better results than traditional Least Squares techniques. The decay time τ_k is directly computed from the first-order coefficient of the fitted line.
4. Once we have a set of modal frequencies ω_k and their relative decay times τ_k, a polynomial of desired order N_d is fitted over the decay rates $r_k = 1/\tau_k$ distribution. The fitting procedure can be compromised by the presence of several outliers in the decay times estimate. Nevertheless, the employment of a robust fitting procedure such as LADR has been found to usually perform better than least-squares fitting in most cases. Fig. 2.8 compares the fitting result of a first-order polynomial obtained with least-squares regression against a second-order LADR fit. It can be clearly seen that, especially in the low-frequency range, the second order model follows better the decay-times distribution.

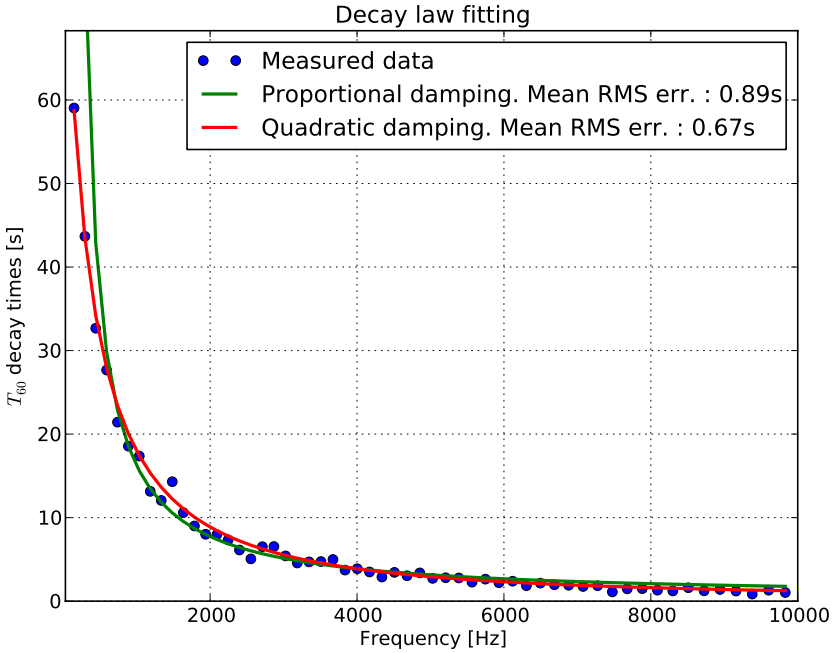

Fig. 2.8. Fitting of a polynomial damping law over a set of estimated modal frequencies and decay times (blue dots). The green line shows a linear fit obtained with least-squares regression, while the red one represents a second-order fit by LADR.

The proposed method for estimating the damping parameters is quite simple to implement and, if the goal is to match the overall damping behaviour, produces good result. Other methods proposed in literature differ mainly for the way in which the decay time of the single partial is estimated. In FZ-ARMA modeling [70] standard system identification techniques (such as Prony's method or Stieglitz-Mc Bride iterations [94]) can be used to estimate more than one mode on each single heterodyne envelope. Recently, a modal estimation procedure for impact sounds based on Gabor transforms has shown to produce very accurate results [107]. The technique can also be used for joint frequencies-decays estimation, in applications where all the synthesis parameters are derived from a prerecorded sound.

2.4 Radiation Modeling

In a virtual reality application sounding objects are immersed in a *listening environment*, which is also described by the listener's position in the scene. Thus, we need a method to compute how the vibrations on the surface of the objects are radiated towards the listener according to the law of acoustic pressure propagation in the air [84]. The phenomenon is hard to model properly and is taken into account only in few works coming from computer graphics researchers [29, 66, 87], while it is almost neglected in all the other publications on interactive sounding objects. It is however indirectly considered for the synthesis of some musical instruments, especially for the piano [14], since in this case the instrument body is usually simulated by measuring its force/pressure impulse response with an experimental setup.

Radiation from a particular vibration mode interacts with the object's geometry in a complicated way to affect its *radiation efficiency* [31, 84], boosting radiation from some frequencies and suppressing others. This should be taken into account into sound modeling, since audible sounds have wavelengths that are comparable to the size of objects in human environments. In the following, we will summarize the radiation problem for a surface vibrating with a single harmonic frequency and we will give an expression for a simple near-field approximation. Notice that we neglect the effects of *reverberation* [141], as this will be complicate too much the problem formulation. Eventually, reverberation effects can be added in a post-processing stage. In addition, the radiation from each object is investigated separately, i.e. the interaction between different radiating sources is neglected.

2.4.1 Helmholtz Equation

With acceptable approximations [84], in an homogeneous medium at rest the sound pressure P satisfies the wave equation

$$\nabla^2 P - \frac{1}{c^2}\frac{\partial^2 P}{\partial t^2} = 0, \tag{2.28}$$

defined over the domain $D = \mathbb{R}^3 \setminus O$, where $O \subset \mathrm{R}^3$ is the volume enclosed by the object. The physical quantity c is the speed of sound and ∇^2 is the Laplacian

operator. We denote with $\mathbf{x}$ a general "listening" position in the domain, whose origin is taken arbitrarily inside the object. Let us focus on a single vibrational mode having frequency ω. We can then consider the case of time harmonic acoustic propagation, that correspond to assume a pressure field of the form

$$P(\mathrm{x}, t) = \Re\left(p(\mathbf{x})e^{-j\omega t}\right) \tag{2.29}$$

Substituting (2.29) in (2.28), we can see that p satisfies the Helmholtz equation

$$\nabla^2 p(\mathbf{x}) + k^2 p(\mathbf{x}) = 0, \qquad \mathbf{x} \in D \tag{2.30}$$

where $k = \omega/c$ is the *wave number*. Boundary conditions are required to solve Eq. (2.30) for sound radiation from surface vibrations. First, the normal derivative of pressure on the vibrating object's surface ∂D is given by an impedance boundary condition

$$\frac{\partial p}{\partial n} = -j\omega\rho\nu \quad \text{on } \partial D \tag{2.31}$$

where $\partial/\partial n$ denotes the normal derivative on the boundary, taken in the direction outgoing from the surface, ρ is the fluid density and ν is the surface's normal velocity, that can be computed from the mode shape. The second boundary condition is given by the *Sommerfeld radiation conditions*, which express the idea that the acoustic field is traveling outwards towards infinity:

$$p(\mathbf{x}) = O\left(r^{-1}\right) \tag{2.32}$$

$$\frac{\partial p}{\partial r}(\mathbf{x}) - jkp(\mathbf{x}) = o(r^{-1}) \tag{2.33}$$

as $r \to \infty$, where $r = |\mathbf{x}|$ is the distance of $\mathbf{x}$ from the origin.

The solution of each mode's radiation problem can be done using several numerical methods. Among the different choices, which include for example finite differences or finite element methods [64], the boundary element method (BEM) is widely used in the acoustic community [31]. It is based on an integral formulation of the problem via Green's theorem. The most noticeable advantage of the method is that the solution at any point in the domain D can be expressed as a linear combinations of elements defined only on the boundary ∂D, thus significantly reducing the size of the computational domain.

2.4.2 Accurate Radiation Modeling

We will briefly discuss here one of the advanced algorithms proposed for real-time radiation modeling [66], based on a multipole expansion of the precomputed transfer function from BEM simulations. In this work this technique has not been investigated thoroughly, mainly because contrarily to the Finite Element Method, there are much fewer Boundary Element Method solvers available and almost none which is both open source and targeted at acoustic problems. For instance, the authors of the above mentioned work have used a commercial BEM solver which cost is somewhere in the ten-thousand dollars range.

Once the pressure field has been computed for each mode, it is possible to use this information to get the proper modal output weights. The problem here is

that, even if the solution of Helmholtz equation is done offline, it is not practical to store it for a later use at run-time, due to the prohibitive memory requirements even if we limit the size of the acoustic domain. If BEM is used, the pressure field $p(\mathbf{x})$ at non-boundary locations $\mathbf{x}$ can be computed by a linear contribution of M boundary elements. Unfortunately, this leads to a computational cost of $O(NM)$ for N modes, which is still too demanding for real-time use.

Thus, James et al. [66] proposed an algorithm for the fast evaluation of the pressure field at run-time. As a first offline stage, they solve Helmholtz's equation (2.30) for each mode using BEM. Then, the pressure field in the whole domain is approximated by a *multipole expansion*, i.e. by a linear combination of M spherical multipoles [84]. The order of the multipole is a free parameter in the approximation procedure and, in their examples, they suggest the use of *dipole sources*. Fig. 2.9 shows the results in approximating the radiating field of a single mode with few dipoles. Multipoles positions are selected using a greedy placement algorithm, which takes into account the boundary conditions of the original problem. Finally, the weighted contribution of each multipole is calculated by solving a least squares problem through truncated singular value decomposition.

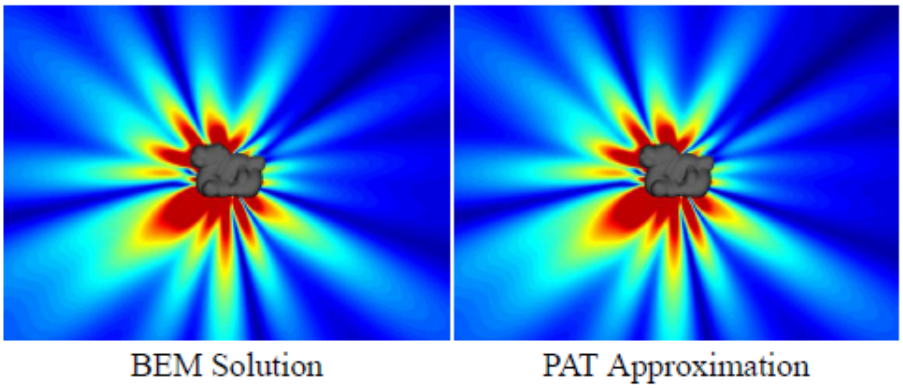

Fig. 2.9. Radiation of a mid-frequency mode for a complex geometry object as computed from a BEM simulation and from a multipole approximation with 63 dipoles. Figure taken from [66]

Once the multipole sources are placed and their amplitude is known, at run-time it is possible to sum these contributions according to the relative position of the listener to the object. This process is referred as Precomputed Acoustic Transfer (PAT) evaluation. As they are not considering phase-related effects, this evaluation is performed at *control rate* (30-100 Hz), so that the computational amount for this evaluation is significantly cut down. Nevertheless, it is an open issue how to efficiently include phase information from the multipoles, which can be used for example for Doppler's effect simulation.

In the discussion of the results, they note how, in general, a complex-shape object requires a number of multipoles that increases with the frequency ω of the mode. This is quite intuitive, as many direction-depending acoustic phenomena are

usually relevant only in the high-frequency range [84]. As a direct consequence, high-frequency modes can have a much more expensive PAT evaluation than low-frequency modes. This can easily lead to high computational costs in the presence of many objects with several high-frequency modes.

Further research in radiation has been presented by the same authors in a work focused mainly on nonlinear shell vibration by modal synthesis [29], where the large number of dipoles required for high-frequency modes is significantly reduced with a technique called *Far-Field Acoustic Transfer Maps* (FFAT). The key idea is to use a far-field approximation in terms of a combinations of spherical functions instead of dipoles.

A different approach for accurate radiation modeling has been followed by O'Brien et al. [87], where nonlinear finite elements are used to discretize not only the vibrational behaviour of the object surface, but also the surrounding air space. Even nowadays, the method is extremely expensive due to the large number of elements required and thus not suitable for real-time implementations.

2.4.3 Near-Field Approximations

As we have seen, accurate radiation modeling is usually a hard task, due to the complex preprocessing techniques and the significantly increase in computational cost. Even the most efficient of the algorithms cited in the previous paragraph can require up to 50 output weights per mode, whose computational cost is roughly more than one order of magnitude of the cost for the time-domain realization of a single resonator.

A simpler yet less effective approach is to model only the near-field radiation for each mode, as it has been suggested in [88]. The pressure field near the surface of a triangular element e is given by:

$$p^e = \frac{Z}{A^e} \int_{A^e} \mathbf{v}^e \cdot \hat{\mathbf{n}}^e, \tag{2.34}$$

where Z is the material-air impedance and $\mathbf{v}^e$ is the displacement velocity on the element, i.e. the first time derivative of the displacement $\mathbf{u}^e$. The procedure for computing the *pressure output weights* is then similar to the one we used previously to compute the average normal displacement in Eq. (2.21). The time-derivative of the continuous-time impulse response of a single resonator Eq. (2.19) is:

$$\dot{y}_{\delta,k}(t) = A_k e^{-t/\tau_k} \left(\frac{1}{\tau_k} \sin(\tilde{\omega_k} t) + \tilde{\omega_k} \cos(\tilde{\omega_k} t) \right), \tag{2.35}$$

thus we can express the (complex-valued) pressure output weights for the k-th mode as:

$$w_{p,\text{out},k} = \omega_k \int_A \mathbf{\Phi_k} \cdot \hat{\mathbf{n}} \left(-\frac{1}{\tau_k} + j(\tilde{\omega_k} + \frac{\pi}{2}) \right), \tag{2.36}$$

i.e. an integral over the whole surface A of the scalar product between the k-th column of the modal shape matrix $\mathbf{\Phi}$ and the normal to the surface, plus a phase offset of $\pi/2$ due to the cosine in the derivative of the impulse response. Chapter 2 shows how it is easily possible to incorporate phase shifts in the common

implementations of a digital resonator. The integral of Eq. (2.36) can be performed using the expression we have already derived for the average normal displacement over a set of elements in Eq. (2.22). The summation has to be carried among all the elements, but it can quickly done offline without hurting real-time performance.

2.5 Modal Shapes Reduction

Among all the data vectors that we need to store for the real-time simulation of an object, the output weights are by far the most demanding structure. For instance, the accurate simulation up to 15 KhZ of a bell-shaped steel object having a radius of around 20 cm results in a triangular mesh of approximately 9000 elements for 250 modes. If full three-dimensional displacement weights are computed, we need to store $250 \times 9000 \times 3 = 6,750,000$ floating point values (around 27 MegaBytes). This is clearly a huge number for the sound synthesis of a small object, especially on modern hardware where memory access costs are usually much bigger than those for floating point operations.

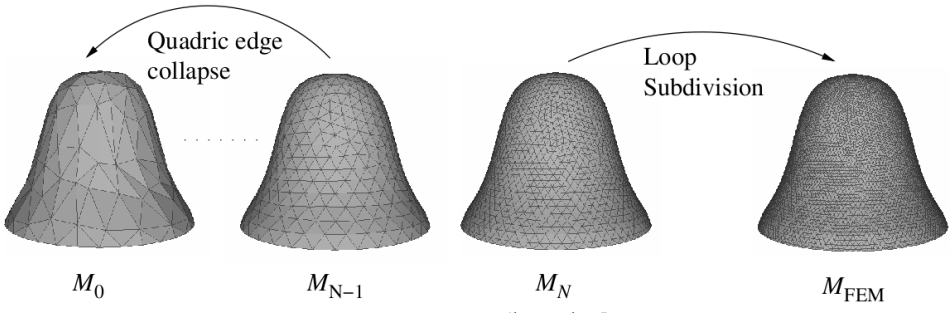

Fig. 2.10. Multi-resolution mesh for storing modal shapes can be obtained by mesh refinement (via Loop Subdivision Surfaces) or coarsening (e.g. with the Quadric Edge Collapse algorithm).

The problem of modal shapes reduction has already been addressed in the computer graphics community, e.g. with the use of compression techniques based on Wavelet transforms [68]. However, for audio simulation we can further simplify the question by a drastic reduction of mesh complexity used for real-time simulations. It is necessary to separate the mesh resolution requirements for the offline and real-time steps of the simulation. During offline FEM computation of the modal shapes, a high resolution mesh is required to get accurate results for high-frequency modes. For the real-time simulation phase, however, large meshes are intractable and, moreover, from a perceptual point of view it is not needed to model precisely the amplitude distribution of the high-frequency partials [104].

Even in most rigid body simulators used in virtual reality environments, it is a common practice to store the physical simulation mesh with a lower resolution compared to the one used for graphics rendering [20]. We applied the following procedure to handle different resolution meshes for modes extraction and output weights storage:

1. The starting mesh is usually the one used for graphics rendering. Typically, the resolution of this model is not high enough for accurate FEM simulations and often too big for real-time simulations.
2. The starting mesh is refined into a finer one by using the Loop Subdivision Surfaces algorithm [77]. The algorithm is widely known in computational geometry and it is based on an iterative mesh refinement via binary subdivisions of triangular splines.
3. Eventually, a coarser mesh than the original one can be obtained by a mesh reduction algorithm, such as the Quadric Edge Collapse method [78]. The drawback of these methods is that it is hard to ensure that each triangle in the coarse mesh can be composed exactly by a number of triangles of the finer mesh. On the other side, there are robust implementations available in common meshing softwares [30].
4. The fine mesh is used for FEM elasticity computation and modal shapes extraction. The results are stored in an intermediate file and the mesh nodes are inserted in a OctTree structure [38] for fast position-based queries.
5. Pressure output weights are computed directly from the fine mesh, since we need to store just one weight for each mode.
6. For each triangular element in the coarse mesh, we need to compute the average modal displacement. This is done by finding the fine mesh elements that compose the larger element in the low-resolution mesh. First, we perform an OctTree query on the previously stored point tree using a search radius of twice the circumcenter of the coarse triangle element. Then, we perform a point-in-triangle test [38] to keep only the fine triangles whose vertices completely fall inside the coarse triangle. Finally, we compute the average normal displacement on this set of elements using Eq. (2.22)

With the proper use of available meshing software and programming library, the procedure is simple to implement. Details on the tools used are discussed in Appendix A. Referring to the above-mentioned example of the bell-shaped steel object, it is possible to reduce the memory requirements as low as 50,000 floating point values for a typical run-time mesh of 200 triangles, i.e. we can achieve good perceptual approximations with less than 1% of the memory required for the full mesh.

2.6 Conclusion

In this chapter, the theoretical foundations of Finite Element Modeling for vibroelastic simulations were reviewed, along with some operations that significantly reduce the cost of models in a real-time context. Although it might look complicated, the procedure of obtaining the real-time data structures starting from a geometrical mesh and material properties is quite simple with the proper software aids, which are discussed in Appendix A. Summarizing the content of the previous paragraphs, the steps of the procedure are:

1. Refine the starting mesh in order to fit the requirements for FEM simulations

2. Define the necessary FEM parameters, such as material properties (density, Poisson's ratio, Young's modulus) and boundary conditions
3. Use of any standard FEM software with support to triangular shell elements to assemble the system matrices $\mathbf{M}$ and $\mathbf{K}$
4. Obtain the modal shapes matrices and modal frequencies via diagonalization of the system matrix, possibly with the use of a eigenvalue linear algebra package (e.g. ARPACK [72])
5. Precompute the pressure output weights vector, then eventually precompute the average normal displacement weights on a reduced mesh
6. Define modal decay times either by manual adjustment of significant parameters or by analysis of a target sound. These parameters do not depend on FEM discretization and can be modified at any time.

At this point, all the information that is needed for the simulation of a 3D resonator is available, i.e. frequencies, decay times and modal weights. In the next chapter we will see how we numerical equivalents of the continuous-time resonator can be derived for the integration in the time domain.

3

Second-Order Digital Resonators

The fundamental building block of every modal synthesis implementation is the discrete-time digital resonator, which numerically implements the second-order ODE of the damped harmonic oscillator acting under an external force Eq. (2.18).

In the following, the numerical details necessary to the simulation of this system are discussed. One of the most common implementations, the *all-pole* second-order digital resonator, is chosen as the best compromise in terms of accuracy and efficiency for the realization of resonating objects. The other method considered is the so-called *quadrature-phase* resonator (often improperly cited as state-variable resonator), which offers a greater flexibility for the run-time control of the parameters but comes at almost twice the computational cost of the simple all-pole resonator. Floating-point realizations will be assumed, and thus all the details regarding round-off and accumulation errors in fixed-point implementations will be neglected.

A simple and efficient way to convert filter state variables between all-pole and quadrature-phase implementations is presented. Such a transformation can be useful whenever derived variables like velocity and energy of oscillation are needed while using all-pole filters.

3.1 Impulse-invariance Discretization of the Continuous-time Resonator

The discretization with respect to time of Eq. (2.18) can be done by various methods, the most popular choices being the bilinear and impulse-invariance transform [91]. Both techniques lead to stable filters whenever the continuous-time system is also stable, which is always the case for the damping behaviour considered. However, resonators obtained by bilinear transform, as they are e.g. formulated in [104], suffer from the frequency-warping phenomenon, i.e. there is a nonlinear warping between the continuous and discrete time frequency axes, which has to be corrected by pre-warping the resonating frequency [110]. Moreover, digital filters obtained in this way have a direct path between the input and output of the system. This means that, whenever the resonator is connected in feedback with other systems, methods to compute *delay-free* loops have to be used [24, 46].

Here, the impulse-invariance transform is used. Basically, it involves sampling the continuous-time impulse response with a time period of T seconds, corresponding to a sampling frequency of $f_\mathrm{S} = 1/T$ Hz. Since the continuous-time system defined by Eq. (2.18) is band-limited within a very good approximation in the underdamped case, the aliasing coming from this transform is usually negligible. The only obvious precaution we need to take is to avoid implementing any mode which resonant frequency is near or above the Nyquist frequency $f_\mathrm{S}/2$. Moreover, the discrete-time impulse response of a resonator will have a leading zero, thus avoiding most delay-free loops whenever the system is connected e.g. with an exciter block.

Sampling the continuous-time impulse response Eq.(2.19) yields to:

$$y_\delta[n] \;=\; y_\delta(nT) \;=\; TA_0 e^{-\frac{T}{\tau}} \sin(2\pi f_\mathrm{r} T n), \tag{3.1}$$

where we have dropped for the sake of clarity the subscript k since we will refer to a single resonator for the following of the chapter. As for the continuous system, the parameters are the resonator amplitude A, the time constant τ and the resonant frequency f_r. Taking the Z-transform of Eq. (3.1) gives a second-order discrete-time system which has complex-conjugated poles[1]:

$$p_\mathrm{r} = R\, e^{j\omega_\mathrm{r}}, \qquad p_\mathrm{r}^* = R\, e^{-j\omega_\mathrm{r}} \tag{3.2}$$

$$R = e^{-\frac{T}{\tau}} \tag{3.3}$$

$$\omega_\mathrm{r} = 2\pi f_\mathrm{r} T. \tag{3.4}$$

The resulting system has the following transfer function:

$$\begin{aligned} H(z) &= \frac{b_1 z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-2}} \\ b_1 &= A_0\; R\; \sin(\omega_\mathrm{r}) \\ a_1 &= -2\; R\; \cos(\omega_\mathrm{r}) \\ a_2 &= R^2. \end{aligned} \tag{3.5}$$

The corresponding difference equation used for the state update is:

$$y[n] = b_1 u[n-1] \;-\; a_1 y[n-1] \;-\; a_2 y[n-2], \tag{3.6}$$

i.e. we are using a standard Direct Form I realization [110]. Note that the input signal $u[n]$ does not have any direct relationship with the continuous displacement variable u used in Chapter 2.

For the numerical integration of whole resonators such as those described in Chapter 2, it is sufficient to substitute the structure of the digital resonator in place of the continuous-time blocks of the diagram in Fig. 2.6 and optionally use two set of output weights if both pressure and displacement variables need to be computed. The resulting DSP algorithm is a weighted bank of N parallel second-order resonators and is depicted in the block diagram of Fig. 3.1. It is of course possible to incorporate the input weights and one of the two sets of output weights into the numerator coefficients $b_{1,k}$. Usually, the output weights vector which is used most often (e.g. the one for sound pressure) is chosen for this operation.

[1] For the algebraic details, see for example [9], Chapter 2.3 or [92], Appendix A

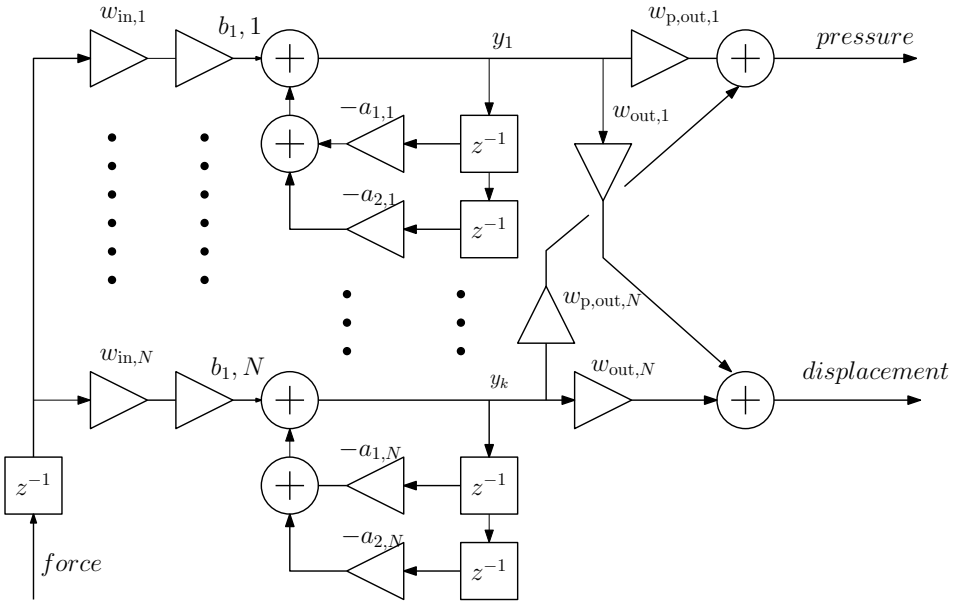

Fig. 3.1. Full block diagram for the discrete-time realization of a modal resonator.

Initial Phase Offsets

The resonator in Eq. (3.5) has a zero-phase offset, i.e. its amplitude is null at time instant $n = 0$. Physically, this corresponds to a system which has null initial conditions, being at rest at the beginning of the simulation. It might be useful for some applications (especially radiation, see Chapter 2) to consider resonators which have an initial phase offset φ_0 at time instant $n = 0$. Considering again the Z-transform of the proper discrete impulse response, it is easy to derive the transfer function of the system for this situation:

$$\begin{aligned} H(z) &= \frac{b_0 z + b_1 z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-2}} \\ b_0 &= A_0 \ \sin(\varphi_0) \\ b_1 &= A_0 \ R \ \sin(\omega_\text{r} - \varphi_0) \\ a_1 &= -2 \ R \ \cos(\omega_\text{r}) \\ a_2 &= R^2. \end{aligned} \tag{3.7}$$

Note that in this case there is a delay-free path between the input and the output, since the coefficient b_0 is generally non-null. Nevertheless, most of the situations where the phase offset can be used (e.g. for radiation modeling) do not involve feedback connections with other blocks, so this feature just cost one additional per-sample multiplication.

Resonator Bandwidth

In the context of practical audio applications, it is often useful to examine the digital resonators defined by Eq. (3.5) as traditional bandpass filters [110]. The most relevant property of a bandpass filter, beside its resonant frequency f_r, is the *resonance bandwidth* Δf, which is typically defined as the width of the frequency interval where the magnitude response of the filter is above -3 dB of its peak response.

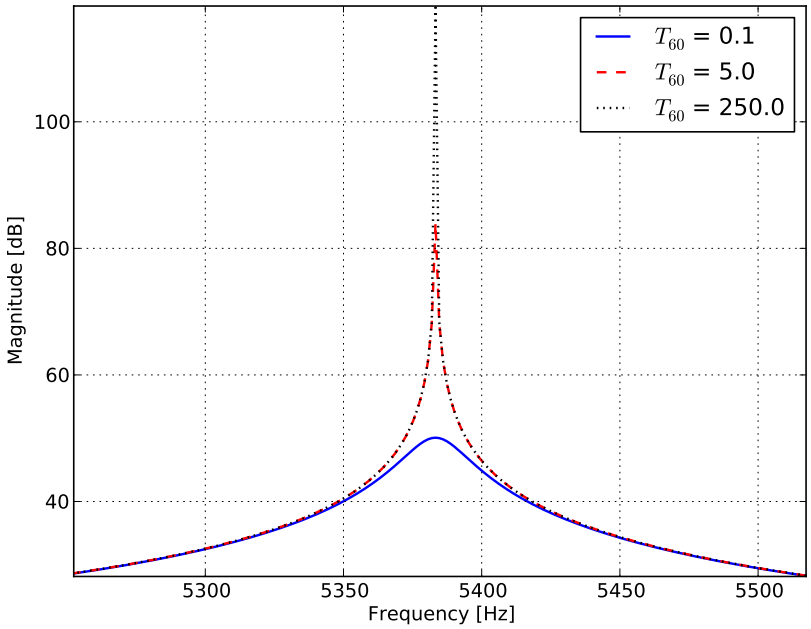

Fig. 3.2. Magnitude of the Frequency Response Function of three different digital resonators, having the same center frequency and varying decay times, expressed here as T_{60}. The resonating frequency is set at one of the discrete points used for transfer function evaluation, in order to have an exact (up to numerical round-off errors) magnitude value at that frequency.

It is useful to first define the Q factor of the resonator [91], which is usually expressed as the ratio of resonance frequency and bandwidth:

$$Q = \frac{f_\mathrm{r}}{\Delta f}. \tag{3.8}$$

We are only considering *underdamped* resonators, and so the constraints on the damping term stated in Chapter 2 translates into the inequality $Q > 1/2$. In terms of the previously discussed resonator parameters, it is easy to derive Q in terms of those values as:

$$Q = \frac{\pi\tau}{f_\mathrm{r}}. \tag{3.9}$$

Consequently, the bandwidth Δf of the resonator can be expressed as:

$$\Delta f = \frac{f_\mathrm{r}^2}{\pi\tau}. \tag{3.10}$$

Note that the expressions are not dependent on the particular discretization technique used. In fact, they are just relations between the continuous-time parameters and are also valid for the continuous-time resonator described by Eq. (2.18).

Eq. (3.10) states that the resonator bandwidth is inversely proportional to the decay-time of the resonator. It is also possible to note that the amplitude of the resonant peak varies significantly for different decay times, and also with frequency (not shown in the plot). This comes from the fact that we are forcing a constant time-amplitude peak A on the resonator. The problem is well known in digital filter design contexts [110]. For some audio applications, it might be better to have a *constant-peak-gain* resonator, by scaling the input coefficient b_1 with the magnitude response of the transfer function Eq.(3.5) computed at the resonant frequency (see [110] for the details in the case of all-pole and two-zero resonators). However, in this way we are obviously modifying the time-amplitude of the impulse response, loosening in some way the connection with the continuous-time modal resonator.

3.2 Space State Formulations

With the Direct Form I realization of the all-pole resonator, it is necessary to update at each time sample the current and previous output value of the filters, which are then stored in the delay elements shown in the diagram of Fig. 3.1). It is possible to rewrite the difference equation Eq. (3.6) in state-space form [110]:

$$\begin{bmatrix} y[n] \\ y[n-1] \end{bmatrix} = \begin{bmatrix} -a_1 & 0 \\ 0 & -a_2 \end{bmatrix} \begin{bmatrix} y[n-1] \\ y[n-2] \end{bmatrix} + \begin{bmatrix} 0 \\ b_1 \end{bmatrix} u[n]. \tag{3.11}$$

If we denote with the state with the vector $\mathbf{x}[n] = [y[n], y[n-1]]^T$, we can rewrite the filter update equation in canonical form as:

$$\mathbf{x}[n] = \mathbf{A}\mathbf{x}[n-1] + \mathbf{B}u[n] \tag{3.12}$$

$$y[n] = \mathbf{C}^T\mathbf{x}[n] + Du[n], \tag{3.13}$$

where $\mathbf{C}^T = [1, 0]^T$ and $D = 0$.

State-space representations are a popular form to express a different realization of the second-order resonator, derived from the *quadrature-phase* oscillator [79]. The idea behind these structures is to decouple the oscillating and damping behaviour in the numerical scheme, whereas both parameters are forcedly coupled in the recursive coefficients a_1, a_2 of Eq. (3.5). The difference equation[2] is expressed with a couple of variables $x[n], y[n]$ which have a relative phase-offset of 90 degrees:

[2] Note that, for consistency with the all-pole equation, we have chosen to add the input to the "sine" variable y instead than to x as it is done in [79].

$$\begin{aligned} x[n] &= R_x x[n-1] - R_y y[n-1] \\ y[n] &= R_y x[n-1] + R_x y[n-2] + b_1 u[n], \end{aligned} \tag{3.14}$$

with $R_x = R\cos(\omega_\mathrm{r})$ and $R_y = R\sin(\omega_\mathrm{r})$ being the cartesian coordinates of the complex pole. The correspondent canonical state-space form is:

$$\mathbf{x}_\mathrm{Q}[n] = \mathbf{A}_\mathrm{Q}\ \mathbf{x}_\mathrm{Q}[n-1] + \mathbf{B}_\mathrm{Q}\ u[n] \tag{3.15}$$

$$y[n] = \mathbf{C}_\mathrm{Q}^T\ \mathbf{x}[n] + Du[n]. \tag{3.16}$$

The subscript Q has been used in order to distinguish between the previous state-space relation. In this case the state-to-output matrix is $\mathbf{C}_\mathrm{Q}^T = [0,1]^T$. It is immediate to see that, from the form of the state update matrix $\mathbf{A}_\mathrm{Q}$, the quadrature-phase resonator requires twice the floating point operations at each sample compared to the all-pole resonator.

The advantage of the quadrature-phase resonator resides in the possibility to independently change at run-time the resonance frequency ω_r and the decay time, by acting on the pole radius R. With the all-pole resonator, it is only possible to change the decay time without artifacts, e.g. by precomputing a set of proper recursive coefficients a_1, a_2. This has been used for example for the modeling of musical instruments, such as the piano, which exhibit complex two-stage decay modal envelopes [135]. However, it is not trivial to change the frequency of an all-pole resonator without causing audible jumps in the output amplitude.

Another benefit of the quadrature-phase form is the direct access to the cosinusoidal variable $y_1[n]$. In physical modeling sound synthesis, this value is very useful for computing derived physical variables such as resonator velocity Eq. (2.35) or energy. We might ask if it is possible to compute such quantities from the states of the all-pole resonator, without recurring to numerical derivation techniques.

We consider a digital resonator in free evolution, i.e. the input signal u is null in Eq. (3.6). In order to take into account the previous history of the resonator, we can explicitly add an instantaneous amplitude A_n and a phase offset φ_n in the expression of the steady-state response:

$$y[n] = A_n\ R^n\ \sin(\omega_\mathrm{r} n + \varphi_n). \tag{3.17}$$

The goal is to express the cosinusoidal state $x[n] = A_n R^n \cos(\omega_\mathrm{r} n + \varphi_n)$ in terms of the two sinusoidal states $y[n], y[n-1]$. Expanding the expression for $y[n-1]$ and using some basic trigonometry yields to:

$$\begin{aligned} y[n-1] &= A_n R^{n-1} \sin(\omega_\mathrm{r}(n-1) + \varphi_n) \\ &= R^{-1} A_n R^n \sin((\omega_\mathrm{r} n + \varphi_n) - \omega_\mathrm{r}) \\ &= R^{-1} A_n R^n \Big(\sin(\omega_\mathrm{r} n + \varphi_n)\cos\omega_\mathrm{r} - \cos(\omega_\mathrm{r} n + \varphi_n)\sin\omega_\mathrm{r} \Big) \\ &= R^{-1} \Big(y[n]\cos\omega_\mathrm{r} - x[n]\sin\omega_\mathrm{r} \Big). \end{aligned} \tag{3.18}$$

Consequently, the quadrature component $x[n]$ can be expressed as:

$$x[n] = \frac{y[n]\cos\omega_\mathrm{r} - Ry[n-1]}{\sin\omega_\mathrm{r}}. \tag{3.19}$$

In terms of state-space vectors, the mapping between the quadrature-phase state vector $\mathbf{x}_\mathrm{Q}$ and the all-pole vector $\mathbf{x}$ is written with a transform matrix $\mathbf{S}$:

$$\mathbf{x}_\mathrm{Q} = \mathbf{S}\mathbf{x} = \begin{bmatrix} -\frac{R}{\sin\omega_\mathrm{r}} & \frac{\cos\omega_\mathrm{r}}{\sin\omega_\mathrm{r}} \\ 0 & 1 \end{bmatrix} \mathbf{x}. \tag{3.20}$$

The matrix $\mathbf{S}$ is non-singular under the assumptions we made for the resonator parameters. The inverse relationship, which can be used to go back to the all-pole states, is

$$\mathbf{x} = \mathbf{S}^{-1}\mathbf{x}_\mathrm{Q} = \begin{bmatrix} -\frac{R}{\sin\omega_\mathrm{r}} & \frac{\cos\omega_\mathrm{r}}{\sin\omega_\mathrm{r}} \\ 0 & 1 \end{bmatrix} \mathbf{x}_\mathrm{Q}. \tag{3.21}$$

From a computational point of view, the cost of integrating an all-pole resonator and then apply the transformation to get the quadrature-phase states is obviously larger than that required by directly running a quadrature-phase resonator. Nevertheless, the transformation turns out very useful whenever we need the flexibility of quadrature-phase filters *only for a small amount of time*. For example, energy estimation techniques as the one proposed in the next paragraph are usually performed once every audio buffer (usually 64-512 samples). Another situation typically happens when we require velocity information only during a short contact phase, as it is required for hysteretic impact models (see Chapter 4).

In these cases, we can simply use a cheap all-pole resonator and switch to and from the quadrature-phase form only for a limited amount of simulation time. Consequently, it is possible to save a significant amount of computations without sacrificing the accuracy of the results, since (up to machine floating-point precision) the transformation of Eq. (3.20) is numerically exact.

3.3 Derived Variables

In this section, we will make use of the state transform Eq. (3.20) to derive the expressions for some useful variables when the simple all-pole resonator is used.

3.3.1 Resonator Velocity

The expression for the velocity of a continuous-time modal resonator Eq. (2.35) can be easily used in discrete-time implementations once we know the quadrature-phase variable $x[n]$, which in this case comes from the derivative of the sinusoidal oscillation. Simply inserting the transformation Eq. (3.20) into the sampled version of Eq. (2.35) yields to the following equation:

$$\begin{aligned} v[n] &= b_{0,\mathrm{v}}y[n] + b_{1,\mathrm{v}}y[n-1] \\ b_{0,\mathrm{v}} &= \frac{\omega_\mathrm{r}\cos\omega_\mathrm{r}}{T\sin\omega_\mathrm{r}} - \frac{1}{\tau} \\ b_{1,\mathrm{v}} &= -\frac{R\omega_\mathrm{r}}{T\sin\omega_\mathrm{r}}, \end{aligned} \tag{3.22}$$

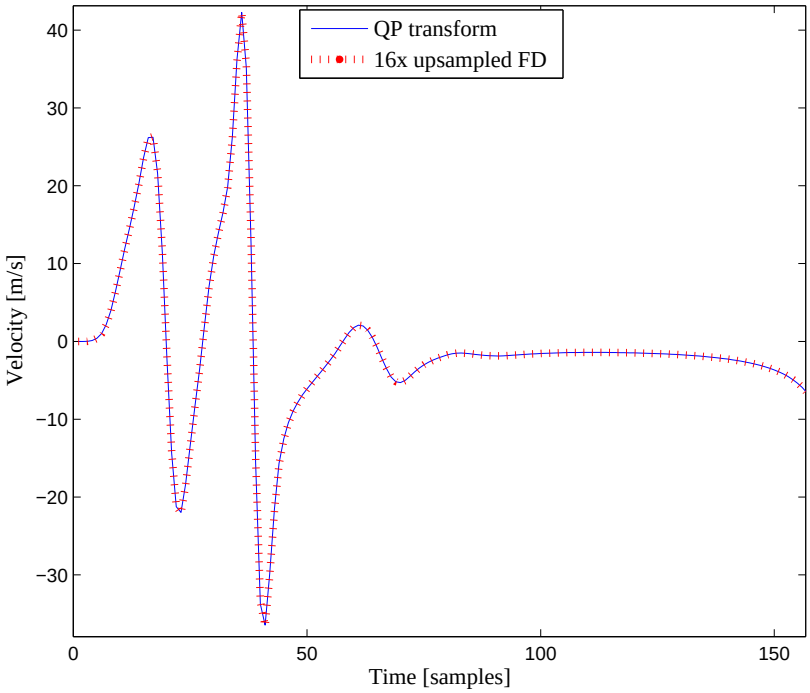

Fig. 3.3. Velocity of a modal resonator composed of 64 second-order filters. The output computed with the proposed quadrature-phase method (solid blue line) is plotted against the result obtained by finite difference (with 16x upsampling) of the resonator displacement.

where $v[n]$ is the discrete-time resonator velocity expressed in terms of the state variables $y[n], y[n-1]$. Fig. 3.3 shows one complex situation where the velocity of a 64-modes resonator struck by a nonlinear hammer model is computed in this way. The result is practically indistinguishable from the reference, which is taken as the velocity computed from a 16x upsampled displacement signal via first-order finite differences. Actually, the small differences come from the truncation error of finite differences, while the velocity computed by Eq. (3.22) is numerically exact. We have put the example just to show that, even if we did not directly considered the input signal in the derivation of Eq. (3.19), the results are still valid with transient excitations.

3.3.2 Instantaneous Amplitude and Phase

One common interpretation for the state variables of the quadrature-phase resonator is to associate them to the real and imaginary components of a complex resonator [79]:

$$z[n] = x[n] + jy[n]. \tag{3.23}$$

In this way, it is immediate to express the instantaneous amplitude $A[n]$ and phase $\varphi[n]$ using the well-known properties of complex numbers. With respect to the usual all-pole states $y[n], y[n-1]$ we have:

$$A[n] = \sqrt{\left(\text{cotan}\,(\omega_\text{r})\; y[n] - \frac{R}{\sin\omega_\text{r}} y[n-1]\right)^2 + y[n]^2} \tag{3.24}$$

$$\varphi[n] = \frac{y[n]\sin\omega_\text{r}}{y[n]\cos\omega_\text{r} - Ry[n-1]}. \tag{3.25}$$

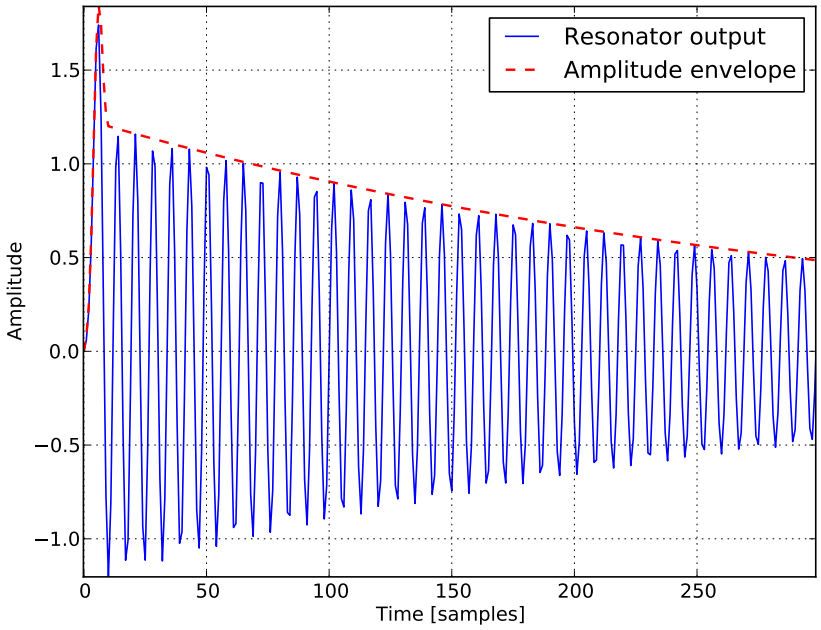

Fig. 3.4. Output of a 6 Khz resonator at $f_\text{S} = 44100$ Hz, with decay-time $T_{60} = 0.5$ sec, excited with a 12 samples Hanning window. The amplitude envelope extracted by Eq. (3.25 is plotted with a dashed red line.

Fig. 3.4 shows an example of envelope computation via Eq. (3.25). It can be seen that, even in this case, the technique works well also during the transient excitation.

A different method for the computation of the amplitude envelope of a second-order resonator has been used in [12]. There, the author approximated resonator velocity by the means of first-order centered finite differences, leading to the expression

$$A[n] = \sqrt{\left(y[n] + y[n-1]\right)^2 + \left(\frac{y[n] - y[n-1]}{\omega_\text{r}}\right)^2}, \tag{3.26}$$

which might be slightly faster depending on the implementation architecture (one less multiplication than Eq. (3.25)). However, while the approximation of Eq. (3.26) has found to be very good for most practical resonator parameters, it might have some problems in estimating the amplitude for fast-decaying, high-frequency resonators.

3.3.3 Residual Energy

Due to the elastic component in the modal ODE Eq. (2.18), the mechanical resonator stores an internal energy during the forced motion, and releases it while moving in free conditions.

For the continuous-time resonator, it can be shown [12] that the energy that remains in vibration from a generic time t_n to full decay is

$$\begin{aligned} E(t_0) &= \int_{t_n}^{\infty} y(t)^2 dt \\ &= \int_{t_n}^{\infty} \left(A_0 e^{-\frac{t}{\tau}} \sin(\omega_\mathrm{r} t) \right)^2 dt \\ &= \int_{t_n} A_0^2 e^{-\frac{2t}{\tau}} \frac{1 - \cos(2\omega_\mathrm{r} t)}{2} dt \\ &\simeq \int_{t_n}^{\infty} \frac{A_0^2}{2} e^{-\frac{2t}{\tau}} dt \\ &= \frac{A_0^2 \tau}{4} e^{-\frac{2t_n}{\tau}}. \end{aligned} \tag{3.27}$$

The approximation resides in neglecting the integral of the cosine function, which unless in the case of very small decay time is usually much smaller than the contribution of the exponential function.

The expression can be further developed into

$$E(t_n) = \left(A_0 e^{-\frac{t_n}{\tau}} \right)^2 = A(t_n)^2 \, \frac{\tau}{4}, \tag{3.28}$$

i.e. the residual energy is simply given by the square of the instantaneous amplitude multiplied by one fourth of the decay time. Using the formula for the all-pole resonator amplitude Eq. (3.25) yields to the following result holds in the discrete-time domain:

$$E[n] = \frac{\tau}{4} \left(\left(\operatorname{cotan}(\omega_\mathrm{r}) \, y[n] - \frac{R}{\sin \omega_\mathrm{r}} y[n-1] \right)^2 + y[n]^2 \right). \tag{3.29}$$

3.4 Conclusion

The second-order all-pole resonator is probably the most common block in audio physical modeling by modal synthesis. In this chapter, its numerical details were described and a comparison with the more expensive yet more flexible quadrature-phase resonator was given. Finally, a novel transformation procedure between the

two form of resonators has been proposed. This mapping comes useful in all the situations where the flexibility of the quadrature-phase resonator is required only for a short amount of simulation time. We also apply the mapping to get some derived physical quantities directly from the two states of the all-pole resonator.

4

Impact Modeling

In this chapter we review some techniques for the simulation of contact-like excitation between two modal resonators. Accurate simulation of contact mechanics phenomena can be very computationally demanding for high sample rates such those used for audio rendering. Therefore, hard assumptions and simplifications need to be made in order to formulate algorithms suitable for real-time implementation.

High-quality models of contact have been used by Bensoam for offline sound rendering, without and with friction simulation [15, 16]. The contact between two deformable objects, defined by their FEM discretization, is solved using a variational approach that consider a non-penetration constraint between the bodies. Much of the complexity of the method resides in the update at each time-step of the contact surface, which varies over time and requires expensive collision-detection algorithms for its update.

Another accurate yet expensive technique has been recently proposed by Zheng and James [138]. According to the authors, a frictional multibody contact formulation is employed together with an accurate numerical scheme that reduces the artifacts present in simpler models. Vibrational contact damping and coupled energy transfer between the bodies are also taken into account.

A simplified spatially-lumped contact model which is suitable for real-time implementations, derived from Hertz's contact law [55], is used here for audio-rate simulations following a common approach in sound synthesis [7]. Similar methods are widely used for robotic simulations, especially when interactive haptics feedback is provided [53]. By properly combining many short impacts, more complex interaction mechanisms such as rolling and crumpling can be implemented [101,104].

The definition of the contact model used within this thesis starts from the general architecture of the considered VR environment (see Fig. 1.2), where the audio synthesis module is controlled by a low-rate rigid body engine as e.g. those used for computer graphics simulations [19]. Within these engines, contacts between rigid bodies are already solved with e.g. impulse-based techniques [81]. However, the same algorithms cannot be use to model vibrational excitation by impact at audio rates, mainly because they assume the bodies to be rigid instead than elastic.

Nevertheless, rigid body simulators can prove useful in providing the control-rate information necessary to initialize a proper audio-rate impact model. Most of

the available rigid body engines, e.g. the *Bullet* framework [106] employed in this work, offer the possibility to query their internal collision detection engine about which objects are in contact and where the contact position is on their surface. Together with the information on the rigid body motion of the objects we are thus able to set the initial conditions required for the initialization of the audio-rate contact engine. Depending on the particular rigid body implementation, a threshold on the minimum relative velocity between two objects may be applied in order to avoid spurious contacts. Care needs to be taken in the calibration of the threshold, since high values can limit the possibility of modeling light micro-impacts such as those occurring in rolling motion.

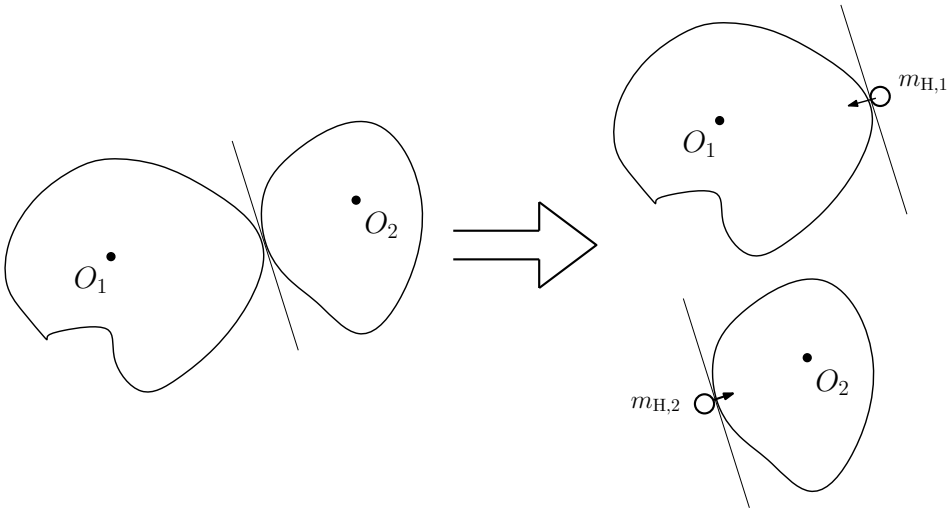

Fig. 4.1. Proxy impacts between two sounding objects. Instead than direct coupling of the two resonators, we connect each object with a point mass impacting along the surface normal and initialized using the parameters of the other body.

We assume that the contact time is shorter than the simulation time of a control-rate frame, which is usually a good approximation since contacts rarely exceeds 10 milliseconds and typical control rates are in the range from 30 to 100 Hz. There is no feedback from the audio-rate contact engine and the rigid body simulation, i.e. the two impact models (rigid and vibrational) are assumed to be independent for the sake of simplicity.

Another strong simplification resides in the nature of the coupling between the motion of two colliding objects. Full-coupling between objects is theoretically possible with the above mentioned assumptions, and has been tried as the first approach for the system hereby presented. However, full three-dimensional displacement of the surface nodes need to be computed, leading to a larger memory cost than the one required by storing only the normal component as it was proposed in Eq. (2.21). Moreover, we have to integrate the objects' rigid body motion in a three-dimensional space at audio-rate. Besides these computational disadvantages, there are more important problems that arise from the integration with a

system (the rigid body engine) over which we do not have full control. During the experiments, the author has found out that this kind of three-dimensional coupling requires implicit numerical methods for the integration of contact motion in order to guarantee the stability, when much more efficient explicit methods can be used for impacts with a point-like unidimensional mass as it is the case e.g. in [7] or in the modeling of piano hammer-string interactions [14].

Given the previous considerations on the complexity of full coupling with three-dimensional objects, we chose to approximate the contact interaction between two resonators using a *proxy impact* scheme. The idea behind this approach is depicted in Fig. 4.1. We associate to each object O_n $(n = 1, 2)$ a lumped mass $m_{\mathrm{H},3-\mathrm{n}}$ whose value and motion are set from the other object's state at the moment of the impact. In other words, the first object is excited by a point mass which correspond to the second object and viceversa. As an additional simplification, the objects are considered to have negligible rigid body motion during the short contact phase. In this way, with the proper choice of a coordinate system parallel to the contact surface normal and with the use of average weights given by Eq. (2.21), we have reduced the spatial domain of the simulation to one dimension.

The point masses $m_{\mathrm{H,n}}$ act as proxies between the two resonators and permit the reuse of simple and efficient methods for impact simulation. However, we are neglecting any energy exchange between the two objects, which somehow reduces the accuracy of the simulation. Nevertheless, this is not a strict assumption since we already assumed the absence of feedback from the audio-rate contact engine to the control-rate simulation, for practical integration with the rigid body simulator.

4.1 Hertz Contact Model

The Hertzian theory of contact approximates the force between two colliding objects with a lumped expression that depends on the relative motion of the bodies [55]. The main assumptions behind the theory are linear elastic behaviour[1], absence of friction and smooth contact surfaces which are small compared to the objects' radius.

With another physical approximation, the objects are permitted to penetrate with each other, so that the force can be conveniently expressed as a nonlinear funcion of the relative indentation δu between the objects:

$$F(\delta u) = \begin{cases} K(\delta u)^{\alpha} & \text{if } \delta u \geq 0 \\ 0 & \text{if } \delta u < 0. \end{cases} \tag{4.1}$$

In the original theory, the constants K and α are derived analytically from the objects geometry and elastic parameters for some simple shapes, e.g. for the contact between two elastic spheres or cylinders. However, following a common

[1] This assumption on the linearity of the system might seem counter-intuitive, since the force is expressed using a nonlinear relation. The nonlinearity in Eq. (4.1) comes from the time-dependent variation of the contact surface area. In other words, it can be said that Hertz theory trades time variance with nonlinearity for the system description.

approach used for sound synthesis applications, we use them here as free parameters set by the user, corresponding to the relative *hardness* between the objects. Within the context of lumped physical modeling, Eq. (4.1) can also be interpreted by considering a nonlinear spring connected in series with the point mass m_{H}. It has been noticed that large variations of K need to be applied in order to produce perceivable differences in the output sound. For this reason, it might be preferable to control the derived variable $\tilde{K} = K^{1/\alpha}$ which behaves better under this aspect [22].

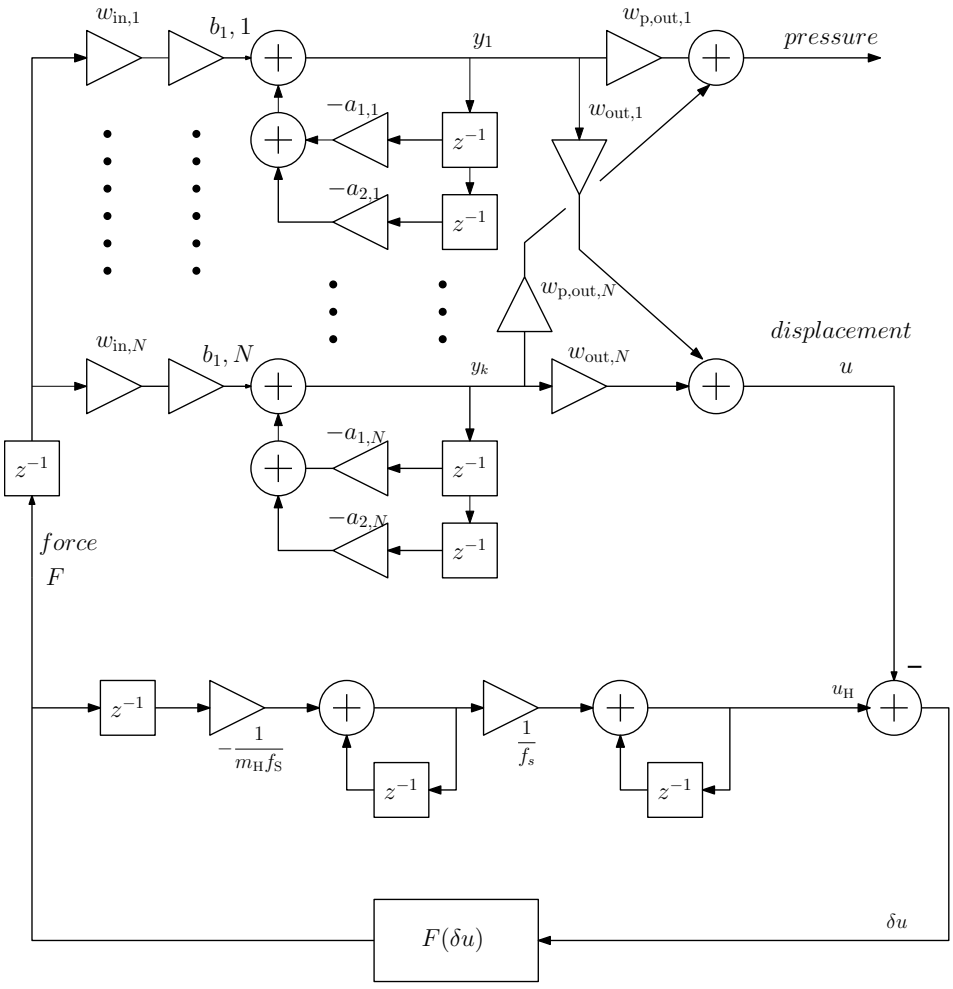

Fig. 4.2. DSP block diagram for the excitation of a modal resonator with a contact force computed from the nonlinear impact with a point mass.

With the proxy-mass impact approach, the indentation variable is computed as $\delta u = u - u_{\mathrm{H}}$, where u is the object displacement along the normal as computed from the weighted output of the resonator bank and u_{H} is the position of the proxy

mass, whose update is simply given by Newton's equation:

$$\ddot{u}_{\mathrm{H}} = -\frac{F(\delta u)}{m_{\mathrm{H}}}. \tag{4.2}$$

We chose to employ the explicit Euler method for the numerical discretization of Eq.s (4.1, 4.2). We can then complete the description of the audio-rate simulation with the inclusion of the feedback loop shown in Fig. 4.2. As it has been noted in [14], although the explicit scheme can be unstable, the presence of a double integrator limits such instabilities in a parameter range which is well above the common values used in physical simulations of real objects.

While not often used in previous sound synthesis realizations, it is easily possible to extend Eq. (4.2) in order to include the effects of viscous damping and gravity:

$$\ddot{u}_{\mathrm{H}} = -\frac{F(\delta u)}{m_{\mathrm{H}}} + b_{\mathrm{loss}}\dot{u}_{\mathrm{H}} + g_n, \tag{4.3}$$

where b_{loss} is a generic viscous coefficient and g_n is the component of the gravity force along the normal to the contact surface. Example force simulations with varying loss coefficients are shown in Fig. 4.2 for the contact of a proxy mass against a bell-like object with a radius of 50cm and implemented by approximately 200 resonators.

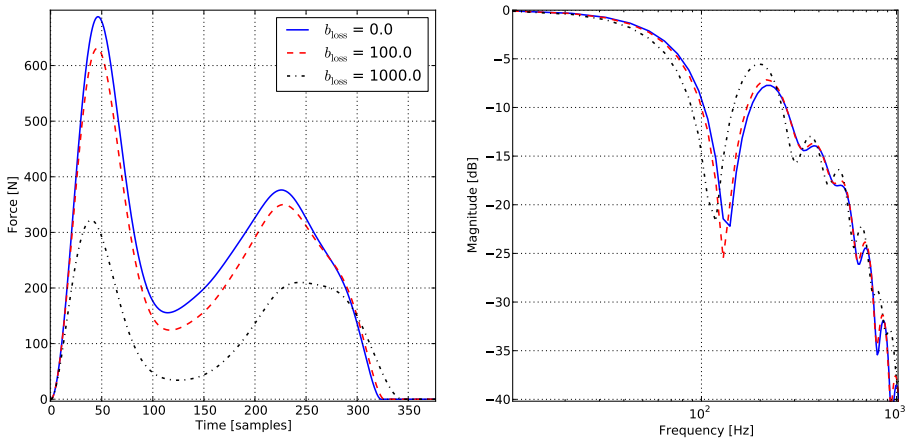

Fig. 4.3. Contact forces between a proxy mass and a bell-like FEM resonator implemented with modal synthesis, for varying values of the viscous damping parameter b_{loss}. Initial impact velocity is $5m/s$ and the contact parameters are $\tilde{K} = 5000N/m$, $\alpha = 3.0$. The sampling rate is set to 44100 Hz.

4.1.1 Hysteresis

Experimental measurements have shown that impacts between real objects are bettered approximated with a hysteretic contact force, i.e. the force law shows a

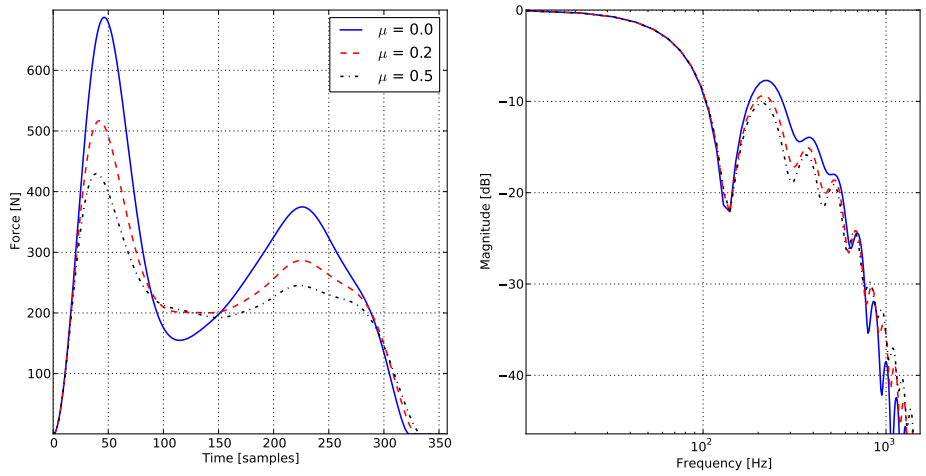

Fig. 4.4. Contact force simulation with the same parameters as those used for Fig. 4.3, but with varying hysteresis coefficient μ.

different behaviour between the compression and relaxation phases of the impact. A compact lumped model which includes this effect has been developed by Hunt and Crossley in 1975 [63] and first used for sound synthesis by Avanzini et al. [7, 104]:

$$F(\delta u, \delta \dot{u}) = \begin{cases} K(\delta u)^{\alpha} \cdot (1 + \mu \delta \dot{u}) & \text{if } \delta u \geq 0 \\ 0 & \text{if } \delta u < 0. \end{cases} \tag{4.4}$$

As it can be seen from the previous equation, hysteretic behaviour is modeled with a term dependent on the relative compression velocity $\delta \dot{u}$ and weighted by the coefficient μ. Fig. 4.4 shows the resulting forces for varying values of μ. The plotted spectral magnitudes show how with higher hysteresis coefficient the system assumes a more pronounce low-pass characteristic.

Numerical discretization is very similar to non-hysteretic Hertz contacts. We use here the same state-space description of the proxy mass in terms of its position and velocity as it is proposed in [104]. However, instead of approximating the resonator velocity $\dot{u}$ with e.g. first-order time-domain finite differences of the object displacement u, we are able to get an "exact" value by using the transformation to quadrature-phase components presented previously in Eq. (3.22). Especially if there are high-frequency modes in the resonator, numerical differentiation can lead to inaccuracies or worse instabilities in the system. This can be informally explained in DSP terms by noting that numerical differentiation acts as a high-pass filter which, combined with the bandwidth expansion of the nonlinear function, can quickly drive the system beyond the Nyquist limit.

Various numerical schemes for the discretization of the contact model described by Eq. (4.1) have been recently reviewed in terms of energy conservation for the case of the impact of an inertial mass against a rigid surface [92, 93]. In the same work, analytical results on the Hamiltonian of the system at the end of the contact phase are used to improve the accuracy of numerical schemes. However, results

directly applicable to the impacts with resonating objects are still an ongoing research.

4.2 Feed-forward Approximations

Hertzian-like impacts involve a quite strong simplification of the complex physical interactions happening during contact, necessary to achieve efficient solutions to the problem. Nevertheless, there are situations where even simpler and faster methods might be chosen. The most common approach, followed among the others in [125], consists in the use of a precomputed feed-forward force, defined e.g. by a raised-cosine function:

$$F(t) = \begin{cases} A_{\mathrm{F}} \left(1 - \cos(\frac{2\pi t}{t_{\mathrm{F}}})\right) & \text{if } t \leq t_{\mathrm{F}}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

The signal-based parameters are the force amplitude A_{F} and the duration of the contact t_{F}. The direct relationship with the user interaction parameters is lost with this approach and is not possible to model of rapid multiple impacts due to the object's vibration (see e.g. the two peaks in Fig. 4.3). On the other hand, these parameters are orthogonal if considered as conventional synthesis controls of *gain* and *cutoff frequency*, whereas the same variables are intrinsically coupled when using physical parameters. Thus, the signal-based parameters might be preferable whenever the goal is to precisely control the sound for a restricted class of objects. Complex feed-forward excitation signals, including the modeling of repeated pulses, have been developed for ad-hoc cases such as hammer-string contact force approximations in the piano [127, 134].

A common problem of simple feed-forward excitation function such as Eq. (4.5) is that it is not possible to differentiate the influence of impact velocity from the hardness of the objects, since both lead to shorter, higher-amplitude signals. While comparing feed-forward formulations to the output of impacts against a hard surface, the author has empirically found out that a better model may be given in terms of Kaiser windows [91]. Preliminary experiments seem to show that the β parameter of these windows can be associated with material properties (mass and hardness), in a way that does not depend on the initial impact velocity. Further work needs to be carried out in this sense in order to validate these observations by e.g. analytical solutions of the contact problem.

5

Efficient Algorithms for Modal Synthesis

This chapter deals with the computational aspects of modal synthesis, analyzing optimizations of the parallel second-order bank of resonators that can give the same perceptual results at less expense in CPU time. This kind of optimizations are often necessary when we need to simulate in real-time many objects, each one composed by a large number of modes. Although some non-trivial psychoacoustics phenomena such as frequency masking [34] can be exploited, we make use only of simpler principles like amplitude thresholding. In this way, the resulting algorithm might not be the fastest possible, but it is easy to assess the perceptual equivalence without resorting to psychoacoustics experiments.

A different approach that can be used to effectively render many modes in real-time is to make use of parallel and vectorial operations available on modern hardware. The parallel bank of second-order resonators, in the absence of feedback interaction, can be clearly classified as an *embarrassingly parallel* algorithm [61]. Therefore, optimal linear speed-up can be obtained on most parallel hardware for feedforward structures, while with the proper assumptions even feedback interactions can benefit from parallelization. We do not explicitly analyze parallel structures for the algorithms presented in this chapter, although the question has been considered in the choice of methods. Instead, an example of data-parallel implementation of a modal synthesis engine is reported in Appendix A for the case of general purpose computer CPUs.

5.1 Resonator Pruning

The first optimization that can be used in order to decrease the computational cost of the system is simply a reduction of the number of resonators. Modal synthesis frameworks are intrinsically scalable, in the sense that we can for example put a limit on the highest frequency of the modes in order to reduce the total cost. Clearly, this trivial limit can have an impact on the overall sound quality of the synthesis. It is thus more interesting to derive resonator pruning techniques that do not have impact on the perceptual result. The techniques can be divided into off-line approaches, where we completely exclude some resonators from the synthesis,

and on-line methods that activate and deactivate the modes by evaluating their impact on the current simulation instant.

In the context of sound synthesis for virtual reality environments, some methods have already been proposed in literature. Bonneel et al. [21] developed a sound synthesis pipeline based on the short-time Fourier Transform of a damped sinusoid. The method can provide a high speed-up over time-domain synthesis, at the cost of increased latency and by assuming a sparse mode distribution in the frequency domain. This assumption is valid when modal parameters are estimated using signal analysis techniques, but is often violated when FEM or other numerical methods are used for the computation of complex-geometry objects resonances. Moreover, some artifacts in the attack can be present depending on the frame size and the type of windows chosen, and nonlinear feedback between resonators is not possible.

Van den Doel et al. [126] discussed a method for reducing the number of modes estimated from impulse responses of everyday objects. They exploit masking phenomena by choosing a small subset of the original detected modes with appropriate thresholding on a Bark scale representation. Another signal based technique, developed in a more general context, has been investigated extensively by Bank [10,11]. Here, the resonator poles can be estimated with various methods (e.g. Prony or warped filter design) and the zeroes are then optimized in a least-square sense to match the target impulse response.

Run-time pruning of the resonators has been used by Van den Doel in another work [41], where non-audible modes are turned off during the simulation based on frequency masking considerations. There is a trade-off between computational speed-up and sound quality, and the pruning procedure itself has a non-negligible cost. The computational speed-up that can be achieved ranges between 2 and 10. However, if vector computation of the modes is employed, the method might not be as profitable as in the scalar case, since in most cases only one or two of the modes inside a vector block are turned off at run-time. Moreover, this type of run-time resonator deactivation requires a conditional statement inside the loop for the update of the resonators. If sample-by-sample synthesis is required, which is the case with feedback interactions, the evaluation of the condition can often lead to drastic reductions e.g. in compiler and CPU pipeline optimizations [42].

In this work, a conservative approach to mode pruning is taken, leading to a modest amount of computational savings but without affecting the quality of the synthesis. Off-line mode pruning is done by simply putting a threshold on the absolute value of pressure output weights (see Eq. 2.36). We do not put any threshold based on the input weights vectors, i.e. the modal shapes, since these can vary significantly depending on the impact position on the surface. Using for example a threshold of -60 dB from the highest pressure weight value, the average number of modes that can be throw away is around 5-10% of the total, depending on the geometry of the object.

A more efficient reduction of the resonators is actuated at run-time during the simulation. We consider for this purpose the residual energy of the resonators as in Eq. (3.29), evaluating only those objects which are evolving in a steady-state situation, i.e. after the contact phase. In order to not interfere with possible vector or data parallel implementations, we perform this computation on *blocks of*

resonators, where the size of the block N_B is related to the dimension of the vector computation unit and usually varies between 4 and 16 on common hardware. Since the decay times distribution is monotonically decreasing, the algorithm evaluates first the highest-frequency resonators. The procedure is performed once per every audio buffer (64-512 samples) and consists of the following steps:

1. For each object of N resonators, a counter $N_a \leq N$ is used for tracking the number of active modes.
2. When a contact is detected and the impact model is activated, N_a is set to the maximum value N.
3. After the short contact phase, the residual energy E_{last} of the resonators in the interval $[N_a - N_B, N_a]$ is computed by summing the energies of the single modes obtained by Eq. (3.29).
4. The energy computed in the previous step is compared to the mean-square value of the output of all the objects in the current audio buffer. If the difference is higher than a determined threshold, e.g. -90 dB, the number of active resonators for the object is updated as $N_a \leftarrow N_a - N_B$.

In typical simulation conditions, where the average decay time of the modes is short (less than 1 second) and impacts are relative sparse, the average number of active resonators at each instant is between 10 and 30 percent of the total. This technique has been recently employed within a physical-based virtual piano synthesizer [115, 134] and, together with some advanced polyphony management methods, makes possible the reduction from more than 15,000 to about 5,000 resonators without noticeable artifacts, even with complex musical interactions.

5.2 Multirate Implementations of a Resonator Bank

One nice properties of the resonators used for modal synthesis is that their bandwidth is very small compared to the sampling rate of the overall system, e.g. 44100-48000 Hz. Therefore, it is possible to employ different sample rates for the simulation of the modes in a *multirate* fashion [82, 112, 122]. For example, if all the modes could be simulated at half the required sampling frequency, it would be possible to achieve almost a 2x speed-up in the computation.

The approach followed for this work is largely inspired by the work of Phillips [95, 96], developed within the context of traditional additive synthesis. Very recently, a similar approach has been taken for the development of a multirate modal synthesis engine used for sound rendering in videogames [76], although only with feedforward excitations.

5.2.1 Multirate Filter Banks

The key components of a multirate system are the procedures which define the signal transformations between the bands. We consider here only sampling rate conversions by an integer factor N, since they are faster and much easier to implement than non-integer conversions [82].

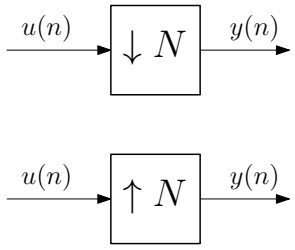

Fig. 5.1. Block representation of downsampling and upsampling operators.

The most basic operations that can be defined to this purpose are the *upsampling* and *downsampling* operators, whose block-diagram representations are shown in Fig. 5.1. Following the notation used in [112], we define the operator $\mathrm{STRETCH}_N$ for the upsampling by a factor N as

$$y(n) = \mathrm{STRETCH}_{N,n}(u) \doteq \begin{cases} u(n/N), & \frac{n}{N} \in \mathbb{Z}, \\ 0, & \text{otherwise.} \end{cases} \tag{5.1}$$

The operator acts by simply inserting $N-1$ zeros between the values of the input signal $u(n)$. In the frequency domain, the upsampler modifies the Zeta-Transform of the input $U(z)$ by replicating the spectrum N times, i.e. $Y(z) = U(z^N)$.

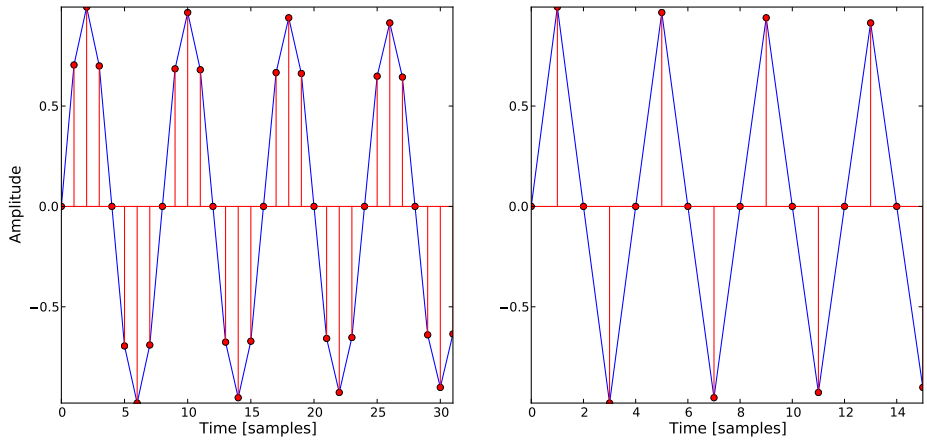

Fig. 5.2. Downsampling by a factor $N = 2$ of a discrete damped sinusoidal signal.

In a similar way, we define the *downsample* operator by taking one every N-th sample of the input signal:

$$y(n) = \mathrm{DOWNSAMPLE}_{N,n}(u) \doteq u(Nn), \quad n \in \mathbb{Z}. \tag{5.2}$$

In Fig. 5.2 the result of downsampling by a factor $N = 2$ is shown in the case of a damped sinusoidal signal. The interpretation of downsampling in the frequency

domain can be thought as expanding the frequency axis by a factor N and then wrapping unto itself N times. With respect to the Zeta transforms of the input and output signals we have thus

$$Y(z) = \frac{1}{N} \sum_{m=0}^{N-1} U(e^{-\frac{2\pi j m}{N}} z^{1/N}). \tag{5.3}$$

Contrarily to most DSP processing blocks, these operators are not time-invariant. However, it is possible to commute up and downsamplers with *memoryless* computational blocks like adders and multipliers [112].

In order to suppress the distortions in the frequency domain caused by the upsampling and downsampling operators, *anti-aliasing filters* need to be employed [82]. These are low-pass filters, which are typically designed as linear-phase FIR filters to avoid further distortions and whose cutoff frequency is set at $\omega_c = \pi/2$ (i.e., Nyquist frequency) with respect to the lowest sampling rate considered. Commonly, filters used together with downsampling operations are called *decimators*, while the name *interpolators* is used in the upsampling case. The details of these filters are discussed in Sec. 5.2.4. In the case of downsampling, it is not possible to change without artifacts the sampling rate if there are frequency components in the input signal above $\omega = \pi/4$, i.e. half the Nyquist frequency of the lower band, independently from the precision of the anti-aliasing filter used.

A generic structure for the multirate implementation of a modal-based resonating object is shown in Fig. 5.3, using a common form employed e.g. in audio compression systems [112]. An *Analysis Bank*, composed of downsamplers and interpolating filters, is used to transform the excitation signal u at the full sampling rate $f_{S,0}$ into M different frequency sub-bands, each running at a sampling rate $f_{S,m}$, $1 \leq m \leq M$. These sub-bands do not necessarily have to be separated or, in other words, overlapping of the bands is possible. A separate parallel bank of second-order resonators is then used to synthesize the output signals in the different sub-bands. In Fig. 5.3 we have omitted for the sake of simplicity the output weights of the previous block diagram (see Fig. 3.1), and a single output signal is shown, although the system can be clearly extended to the multi-output case of pressure and displacement. Finally, the sub-bands signals are transformed back to the original sample rate and recombined using a *Synthesis Bank*, typically implemented using upsamplers and interpolating filters.

In the case of feedforward excitation signals, the analysis stage can be avoided, since it is usually possible to directly synthesize the sub-band signals at the proper sampling rate. The synthesis bank is then the most important part of the system, and particularly the performance of its interpolating filters is essential in order to avoid audible artifacts. As we have already mentioned, the upsampler operator introduces *aliasing replica* of the normal modes. Non-ideal interpolating filters cannot completely suppress these aliasing components on the whole frequency spectrum. Therefore, there will generally be frequency intervals, named *deadbands* in [95], which cannot be used for the synthesis at a particular sampling rate.

The multirate architecture of Fig. 5.3 can be naturally extended to the case of multiple resonating objects. With the feedforward excitation approach, only

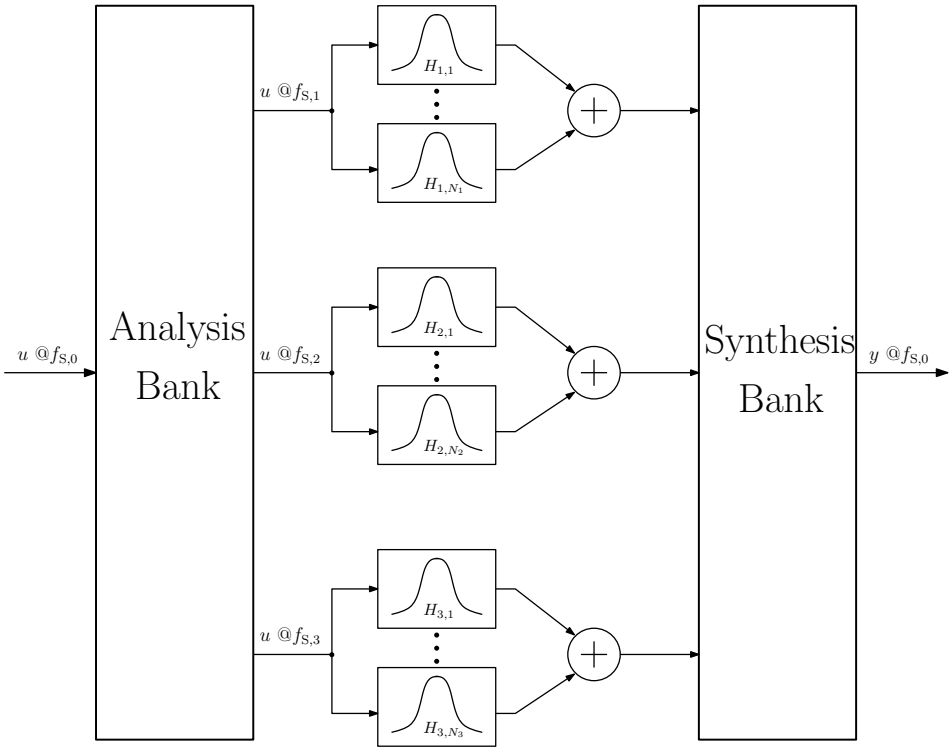

Fig. 5.3. Multirate implementation of modal synthesis resonators. The excitation signal u is divided by the Analysis Bank into different frequency bands (3 in this picture), each one with a different sampling frequency $f_{S,n}$. A set of resonators is used in each band, and the output signals are then recombined at the original sampling rate using a Synthesis Bank.

one synthesis bank is necessary for all the objects, since the lower rate outputs of multiple sounding objects can be summed before upsampling.

On the other hand, feedback interactions are hard to implement using a multirate architecture. The main problem comes from the inevitable phase delay of the interpolation filters in the analysis and synthesis banks. In the presence of a nonlinearity in the feedback loop, this delay will lead to numerical instabilities, which are known to arise even in the case of a single-sample delay [23]. In the following sections, we will see some techniques to bypass this limitation in order to arrive at multirate systems suitable for modal synthesis.

5.2.2 Prior Work

Multirate schemes for modal synthesis have been previously proposed in literature, although their use is not very common. Bank [9] suggested a multirate implementation of a parallel bank of feedforward resonators used for simulating the beating and two-stage decays phenomena in piano strings. He proposes a multistage structure with a *dyadic* frequency subdivision, i.e. the sampling rates of the subbands

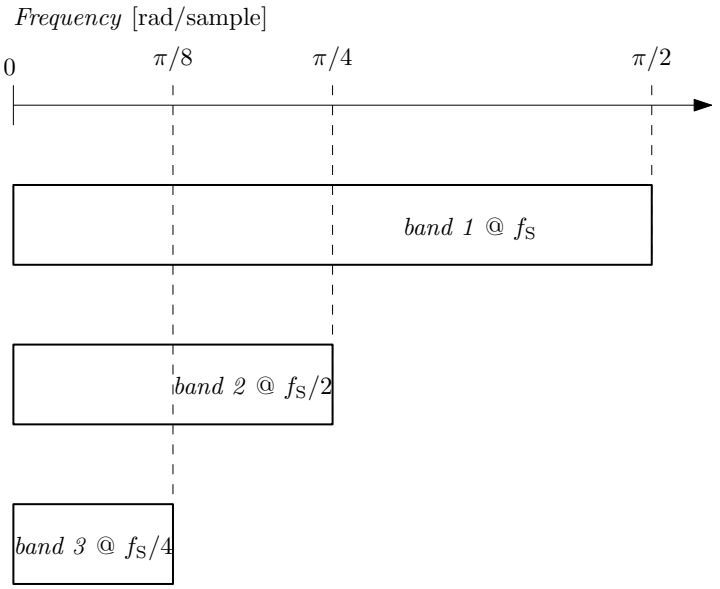

Fig. 5.4. Overlapping dyadic frequency subdivision used e.g. in [9] for the multirate implementation of the parallel bank of resonators.

are $f_{S,0} = f_S$, $f_{S,1} = f_S/2$, ..., $f_{S,m} = f_S/2^m$. Each resonator of frequency f_r is then allocated in the m-th band with the lowest possible sampling rate, satisfying the Nyquist criterion $f_r < f_{S,m}$, eventually with some headroom to take into account the non-ideal behaviour of the interpolating filters near the Nyquist frequency.

The division of the frequency spectrum of this method is shown in Fig. 5.4. Using the naming convention of Phillips [95], this is a *fully-overlapped* octave-spaced subdivision. The advantage of having overlapping subbands is that it is always possible to find a suitable band for each resonance frequency f_r in order to synthesize the mode without aliasing: in the worst case, the full sampling rate f_S can be used. However, the logarithmic distribution of the band edges implies that only few low-frequency resonators can be synthesized with the most efficient sampling rates, thus limiting the speed-up that can be achieved when many high-frequency modes need to be simulated.

A different subdivision scheme was employed by Trautmann and Rabenstein [121], where a DFT-like subdivision of the frequency spectrum is presented for the multirate realization of modal-based string instruments (see Fig. 5.5). In this case, the subbands are non-overlapping and the band edges are linearly spaced. The computational speedup compared to full rate is equal to the number of subbands (i.e. 6x in the case of six subbands), minus the cost for the analysis and synthesis banks. In the actual algorithm, there is actually a mixture of 6x and 4x downsampling bands and nonlinear feedback interaction are possible with a modified algorithm running at the lower sampling frequencies.

The main drawback with non-overlapping bands is that there will always be aliasing for resonators falling inside the transition intervals (the above mentioned

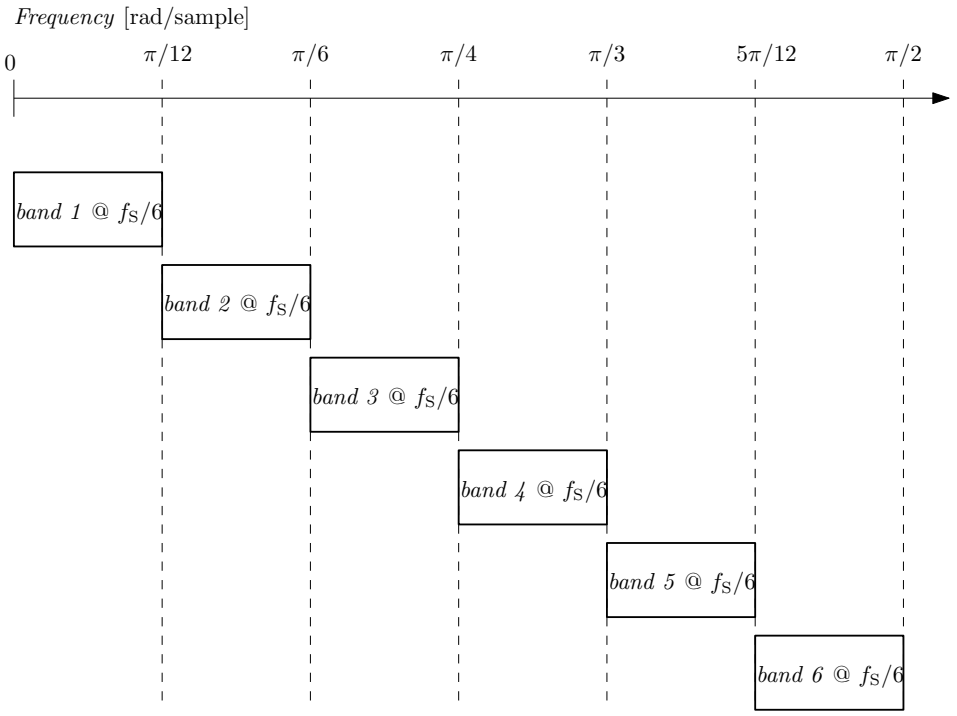

Fig. 5.5. Uniform frequency subdivision in 6 equal non-overlapping subbands, as used in [121].

deadbands, or as they are called in [121], "don't care bands"). Moreover, the algorithm that is used for the feedback interaction, besides being quite complex in its formulation, is not equivalent to the full rate excitation, due to aliasing and delays between modes. This leads to audible artifacts, which are particularly noticeable with high downsampling ratios.

5.2.3 QMF Subdivision

In the context of additive synthesis, Phillips [95] suggested the use of Quadrature Mirror Filters (QMF) subdivision as a very good compromise between band overlapping and sampling rate reductions for multirate systems. The technique is based on a recursive binary division of the frequency spectrum, and is applied here to modal synthesis exploiting some properties of the considered resonators such as their fixed frequency and narrow band.

We will neglect for the following of the section the details of the analysis bank, since the problem will be approached later from a different perspective. Therefore, the aim will be to develop a synthesis bank that is able to reconstruct the output signal from a bank of resonators running at lower sampling rates.

The use of QMF banks is well consolidated for e.g. audio compression and coding, where critically sampled perfect-reconstruction multirate systems are often developed with this technique [82, 122]. QMF banks are defined in a recursive way

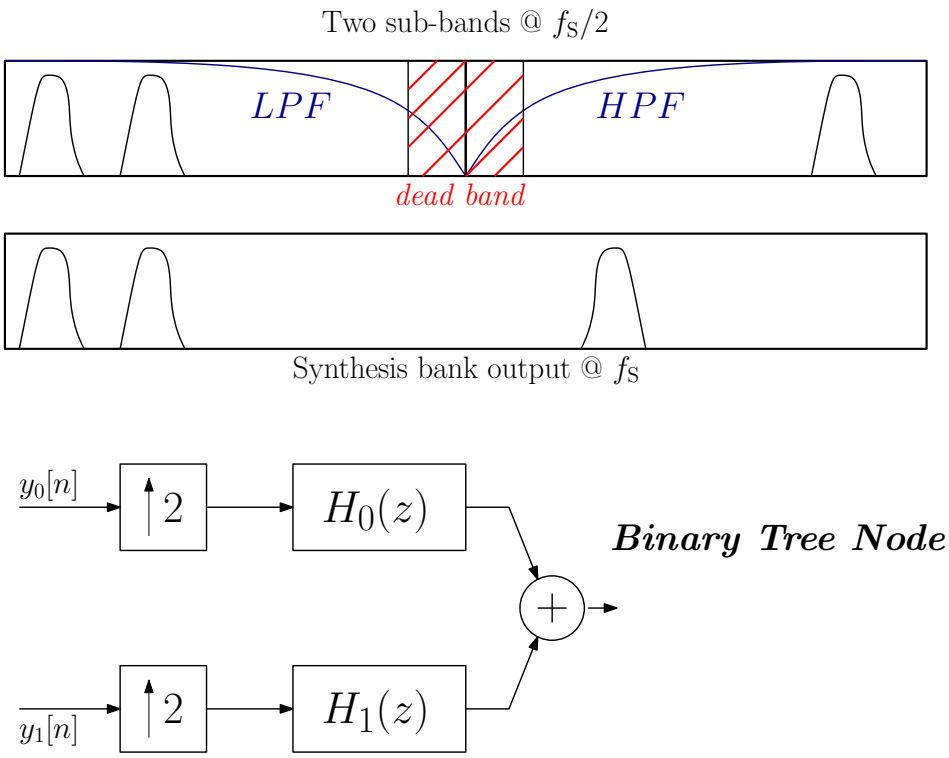

Fig. 5.6. Quadrature Mirror Filter subdivision in two equal bands. The signal synthesized in the lowband can be reconstructed with a conventional 2x upsampler and a lowpass interpolating filter, while for the high-frequency band a mirror high-pass filter is used. The operation can be seen as a node in a binary tree for recursive QMF applications.

starting from the basic binary division shown in Fig. 5.6 for the synthesis stage. The two bands are separated at the angular frequency $\omega = \pi/2$, i.e. half the Nyquist frequency, and each one is synthesized with a 2x downsampling rate. The signal $y_0(n)$ in the low-frequency band can be brought to the full sampling rate by simply using an upsampler followed by a low-pass interpolating filter $H_0(z)$. For the high-frequency band, instead, we synthesize the components in a mirror frequency scale and apply a high-pass filter $H_1(z)$ for the synthesis. The process can be thought as "keeping the aliased component" that comes from the upsampling operator instead than discarding it. The following symmetry constraints apply on the synthesis filters $H_0(z)$, $H_1(z)$ and on their impulse responses $h_0(n)$, $h_1(n)$:

$$H_1(z) = H_0(-z) \tag{5.4}$$

$$h_1(n) = (-1)^n h_0(n). \tag{5.5}$$

When the interpolation filters $H_0(z), H_1(z)$ are linear-phase FIR filters, it is possible to efficiently both the filters using a polyphase decomposition [82], thanks to the high redundancy in the coefficients of $h_0(n)$ and $h_1(n)$.

In order to synthesize a parallel bank of resonators with a binary QMF bank, it is necessary to "invert" the resonance frequencies $w_r' = \pi - w_r$ whenever $w_r > \pi/2$.

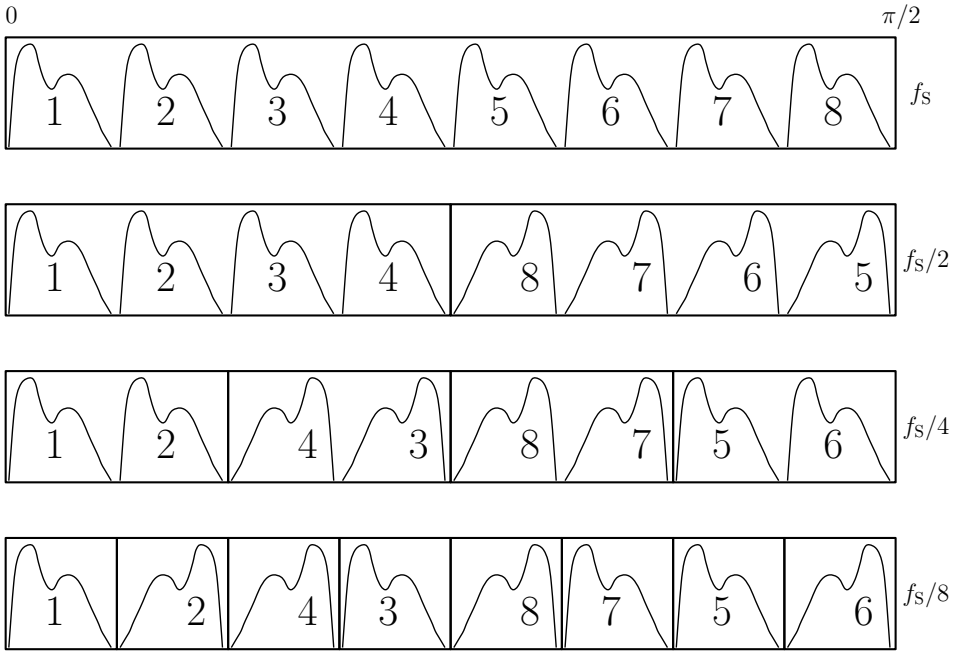

Fig. 5.7. Recursive binary subdivision of the frequency spectrum using QMF stages. Notice how the spectra in the subbands are shifted and combined according to the mirroring process of each step.

Notice how the system is much more efficient than the full-overlapping dyadic subdivision of Fig. 5.4, since we can use the lower sampling rate even for high-frequency resonators. On the other hand, the basic QMF bank is still a non-overlapping subdivision scheme, and thus no resonator can be synthesized in the deadbands near half the Nyquist frequency due to the non-ideal response of the interpolation filters.

The basic binary QMF bank of Fig. 5.6 can be combined in a recursive way to form a dyadic, overlapping frequency subdivision, which is illustrated qualitatively in Fig. 5.7 and in Fig. 5.8. The subbands are divided depending on their depth m in the tree, which corresponds to a sampling rate of $f_S/2^m$. The output of the signals coming from the 2^m bands at depth m is converted to the $m-1$-th band using $m-1$ different QMF basic blocks. The structure can then be described as a *multistage* synthesis bank, since for example three binary stages are required to convert a signal from $f_S/8$ to f_S. The total number of frequency bands, counting all the sampling rates, is $2^{M+1}-1$, where M is the maximum depth tree depth.

With respect to the frequency subdivision scheme of Fig. 5.8, it is possible to notice how the deadbands caused by the interpolation filters are inherited by subsequent bands in the hierarchy, due to the multistage synthesis process. This poses a limit on the maximum depth of the binary tree, i.e. on the lowest sampling rate employed by the system. Moreover, each stage adds an amount of latency due to the need to compensate the group delay of the interpolation filters. We

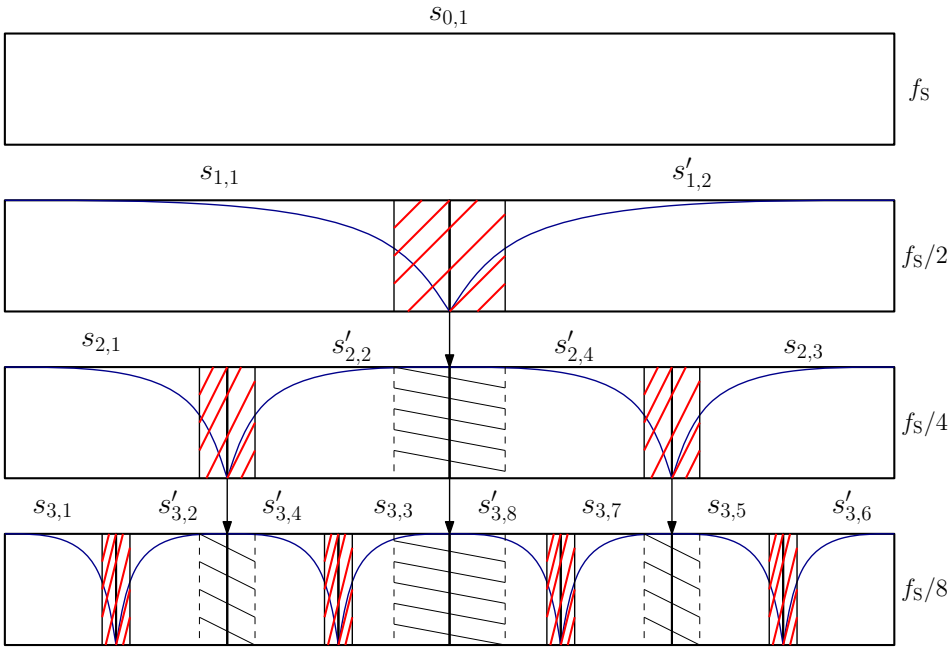

Fig. 5.8. Hierarchical division in subbands with a QMF tree. The depth of a subband $s_{m,k}$ is m and its sampling rate $f_S/2^m$. The notation $s'_{m,k}$ indicates those band whose frequency spectrum is mirrored (see Fig. 5.7). Deadbands in the upper bands (dashed red zones) are inherited by the lower ones (inverted dashed gray lines), because of the multistage structure of the synthesis bank.

consequently chose to limit the structure to a maximum depth $M = 3$, as it is done also in [95] and [76].

With recursive QMF trees, it is necessary to find out in which band a resonator of given angular frequency ω_r will be synthesized. For the base, full sampling rate band we have trivially $\omega_{r,0} = \omega_r$, while for the following stages we can use the recursive relation

$$\omega_{r,m+1} = \begin{cases} 2\omega_{r,m} & \text{if } \omega_{r,m} < \frac{\pi}{2} \\ 2(\pi - \omega_{r,m}) & \text{if } \omega_{r,m} \geq \frac{\pi}{2}. \end{cases} \tag{5.6}$$

In order to avoid the synthesis of components having audible aliasing counterparts, we need to map the deadbands edges on each subband $s_{m,k}$. We can then compute the deepest possible frequency $\omega_{r,M}$ and check if it falls inside a "dead" region in its subband $s_{M,k}$. If this is the case, we go back to the upper band having depth $M-1$ and repeat the test, until eventually the single full sampling rate band is reached. The edges of the deadbands depend on the interpolation filters used for each stage, and their computation is discussed in the next section. When the deadbands are narrow enough, it is possible to put a good amount of resonators in the lowest band, and only few modes (usually less than 10% of the total) have to be synthesized at full sampling rate.

5.2.4 Interpolation Filters Design

The subject of designing linear-phase FIR filters for antialiasing and interpolation in multirate systems is widely discussed in literature [82, 91]. Probably the most common example in this context are Nyquist M-th band filters, also called half-band filters when - as it happens in our case - we need to compensate a 2x upsamping/downsamping stage. The ideal half-band filter has a magnitude response which is unitary in the interval $[0, \pi/2]$ and null from $\omega = \pi/2$ to Nyquist. Practical filters are instead designed imposing some specifications on the magnitude response, particularly the passband edge frequency $\omega_\mathrm{p} < \pi/2$ and the stopband edge frequency $\omega_\mathrm{s} = \pi - \omega_\mathrm{p} > \pi/2$. One nice computational property of Nyquist half-band filters resides in the fact that half of the samples in their impulse response are null, and so their implementation requires less multiplications than general linear-phase filters.

However, especially on modern hardware where the cost of multiplications is comparable to the one of additions and often much cheaper than the cost for memory accesses, it might be better to find alternative design methods which optimize the length of the impulse response. This has the additional advantage of reducing the latency of the system, which is a relevant factor in interactive applications. In order to compensate for the group delay of an order D FIR filter, the usage of a buffer having length at least $D/2$ is required.

In our modal synthesis frameworks, resonators have narrow bandwidths and fixed (i.e., not time-varying) frequencies. Therefore, relative small inaccuracies in the passband of the interpolation filter can be compensated by changing the amplitude and phase offsets of the resonators, as it has been suggested in [9]. In order to do so, we simply have to compute the frequency response of the filter at the resonance frequency, $H_m(\omega_{\mathrm{r},m})$ for the m-th stage in the synthesis bank, and then scale the input coefficients of the resonator by its inverse.

Phillips [95] suggested the use of the well-known Parks and McClellan equiripple method [94] for the design of interpolation filters within a multirate additive synthesis context. The specifications are given in terms of the passband and stopband edge frequencies $\omega_\mathrm{p}, \omega_\mathrm{s}$, the amplitude δ_p of the ripples in the passband and the attenuated amplitude δ_s in the stopband. There are no symmetry requirements on these parameters, as it was the case e.g. with Nyquist filters, and empirical formulae for computing the minimum order D matching a set of specifications are known. An extensive discussion of various methods for interpolation FIR design and implementation can be found in [4].

Rather than using conventional design parameters, we exploit here the ability to compensate magnitude errors in the passband region by changing the coefficients of the resonators. In other words, we can set a very high passband tolerance in order to achieve better stopband attenuation with relative low filter lengths. We used the MATLAB Signal Processing Toolbox procedure `firpm` to design interpolation FIR filters with the Parks-McClellan algorithm. It is possible to specify in this program the relative weight of the passband and stopband error, used for the error evaluation in the iterative procedure. We found out that even with a relative weight of 1:1000 (passband:stopband) it is possible to compensate the error in the passband without degrading the behaviour in the transition band. A filter design example is shown in Fig. 5.9 for the case of $D = 16$, $\omega_\mathrm{p} = 0.45\pi$, $\omega_\mathrm{s} = \pi - \omega_\mathrm{p}$ and

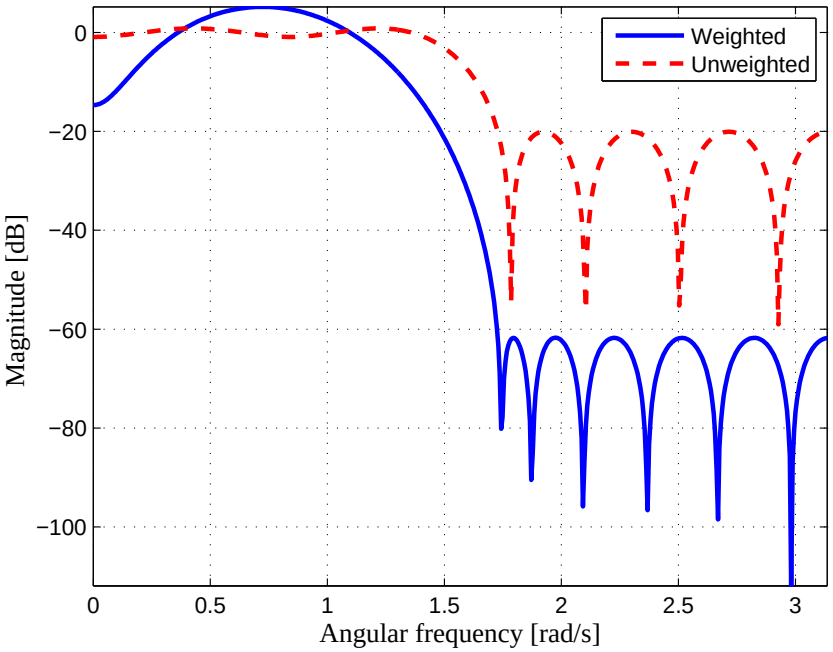

Fig. 5.9. Frequency response of a linear phase FIR filter of order=16 used for one interpolation stage (solid blue line). Passband edge frequency is set to $\omega_{\mathrm{p}} = 0.45\pi$. The result of the conventional unweighted design is also shown (dashed red line). The large error in the passband can be compensated by weighting the coefficients of the second-order resonators.

compared to unweighted filter design with the same specifications. As it can be seen in the picture, the weighted specifications permit to achieve an improvement of approximately more than 40dB in the stopband attenuation.

When using passband-error compensation, it is hard to judge the quality of the interpolation procedure by only looking at the magnitude response of the FIR filter. In fact, by e.g. applying a weight to a component in the passband, we are automatically using the same scale factor for its aliased component in the stopband. Within this synthesis context, it is better to express the accuracy in terms of *attenuation of the aliased component* depending on the resonance frequency $\omega_{\mathrm{r,m}}$ and on the interpolating filters used in the various upsampling stages.

The amplitude $E_m(\omega)$ of the highest aliasing component for a frequency ω in the m-th stage can be computed in a recursive fashion, evaluating the magnitude response of the filter at both the resonance and aliased frequency. We denote with $\mathbf{q}$ a m-length vector whose elements are 0 or 1 depending if the current QMF stage is respectively lowpass or highpass. The order is taken from the deepest band; taking as example the sequence "lowpass, highpass, lowpass" we have $\mathbf{q} = [0, 1, 0]$, corresponding to the subband $s'_{3,4}$ in Fig. 5.8. Given a frequency ω_m inside the

m-th subband, the aliasing error $E_m(\omega_m)$ can be computed with the following algorithm:

$\omega \leftarrow \omega_m$
$\tilde{\omega} \leftarrow \pi - \omega$
$E_m \leftarrow 1$
for $k \leftarrow m$ **to** 1 **do**
  if $\mathbf{q}(k)$ **then**
    $\tilde{\omega} \leftarrow \frac{\pi}{2} - \frac{\tilde{\omega}}{2}$
    $\omega \leftarrow \pi - \frac{\omega}{2}$
  else
    $\tilde{\omega} \leftarrow \frac{\tilde{\omega}}{2}$
    $\omega \leftarrow \frac{\omega}{2}$
  end
  $E_m \leftarrow E_m \cdot \frac{H_{k,q(k)}(e^{j\omega})}{H_{k,q(k)}(e^{j\tilde{\omega}})}$
end

Algorithm 1: Computing the aliasing error function $E_m(\omega_m)$

The algorithm keeps track of the aliased frequency $\tilde{\omega}$ and update the complex-valued error function E_m according to the ratio between the actual aliased error and the compensation factor. It makes uses of the frequency transfer function of each stage $H_{k,q(k)}$ where e.g. $H_{2,0}$ is the lowpass interpolator for the second stage. The aliasing attenuation can then be computed as the inverse of the absolute value of the error function $1/|E_m|$. In Fig. 5.10 the attenuation with respect to frequency is plotted for the different stages (1..3) of the synthesis bank. The three figures are related to different lengths D_m of the interpolation filters h_m used for each stage, which determine a compromise between latency and accuracy in the system. The total latency for M stages, expressed in full-rate samples is

$$\text{latency} \;=\; \sum_{m=1}^{M} D_m \, 2^{m-1}, \tag{5.7}$$

i.e. simply the sum of the filter orders weighted by the downsampling factor. It is therefore more convenient to use shorter filter lengths for the lowest sampling rates, also because in this case the same component will be attenuated by multiple interpolators.

From the computed aliasing attenuation functions, we can finally derive the deadband edges for the multirate system with compensation, for example by imposing a -60dB threshold attenuation. Due to the ripples in the functions, it might be useful to apply a moving average filter to the magnitude of the aliasing function before thresholding, in order to minimize the influence of local minima. The resulting widths of the deadband, expressed as percentage of total available bandwidth, are reported in Tab. 5.1 for the cases corresponding to the plots of Fig. 5.10. We also computed an average computational speed-up for each latency value, assuming a constant distribution of resonators along the frequency spectrum and neglecting the cost of interpolation filters.

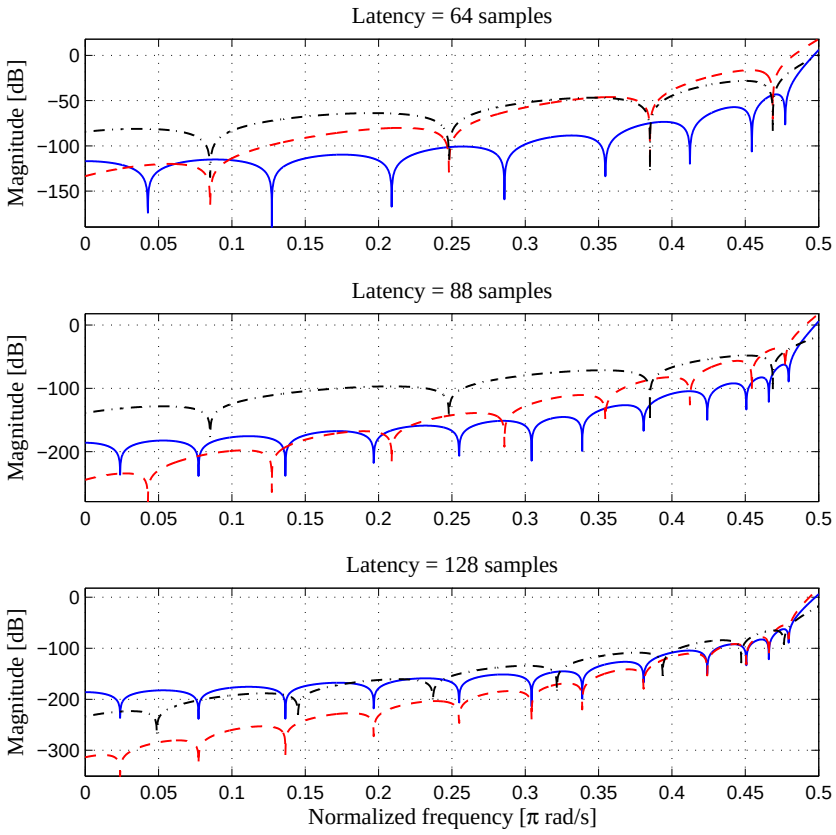

Fig. 5.10. Attenuation of the highest alias replica versus resonance frequency. The three lines are related to different stages in the synthesis bank: $m = 3$ (solid blue line), $m = 2$ (dashed red line), $m = 1$ (dash-dotted black line). The figure is replicated for different values of interpolation filter orders, leading to different total latencies (see. Tab. 5.1).

Total Latency	**Filter Orders**	**-60dB dead bandwidth**	**Average Speed-up**
64	(8, 8, 16)	(10%, 42%, 44%)	1.47
88	(12, 12, 16)	(5%, 9%, 12%)	3.7
128	(16, 16, 32)	(5%, 5%, 3%)	5.67

Table 5.1. Accuracy versus total latency for a 3-stage QMF tree design. Filter orders and dead bandwidth (expressed as percentage of total available bandwidth) are ordered from lower to higher sampling rates. Average speed-up is computed assuming a flat frequency distribution of the resonators.

It is clear to see from this table how there is a compromise between latency, accuracy and speed-up. If for example we would like to achieve better quality, we should set a higher threshold for dead bandwidth computation, e.g. -90dB, thus

resulting in larger deadbands and, consequently, a higher computational cost since we can put few resonators in the lowest bands. Nevertheless, some compromises are better than others: the 64 samples latency case would not probably be worth to implement in many situations, but this does not mean that such lower latencies cannot be obtained in a multirate system. For example, reducing the number of stages significantly reduces latency, obviously with an increased computational cost. Finally, part of the accuracy comes from the Parks-McClellan filter design parameters for each stage; however, these have to be set by trial-and-error considering all the parameters, and the results are then evaluated with the aliasing attenuation functions. It would certainly help to investigate the use of optimization techniques to automate this procedure, possibly resulting in an optimal filter design strategy.

5.3 Handling Nonlinear Feedback : the Adaptive Multirate Approach

So far, we have focused only on the synthesis bank of the multirate system, without saying anything on how the resonators are excited. We have already noticed that conventional analysis/synthesis stages are mostly intractable with nonlinear feedback excitations, due to the phase delay of the interpolation filters.

The solution we propose here exploits the peculiar behaviour of the excitation forces in our case. The average time duration of collision forces (1 to 10 ms) is in fact generally very short compared to total simulation time. Therefore, when the average number of contacts per second is not very large (e.g. max 50 contacts per second), for most of the time the resonators are running in free evolution, i.e. without any excitation signal.

With an adaptive approach, we can then use the full sampling rate during the contact phase, thus avoiding the phase-delay multirate problems, and then switch to a multirate structure for the free evolution of the system. With the above mentioned assumptions on the time distribution between forced and steady state, this method could lead to an amortized computational cost that converges to the multirate cost in the presence of sparse impacts. Moreover, there is no need to implement the analysis bank step in the multirate structure, since all the input are done at full sampling rate.

The main problem we need to consider is how to switch from the full sampling rate of the resonators to their versions at lower sampling rates, and viceversa when the object is restruck. This "switch" have to be performed at run-time and possibly instantaneously in order to avoid amplitude jumps in the output. Following the same scheme used for the derivation of QMF trees, we will develop first the formulae for the switch to and from half the sampling rate, and then see how these can be combined to the generic 2^m downsampling factor.

The starting point is the state-space form of the all-pole resonator given in Eq. (3.11, 3.13). We consider the system in free evolution (i.e., $u(n) = 0$), which is described only by its state variable $\mathbf{x}(n)$. The corresponding system with half the sampling rate is described by its evolution equation:

$$\mathbf{x}_{\downarrow 2}(n) = \mathbf{A}_2 \mathbf{x}_{\downarrow 2}(n) + \mathbf{B}_2 u_{\downarrow 2}(n), \tag{5.8}$$

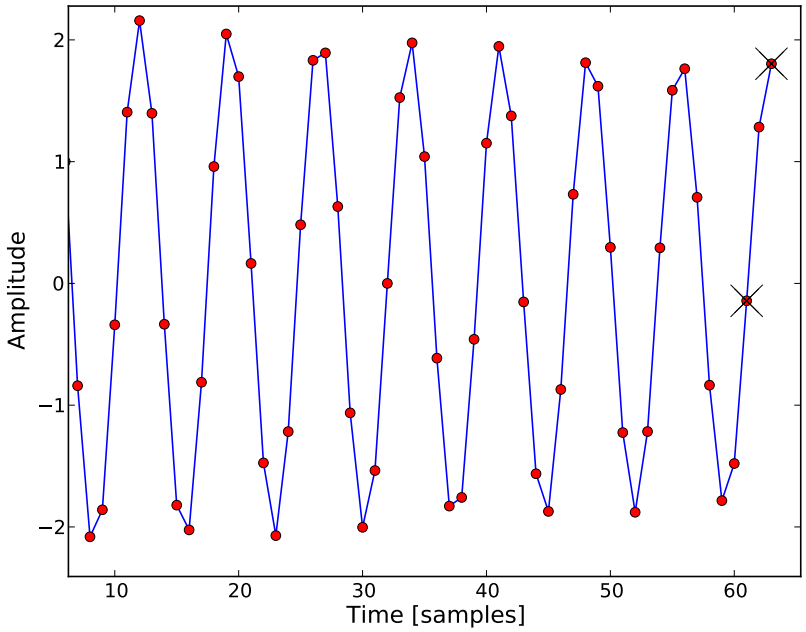

Fig. 5.11. Downsampling the state variables of a second-order resonator. The new states running at half the sampling rate are shown with the black cross markers.

where the downsampled state $\mathbf{x}_{\downarrow 2}(n)$ is defined in terms of the original system variables as:

$$\mathbf{x}_{\downarrow 2}(n) = \begin{bmatrix} y(n) \\ y(n-2) \end{bmatrix}. \tag{5.9}$$

In other words, downsampling the states correspond to shift the second state variable by one sample (see Fig. 5.11). Expanding the expression of the full-rate state Eq. (3.13) for the time instant $n-2$ gives:

$$\boldsymbol{x}(n-2) = \boldsymbol{A}^{-2}\boldsymbol{x}(n) - \boldsymbol{A}^{-1}\boldsymbol{B}u(n) - \boldsymbol{A}^{-2}\boldsymbol{B}u(n-1). \tag{5.10}$$

Since we are assuming a null-input signal, the transformation between the full-rate state $\mathbf{x}$ and its downsampled version $\mathbf{x}_{\downarrow 2}$ is thus simply given by the inverse of the square of the state matrix $\boldsymbol{A}$:

$$\begin{aligned} \mathbf{x}_{\downarrow 2}(n) &= \mathbf{S}_2\ \mathbf{x}(n) \\ &= \begin{bmatrix} 0 & 1 \\ \frac{a_1}{a_2^2} & \frac{a_1^2 - a_2}{a_2^2} \end{bmatrix}, \end{aligned} \tag{5.11}$$

where we have renamed such transformation matrix as $\mathbf{S}_2$ for convenience and a_1, a_2 are the elements of $\mathbf{A}$, i.e. the state coefficients of the resonator. Notice that the null-input signal condition is not necessary to derive a transformation of the

matrix, since all we have to do is solve the equation defined by 5.10, although the expression becomes more complicated and has no use in our context.

Thanks to the state-space formulation of the state downsampling operation, it is easy to formulate expressions for higher sampling ratios 2^m by multiplying together m versions of $\mathbf{S}_2$. However, this is not completely correct, since the coefficients a_1, a_2 in Eq. (5.11) are related to the full sampling rate, while we should refer to lower rates in successive applications of the transformation matrix. Using the formulae for the resonators coefficients given in Eq. (3.5), we can relate the coefficients for a resonator with the same parameters (resonance frequency, decay time) at different sampling rates:

$$a_{1,\downarrow 2} = 2a_2^2 - a_1^2 \tag{5.12}$$

$$a_{2,\downarrow 2} = a_2^2, \tag{5.13}$$

which are obviously valid only if the resonance frequency ω_r satisfies the Nyquist criterion for both sampling rates. With this equation, we can now express a recursive algorithm for the computation of the state resampling matrix $\mathbf{S}_{2^m}$, which can e.g. used for successive stages of the QMF tree.

$\mathbf{S}_{2^m} \leftarrow \mathbb{I}$
for $k \leftarrow 1$ **to** m **do**
  $\mathbf{S}_{2^m} \leftarrow \begin{bmatrix} 0 & 1 \\ \frac{a_1}{a_2^2} & \frac{a_1^2 - a_2}{a_2^2} \end{bmatrix} \mathbf{S}_{2^m}$
  $a_1 \leftarrow 2a_2^2 - a_1^2$
  $a_2 \leftarrow a_2^2$
end

Algorithm 2: Recursive computation of the state resampling matrix $\mathbf{S}_{2^m}$

Once we know the QMF stage m for each resonator, the matrices $\mathbf{S}_{2^m}$ can then precomputed off-line for all the modes. Due to their structures, only two floating point values per matrix are needed. In order to instantaneously switch the states from the full sampling rate to the lower rates, all we have to do is just multiply the state vector of each resonator with the corresponding matrix, which again takes only two MACs per mode, i.e. less than the update of a single sample. Moreover, the weighting coming from interpolation filter compensation can be pre-applied to the state resampling matrices, and also the phase offset compensation and QMF mirror frequency inversion thanks to the transform to quadrature phase components see in Chapter 3.

The state resampling matrices are also always non-singular, meaning that switching back to the full sampling rate just implies a multiplication with the inverse of the matrix:

$$\mathbf{x}(n) = \mathbf{S}_{2^m}^{-1} \mathbf{x}_{\downarrow 2^m}(n), \tag{5.14}$$

where again the inverse matrices $\mathbf{S}_{2^m}^{-1}$ can be precomputed and cheaply applied at run-time. An example of restriking a resonating object with the multirate approach is shown in Fig. 5.12.

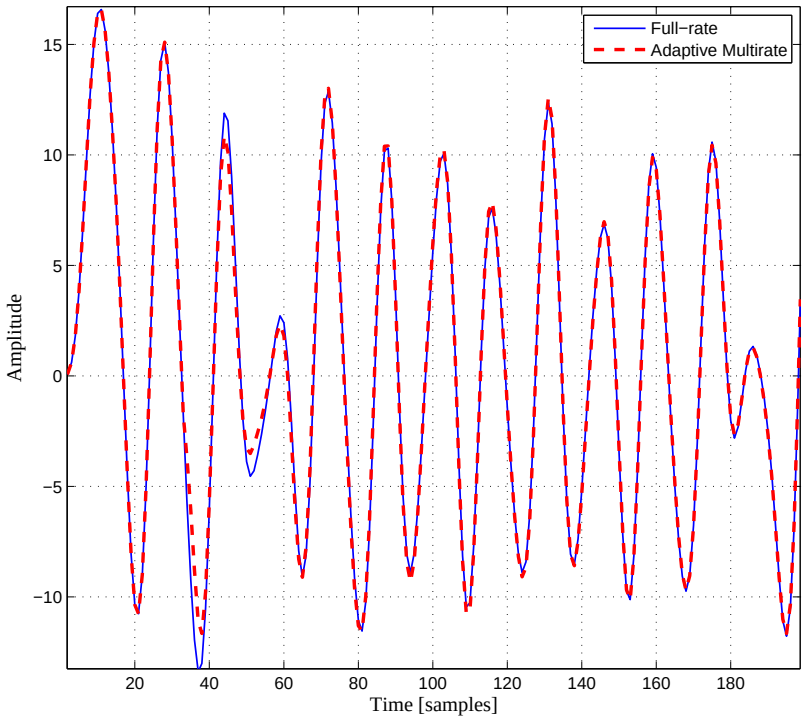

Fig. 5.12. Adaptive multirate restrike of a modal resonator (dashed red line), compared to the output that uses the full sampling rate for the whole simulation (solid blue line). Some smalll artifacts shortly after the switch from full-rate to multirate are noticeable, due to interpolation filters transients.

When multiple objects are implemented sharing the same multirate synthesis bank, care has to be taken when switching a resonator to and from the multirate implementation. In this case, the interpolation filters are continuously running and therefore some audible artifacts will be present due to the memory of FIR filters, and so some workarounds like e.g. applying an envelope to a copy of the resonator signal during the switch has to be employed.

5.4 Conclusion

In this Chapter two methods were reviewed which can be used to obtain computational efficient algorithms for modal synthesis, based respectively on energy-based resonator pruning at run-time and adaptive multirate structures.

The multirate method proposed here has been developed with the MATLAB programming language, and considering only the case of a single resonating object. Future work will require the development of a structured algorithm for the multi-

object case, possibly in a low level programming language in order to test the combined speed-up of multirate and parallel implementations of the resonator bank. Another interesting topic of research is the analysis of optimal filter design methods in this context when compensation weights are applied to the resonators.

Part II

Appendix

A

Example Software

The investigation of some techniques presented in this thesis, and in particular the definition of the requirements of a FEM based real-time audio engine, would not have been possible without proper validation by software prototypes. In this appendix, some of the applications developed are discussed briefly, focusing on the general software architecture and on the choice of external tools/programming libraries. Implementation details are omitted, with the exception of the last paragraph which deals with code optimizations that can help the efficiency of modal synthesis on current general purpose CPUs.

A.1 Modal Objects Data Structures

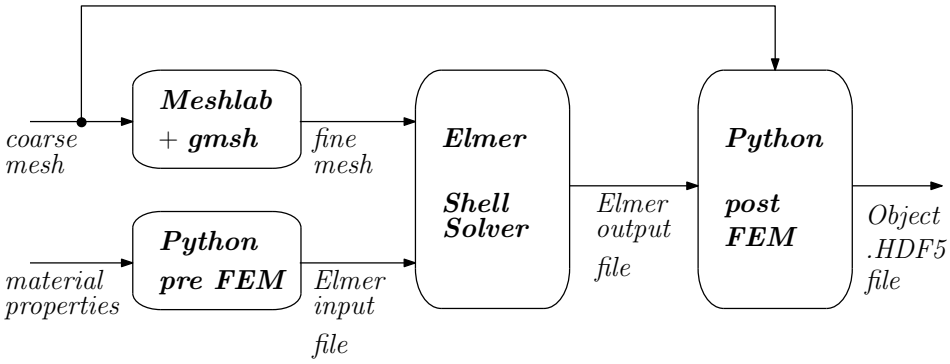

Fig. A.1. Tools for generating the .HDF5 file containing the data for the modal description of a resonator starting from a geometric mesh and material properties.

The proposed framework for modal synthesis based on Finite Element Methods is strongly based on a precomputation stage for the parameters of the resonators (see Chapter 2). We used the *Elmer* open-source toolkit [65] for FEM simulations, and developed a partially automated procedure for the generation of data struc-

tures suitable for real-time audio rendering, which are organized and stored using the Hierarchical Data Format (HDF5, [45]).

The pipeline for the creation of such files is shown in Fig. A.1, starting from a generic geometric mesh and the object material properties, which are density, Young's modulus, Poisson's ratio and shell thickness. Open-source programs *Meshlab* [51] and *gmsh* [51] were used for the manipulation and refinement of triangular meshes. A simple *Python* [128] script drives the initialization of *Elmer*'s solver, by generating a proper input file and running the sequence of necessary command-line operations. Among the many solver algorithms present in *Elmer*, we chose the ad-hoc version for shell elastostatic problems, using first-order triangular elements. An example of a generated solver input file (`.sif`) is shown in Tab. A.1.

The output file generated by Elmer is then processed by a more complex Python post-processing script, making strong use of the *Scipy* [32] framework, which is a MATLAB-like scientific computation environment. In this script, the raw three-dimensional modal shapes are extracted from the FEM output results and the modal frequencies computed. In a subsequent phase, both the meshes (coarse and fine version) are loaded from their file and converted to an ad-hoc structure, and the scalar modal weights are calculated using the approach described in Sec. 2.2, 2.5.

The hierarchical binary format of HDF files makes possible to store efficiently both the raw fine mesh data structures and their optimized equivalents on the coarse mesh. Storing all the information results in large file dimensions (usually 10-50 MegaBytes), but it is possible to keep only the reduced data for real-time simulations (approximately 50-500 KiloBytes, depending mostly on coarse mesh size).

A.2 Sound Object Explorer

When dealing with large amounts of data such as those generated by Finite Element Analysis, visualization tools become very important for the development and validation of simulation techniques. For this reason, a GUI application - named *Sound Object Explorer* has been developed for the analysis of resonating objects stored with the custom `.HDF5` format previously described. The application is written in C++ and is based on the graphical toolkit *Qt* [86] with the use of *OpenGL* [90] for 3D graphics rendering and *Qwt* [102] for scientific data plotting.

The application can be used to visualize the normal modes of an object (Fig. A.2), e.g. by interactively visualization and animation of the modal shapes or plotting the averaged pressure and displacement weights for a set of elements in the mesh. Damping properties can be controlled using the parameters discussed in Sec. 2.2 and the resulting decay time distribution can be plotted. The program also integrates a fast engine for modal synthesis with lumped impact excitations, which can also be visualized at run-time (Fig. A.3). The output signals (computed sound pressure and displacement at contact point) can be written to an audio `.wav` file or in text format for analysis with different tools.

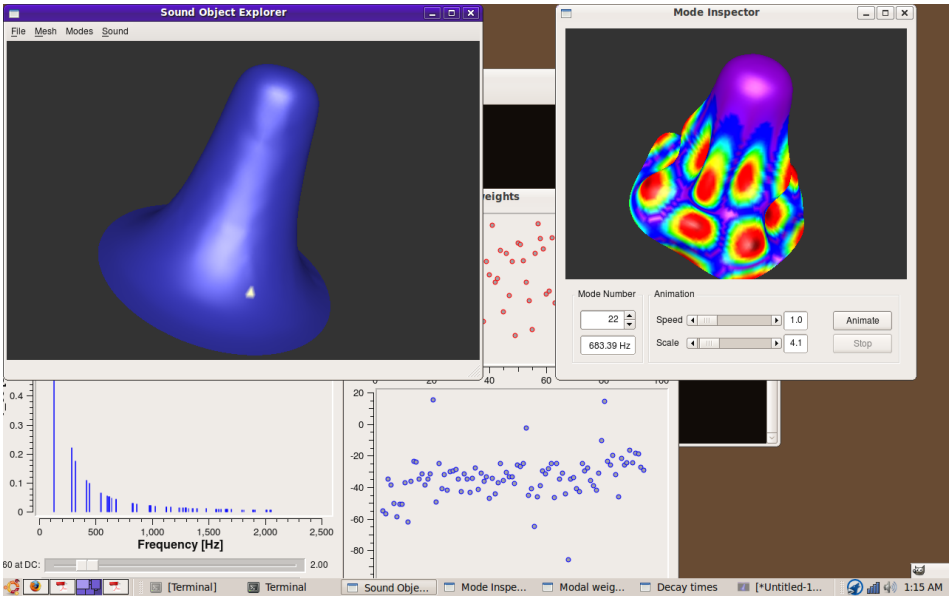

Fig. A.2. Some analysis windows of *Sound Object Explorer*. In clock-wise order starting from top-left, we have mesh/element selection, modal shapes visualization, modal weights plot and decay times distribution.

A.3 Interactive Environment

A simple real-time multimodal environment has also been developed, mostly during a visit at the Shared Reality Lab at McGill University (Montreal, Canada). The context of the simulation is a "virtual floor" where sounding objects can be kicked by the user in real-time. The environment makes use of apposite haptic devices developed for the EU-funded project Natural Walking Environments [130], such as a matrix of tiles augmented with pressure sensors (Fig. A.4). Haptic feedback is provided by shakers positioned under each tile and real-time graphics can be projected on the floor. The environment is controlled by a network of personal computers, one for each row of six floor tiles, and sensing data are automatically sent in broadcast inside the network using the OpensSound Control (*OSC*) communication protocol [131].

The overall architecture of the system is depicted in Fig. A.5. User interactions are detected using a motion-tracking system to analyze body and feet motion, and by four pressure sensors under the corners of each floor tile. The rigid body simulation engine is written in C++ using the *Bullet* physics simulation engine [106] and basic graphic rendering is provided using the template OpenGL engine provided with the library. All the communication between control-rate blocks of the systems is handled by proper OSC messages.

Sound synthesis is performed using the same software developed for the *Sound Object Explorer* application, with optimizations for running multiple objects using a memory pool. The interface between the audio engine and the *Bullet* simulator is provided by a structure describing a single impact, i.e. by pointers to a pair of

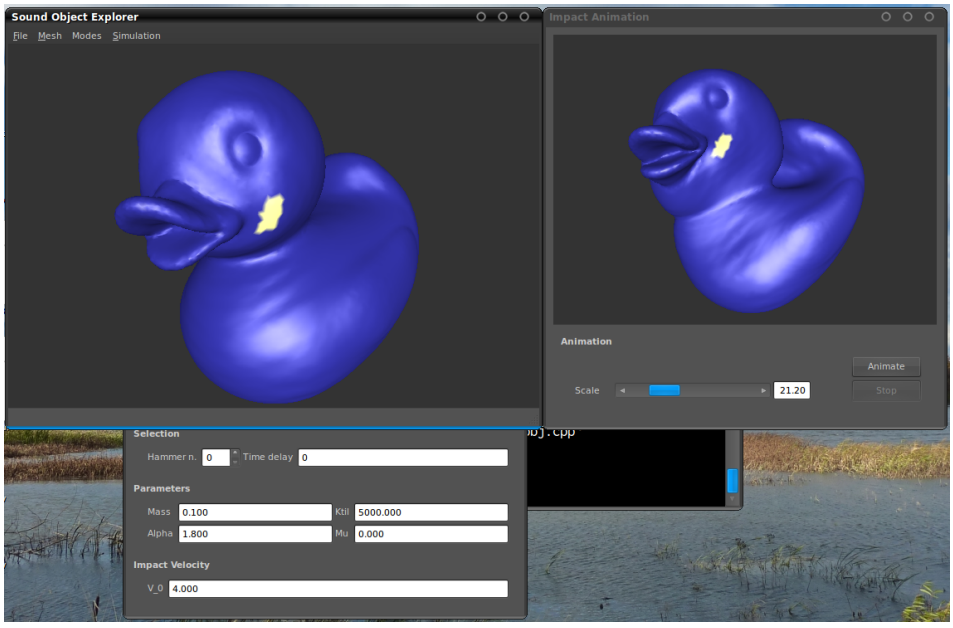

Fig. A.3. Simulation example in *Sound Object Explorer*. The contact surface is selected on the mesh (yellow area in the top-left window) and nonlinear contact parameter can be set with the top-left dialog. A slowed-down animation of the contact can be then visualized at run-time (right window).

objects, indices of contact elements on the two meshes and initial contact velocity. User's feet are modeled inside the engine by using two rectangular rigid (i.e., non-sounding) boxes, which serve to initialize the motion of the other objects. The main difficulty in the implementation resides in getting appropriate impact data by querying Bullet engine's state with the provided API. For this purpose, the `Dispatcher` class and its method `getManifoldByIndexInternal()` are used to query the manifolds (i.e., objects) which are in contact at each control-rate timestep. We applied a threshold on the minimum impact velocity and on the minimum time between impacts between the same pair of objects, because otherwise many "spurious" contacts can be falsely detected by the library, e.g. for non-moving objects.

Haptic feedback relies on a metal plate simulation, which is done using modal synthesis based on spectral methods, using the code previously developed for a plate reverb simulation [133]. The purpose is to simulate feedback from a stiff floor by making all the tiles vibrate accordingly to the position and force of the walking gestures. Parallel processing is used, since each row of tiles is controlled separately by a dedicated computer. However, we chose to simply replicate the same version of the code for each PC: in this way, some computational power is wasted because every processor performs the update of the same bank of resonators, but the implementation is much simpler and no synchronization between the processes is needed. However, the six running programs differ for the output weights vectors, in order to compute the vibration for the actuators that they

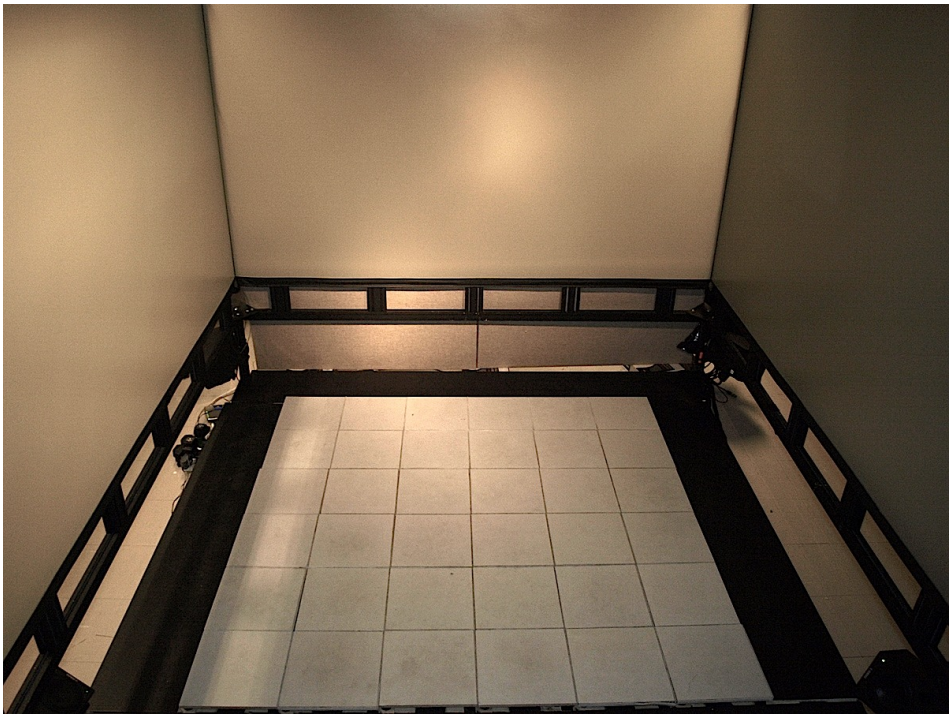

Fig. A.4. The Cave simulation floor environment at McGill University TODO:nome lab!. The floor is composed with a 6x6 matrix of tiles having pressure sensors and haptic actuators.

control. The input of the modal plate engine is given by a structural similar to the one used for impacts between objects.

A.4 Parallel Implementation of the Resonator Bank

An C++ library for the synthesis of a parallel bank of second-order resonators has been developed with specific optimizations targeted to generic purpose x86 CPUs. Exploiting the intrinsic parallel nature of the algorithm, we applied parallelism using the floating point SSE instructions available on modern processors [50], which permit Single Instruction Multiple Data (SIMD) computing. In this way, it is possible to update four (or eight with the newest AVX instructions) resonators with full vector computation, i.e. the speed-up is linear (4x) compared to the scalar case.

Using this kind of computation constructs, there are some constraints on the way the coefficients of the resonator are stored in memory, e.g. only contiguous memory areas can be used to store a vector of coefficients. When cycling over the resonators, two possible ways of organizing the data are generally used, known as Array of Structures (AoS) and Structure of Arrays (SoA). Although AoS is generally considered faster because of better memory and cache management [42],

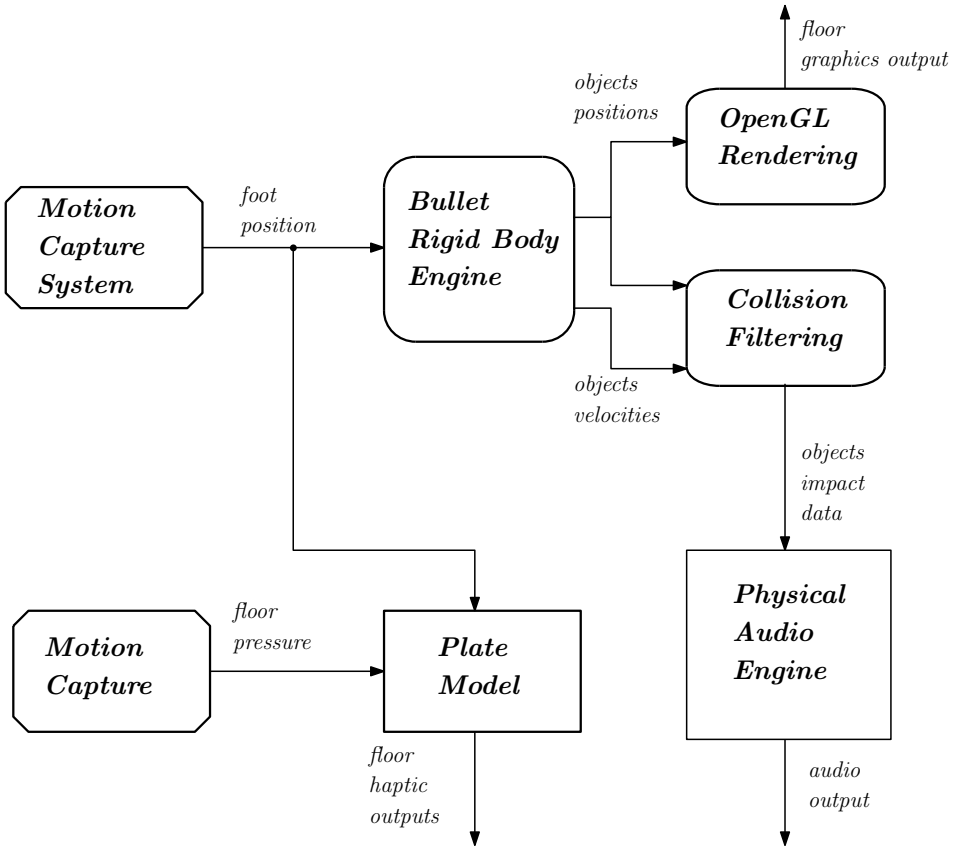

Fig. A.5. Architecture of the Real-Time simulation environment with multimodal (audio, graphics and haptics) rendering. Audio-rate blocks are drawed as rectangular boxes, where instead rounded corners have been used for control-rate blocks directly related to the Bullet engine.

when dealing with SSE instructions SoA implementations are generally used due to the contiguity constraint [50]. In our case, and only when we have a fixed number of weights used, e.g. 2 for one pressure output plus displacement, better performance may be achieved by using an Array of Vector Structures. Referring to the example code reported in Tab. A.2, this implies the use of vector fields packed into a structure, where the size of the vectors equal the one of the SIMD computing unit. On a Core2Duo@2.4Ghz processor with compiler gcc++4.6, we benchmarked an average 10% speed increment over straightforward parallel SoA implementations.

Per-processor parallelism has also been implemented using the *OpenMP* library [37], dividing the computational load among different threads when many sounding objects (i.e., more than one bank of resonators) are simulated. Since all the feedback is local to each thread, synchronization can be ensured at buffer rate when the outputs of the different objects are summed together.

```
Header
  CHECK KEYWORDS Warn
  Mesh DB "." "."
End

Constants
  Gravity(4) = 0 -1 0 9.82
  Stefan Boltzmann = 5.67e-08
  Permittivity of Vacuum = 8.8542e-12
  Boltzmann Constant = 1.3807e-23
  Unit Charge = 1.602e-19
End

Simulation
  Max Output Level = 3
  Coordinate System = Cartesian 3D
  Simulation Type = Steady State
  Steady State Max Iterations = 1
  Output Intervals = 1
  Post File = "shell.ep"
! Output File = "shell.dat"
End

Body 1
  Equation = 1
  Material = 1
  Body Force = 1
End

Material 1
   Density = 8000
   Thickness = 0.001
   Youngs Modulus = 200e9
   Poisson Ratio = 0.3
End

Body Force 1
  Normal Pressure = Real 0
End

Solver 1
  Equation = Shell Solver
  Procedure = "ShellSolve" "ShellSolver"
  Variable = -dofs 6 Deflection
  Linear System Solver = Direct
  Linear System Direct Method = UMFpack
  Steady State Convergence Tolerance = 1.0e-6
  Eigen System Select = Smallest magnitude
  Eigen Analysis = True
  Eigen System Values = 200
  Calculate Weights = Logical True
End

Equation 1
  Active Solvers(1) = 1
End
```

Table A.1. Format of the Elmer input .sif file used for Finite Element Analysis computation.

```
////////////////////////////////////////////////
// Array of Structures
////////////////////////////////////////////////
struct FilterData {
    float b1;
    float a1;
    float a2;
    float z1;
    float z2;
    float w_in;
    float w_out_x;
    float w_out_p;
};

class ResonatorBank {
// ...
// ...
    FilterData _resonators_data[N_RESONATORS];
};

////////////////////////////////////////////////
// Structure of Arrays
////////////////////////////////////////////////
class ResonatorBank {
// ...
// ...
    float _b1[N_RESONATORS];
    float _a1[N_RESONATORS];
    float _a2[N_RESONATORS];
    float _z1[N_RESONATORS];
    float _z2[N_RESONATORS];
    float _w_in[N_RESONATORS];
    float _w_out_x[N_RESONATORS];
    float _w_out_p[N_RESONATORS];
};

////////////////////////////////////////////////
// Array of Vector Structures
////////////////////////////////////////////////
struct FilterChunkData {
    float b1[SIMD_DATA_SIZE];
    float a1[SIMD_DATA_SIZE];
    float a2[SIMD_DATA_SIZE];
    float z1[SIMD_DATA_SIZE];
    float z2[SIMD_DATA_SIZE];
    float w_in[SIMD_DATA_SIZE];
    float w_out_x[SIMD_DATA_SIZE];
    float w_out_p[SIMD_DATA_SIZE];
};

class ResonatorBank {
// ...
// ...
    FilterChunkData resonators_data[N_RESONATORS / SIMD_DATA_SIZE];
};
```

Table A.2. C++ code excerpts for data structures regarding the implementation of a parallel resonator bank using Array of Structures (AoS), Structure of Arrays (SoA) and Array of Vector Structures (AoVS).

References

1. S. Adhikari. *Damping Models for Structural Vibration*. PhD thesis, Cambridge University, Engineering Department, 2000.
2. S. Adhikari. Damping modelling using generalized proportional damping. *Journal of Sound and Vibration*, 293(1-2):156–170, 2006.
3. Jean-Marie Adrien. *The missing link: modal synthesis*, pages 269–298. MIT Press, Cambridge, MA, USA, 1991.
4. Arian, P. *Computationally Efficient Decimators, Interpolators, and Narrow Transition-Band Linear-Phase Finite Impulse Response (FIR) Filters*. PhD thesis, Tamper University of Technology, Finland, 2007.
5. F. Avanzini and P. Crosato. Integrating physically based sound models in a multimodal rendering architecture. *Computer Animation and Virtual Worlds*, 17(3-4):411, 2006.
6. F. Avanzini and R. Marogna. A modular physically based approach to the sound synthesis of membrane percussion instruments. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(4):891–902, 2010.
7. F. Avanzini and D. Rocchesso. Modeling Collision Sounds: Non-linear Contact Force. In *Proc. COST-G6 Conf. Digital Audio Effects (DAFX-01), Limerick, Ireland*, pages 61–66, 2001.
8. F. Avanzini, S. Serafin, and D. Rocchesso. Interactive Simulation of Rigid Body Interaction With Friction-Induced Sound Generation. *Speech and Audio Processing, IEEE Transactions on*, 13(5 Part 2):1073–1081, 2005.
9. B. Bank. *Physics-based Sound Synthesis of String Instruments Including Geometric Nonlinearities*. PhD thesis, Budapest University of Technology and Economics, Hungary, February 2006. URL: `http://www.mit.bme.hu/~bank/phd`.
10. B. Bank. Direct design of parallel second-order filters for instrument body modeling. In *Proc. International Computer Music Conference (ICMC 2007), Copenhagen, Denmark*, pages 458–465, 2007.
11. B. Bank. Perceptually motivated audio equalization using fixed-pole parallel second-order filters. *Signal Processing Letters, IEEE*, 15:477–480, 2008.
12. B. Bank. Residual Energy Estimation for Polyphony Management. Internal Technical Report, Viscount International S.p.A., 2010.
13. B. Bank and L. Sujbert. Generation of longitudinal vibrations in piano strings: From physics to sound synthesis. *The Journal of the Acoustical Society of America*, 117:2268, 2005.
14. B. Bank, S. Zambon, and F. Fontana. A modal-based real-time piano synthesizer. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(4):809–821, 2010.

15. J. Bensoam. A reciprocal variational approach to the two–body frictionless contact problem in elastodynamics. *International Journal for numerical methods in Engineering*, 2002.
16. J. Bensoam. Contact problems with friction applied to musical sound synthesis. *The Journal of the Acoustical Society of America*, 119:3323, 2006.
17. S. Bilbao. *Wave and scattering methods for numerical simulation*. John Wiley and Sons, 2004.
18. S. Bilbao. *Numerical Sound Synthesis*. Wiley Online Library, 2009.
19. A. Boeing and T. Bräunl. Evaluation of real-time physics simulation systems. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, pages 281–288. ACM New York, NY, USA, 2007.
20. A. Boeing and T. Bräunl. Evaluation of real-time physics simulation systems. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, pages 281–288. ACM, 2007.
21. N. Bonneel, G. Drettakis, N. Tsingos, I. Viaud-Delmon, D. James, et al. Fast modal sounds with scalable frequency-domain synthesis. *ACM Transactions on Graphics*, 27:3, 2008.
22. G. Borin. Personal Communication, 2007.
23. G. Borin, G. De Poli, and D. Rocchesso. Elimination of delay-free loops in discrete-time models ofnonlinear acoustic systems. *Speech and Audio Processing, IEEE Transactions on*, 8(5):597–605, 2000.
24. G. Borin, G. De Poli, and A. Sarti. Algorithms and Structures for Synthesis Using Physical Models. *Computer Music Journal*, 16:30–30, 1992.
25. J.P. Boyd. *Chebyshev and Fourier spectral methods*. Dover Publications, 2001.
26. Cynthia Bruyns. Modal synthesis for arbitrarily shaped objects. *Comput. Music J.*, 30(3):22–37, 2006.
27. C. Cadoz, A. Luciani, and J.L. Florens. CORDIS-ANIMA: A Modeling and Simulation System for Sound and Image Synthesis-The General Formalism. *Computer Music Journal*, 17:19–19, 1993.
28. T. K. Caughey and M. E. J. O 'Kelly. Classical normal modes in damped linear dynamic systems. *Transaction of American Society of Mechanical Engineers, Journal of Applied Mechanics*, 32:583–588, 1965.
29. J.N. Chadwick, S.S. An, and D.L. James. Harmonic shells: a practical nonlinear sound model for near-rigid thin shells. In *ACM Transactions on Graphics (TOG)*, volume 28, page 119. ACM, 2009.
30. P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. Meshlab: an open-source mesh processing tool. In *Sixth Eurographics Italian Chapter Conference*, pages 129–136, 2008.
31. R.D. Ciskowski and CA Brebbia. *Boundary Element Methods in Acoustics*. Elsevier Applied Science, 1991.
32. Scipy Developer Community. Scipy - Scientific Tools for Python. `http://www.scipy.org/`.
33. P.R. Cook. Physically informed sonic modeling (phism): Synthesis of percussive sounds. *Computer Music Journal*, 21(3):38–49, 1997.
34. P.R. Cook. *Music, Cognition, and Computerized Sound: An Introduction to Psychoacoustics*. MIT Press, 2001.
35. P.R. Cook. *Real Sound Synthesis for Interactive Applications*. AK Peters, Ltd., 2002.
36. Roland Corporation. V-Piano. http://www.roland.com/V-Piano/.
37. L. Dagum and R. Menon. Openmp: an industry standard api for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55, 1998.

38. M. De Berg, O. Cheong, and M. Van Kreveld. *Computational geometry: algorithms and applications*. Springer-Verlag New York Inc, 2008.
39. G. De Poli, A. Piccialli, and C. Roads. *Representations of musical signals*. MIT press, 1991.
40. G. De Poli and D. Rocchesso. Physically based sound modelling. *Organised Sound*, 3(1):61–76, 1998.
41. K. Doel, D. Knott, and D.K. Pai. Interactive Simulation of Complex Audiovisual Scenes. *Presence: Teleoperators & Virtual Environments*, 13(1):99–111, 2004.
42. K. Dowd. *High performance computing*. O'Reilly & Associates, Inc. Sebastopol, CA, USA, 1993.
43. G. Eckel, F. Iovino, and R. Caussé. Sound synthesis by physical modelling with modalys. In *Proceedings of the International Symposium of Music Acoustics*, 1995.
44. L. Faiget, C. Legros, and R. Ruiz. Optimization of the impulse response length: Application to noisy and highly reverberant rooms. *Journal of the Audio Engineering Society*, 46:741–750, 1998.
45. M. Folk, A. Cheng, and K. Yates. Hdf5: A file format and i/o library for high performance computing applications. In *Proceedings of the Supercomputing Conference*, volume 99, 1999.
46. F. Fontana and F. Avanzini. Computation of delay-free nonlinear digital filter networks: Application to chaotic circuits and intracellular signal transduction. *Signal Processing, IEEE Transactions on*, 56(10):4703–4715, 2008.
47. Association for Computing Machinery. Acm computing classification system. `http://www.acm.org/class/`.
48. W.W. Gaver. How Do We Hear in the World? Explorations in Ecological Acoustics. *Ecological Psychology*, 5(4):285–313, 1993.
49. W.W. Gaver. What in the World Do We Hear?: An Ecological Approach to Auditory Event Perception. *Ecological Psychology*, 5(1):1–29, 1993.
50. G. Gerbert and A. Bik. *The Software Optimization Cookbook, 2nd edition*. Intel Press, 2006.
51. C. Geuzaine and J.F. Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.
52. S. Gibson and B. Mirtich. A survey of deformable modeling in computer graphics. Technical report, Mitsubishi Electric Research Laboratories, 1997.
53. G. Gilardi and I. Sharf. Literature survey of contact dynamics modelling. *Mechanism and Machine Theory*, 37(10):1213–1239, 2002.
54. N. Giordano. Mechanical impedance of a piano soundboard. *The Journal of the Acoustical Society of America*, 103:2128, 1998.
55. W. Goldsmith. *Impact: The Theory and Physical Behaviour of Colliding Solids*. Courier Dover Publications, 2001.
56. G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.
57. J.K. Hahn, H. Fouad, L. Gritz, and J.W. Lee. Integrating Sounds and Motions in Virtual Environments. *Presence*, 7(1):67–77, 1998.
58. J. He and Z.F. Fu. *Modal Analysis*. Butterworth-Heinemann, 2001.
59. T. Hermann and A. Hunt. Guest editors' introduction: An introduction to interactive sonification. *Multimedia, IEEE*, 12(2):20–24, 2005.
60. L. Hiller and P. Ruiz. Synthesizing musical sounds by solving the wave equation for vibrating objects. *Journal of the Audio Engineering Society*, 19(6):462–470, 1971.
61. W.D. Hillis and G.L. Steele Jr. Data parallel algorithms. *Communications of the ACM*, 29(12):1170–1183, 1986.

62. K. Ho-Le. Finite element mesh generation methods: a review and classification. *Computer-aided design*, 20(1):27–38, 1988.
63. KH Hunt and FRE Crossley. Coefficient of restitution interpreted as damping in vibroimpact. *ASME Journal of Applied Mechanics*, 42(2):440–445, 1975.
64. F. Ihlenburg. *Finite Element Analysis of Acoustic Scattering*. Springer, 1998.
65. University of Helsinki IT Center for Sciences. Elmer Open Source Finite Element Software. `http://www.csc.fi/english/pages/elmer`.
66. D.L. James, J. Barbič, and D.K. Pai. Precomputed acoustic transfer: output-sensitive, accurate sound generation for geometrically complex vibration sources. *Proceedings of ACM SIGGRAPH 2006*, 25(3):987–995, 2006.
67. D.L. James and D.K. Pai. DyRT: dynamic response textures for real time deformation simulation with graphics hardware. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 582–585. ACM New York, NY, USA, 2002.
68. D.L. James and D.K. Pai. Multiresolution green's function methods for interactive simulation of large-scale elastostatic objects. *ACM Transactions on Graphics (TOG)*, 22(1):47–82, 2003.
69. M. Karjalainen, P. Antsalo, A. Makivirta, T. Peltonen, and V. Valimaki. Estimation of modal decay parameters from noisy response measurements. *Journal of the Audio Engineering Society*, 50(11):867–878, 2002.
70. M. Karjalainen, P.A.A. Esquef, P. Antsalo, A. Makivirta, and V. Valimaki. Frequency-Zooming ARMA Modeling of Resonant and Reverberant Systems. *Journal of the Audio Engineering Society*, 50(12):1012–1029, 2002.
71. R.L. Klatzky, D.K. Pai, and E.P. Krotkov. Perception of Material from Contact Sounds. *Presence: Teleoperators & Virtual Environments*, 9(4):399–410, 2000.
72. Rich Lecoucqh. ARPACK numerical analysis library. Available online at `http://www.caam.rice.edu/software/ARPACK/`, 2012.
73. A.W. Leissa. *Vibration of shells*. American Institute of Physics, 1993.
74. Y. Li and G.R. Arce. A maximum likelihood approach to least absolute deviation regression. *EURASIP Journal on Applied Signal Processing*, 12(1762-1769):11, 2004.
75. M. Lin and S. Gottschalk. Collision detection between geometric models: A survey. In *Proc. of IMA Conference on Mathematics of Surfaces*, volume 1, pages 602–608, 1998.
76. D.B. Lloyd, N. Raghuvanshi, and N.K. Govindaraju. Sound synthesis for impact sounds in video games. In *Symposium on Interactive 3D Graphics and Games*, pages PAGE–7. ACM, 2011.
77. C. Loop. *Smooth subdivision surfaces based on triangles*. PhD thesis, Department of Mathematics, University of Utah, 1987.
78. D.P. Luebke. A developer's survey of polygonal simplification algorithms. *Computer Graphics and Applications, IEEE*, 21(3):24–35, 2001.
79. M. Mathews and J.O. Smith III. Methods for synthesizing very high q parametrically well behaved two pole filters. In *Proceedings of the Stockholm Musical Acoustics Conference (SMAC 2003)(Stockholm)*. Royal Swedish Academy of Music, 2003.
80. D. Menzies. Phya and vfoley, physically motivated audio for virtual environments. In *Proceedings of the AES 35th International Conference in Audio for Games, London UK*. Audio Engineering Society, 2009.
81. B. Mirtich and J. Canny. Impulse-based simulation of rigid bodies. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 181–ff. ACM, 1995.
82. S.K. Mitra and Y. Kuo. *Digital signal processing: a computer-based approach*, volume 128. McGraw-Hill New York, 1998.
83. Modartt. Pianoteq. `http://www.pianoteq.com/`.

84. P.M.C. Morse and K.U. Ingard. *Theoretical Acoustics*. Princeton University Press, 1986.
85. A. Nealen, M. Muller, R. Keiser, E. Boxerman, and M. Carlson. Physically Based Deformable Models in Computer Graphics. In *Computer Graphics Forum*, volume 25, pages 809–836. Blackwell Synergy, 2006.
86. Nokia. Qt Cross-platform Application and UI Framework. `http://qt.nokia.com/`.
87. James F. O'Brien, Perry R. Cook, and Georg Essl. Synthesizing sounds from physically based motion. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 529–536, New York, NY, USA, 2001. ACM.
88. J.F. O'Brien, C. Shen, and C.M. Gatchalian. Synthesizing sounds from rigid-body simulations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 175–181. ACM Press New York, NY, USA, 2002.
89. R. Ohayon and C. Soize. *Structural Acoustics and Vibration: Mechanical Models, Variational Formulations and Discretization*. Academic Press, 1998.
90. OpenGL. OpenGL High Performance Graphics. `http://www.opengl.org/`.
91. A.V. Oppenheim and R.W. Schafer. *Discrete-time signal processing*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1989.
92. S. Papetti. *Sound modeling issues in interactive sonification - From basic contact events to synthesis and manipulation tools*. PhD thesis, University of Verona, Italy, 2010.
93. S. Papetti, F. Avanzini, and D. Rocchesso. Numerical methods for a non-linear impact model: a comparative study with closed-form corrections. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19, 2011.
94. T.W. Parks and C.S. Burrus. *Digital filter design*. Wiley-Interscience, 1987.
95. D.K. Phillips. *Algorithms and Architectures for the Multirate Additive Synthesis of Musical Tones*. PhD thesis, School of Engineering, Durham University, UK, 1996.
96. D.K. Phillips. Multirate additive synthesis. *Computer Music Journal*, 23(1):28–40, 1999.
97. P. Polotti and D. Rocchesso, editors. *Sound to Sense - Sense to Sound. A state of the art in Sound and Music Computing*. Logos Verlag, Berlin, 2008.
98. A. Quarteroni, R. Sacco, and F. Saleri. *Numerical mathematics*. Springer Verlag, 2007.
99. R. Rabenstein, S. Petrausch, A. Sarti, G. De Sanctis, C. Erkut, and M. Karjalainen. Blocked-based physical modeling for digital sound synthesis. *Signal Processing Magazine, IEEE*, 24(2):42–54, 2007.
100. N. Raghuvanshi and M.C. Lin. Interactive sound synthesis for large scale environments. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 101–108. ACM New York, NY, USA, 2006.
101. M. Rath and D. Rocchesso. Continuous Sonic Feedback from a Rolling Ball. *IEEE Multimedia*, pages 60–69, 2005.
102. U. Rathmann and J. Wilgen. Qwt - Qt Widgets for Technical Applications. `http://qwt.sourceforge.net/`.
103. Curtis Roads. *The computer music tutorial*. the MIT Press, 1996.
104. D. Rocchesso and F. Fontana, editors. *The Sounding Object*. Mondo Estremo, Firenze, 2003.
105. X. Serra and J. Smith. Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. *Computer Music Journal*, 14(4):12–24, 1990.
106. Game Physics Simulation. Bullet Physics Library. `http://bulletphysics.org`.
107. A. Sirdey, O. Derrien, R. Kronland-Martinet, and M. Aramaki. Modal analysis of impact sounds with esprit in gabor transforms. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-11), IRCAM-Paris, France*, 2011.

108. J.O. Smith. Viewpoints on the history of digital synthesis. In *Proceedings of the 1st International Computer Music Conference*. International Computer Music Association, 1991.
109. J.O. Smith. Physical Modeling Using Digital Waveguides. *Computer Music Journal*, 16:74–74, 1992.
110. JO Smith. Introduction to Digital Filters with Audio Applications. Available online at `http://ccrma.stanford.edu/~ jos/filters`, 2012.
111. JO Smith. Physical Audio Signal Processing. Available online at `http://ccrma.stanford.edu/~ jos/filters/pasp.html`, 2012.
112. JO Smith. Spectral Audio Signal Processing. Available online at `http://ccrma.stanford.edu/~ jos/filters/sasp.html`, 2012.
113. W. Soedel. *Vibrations of Shells and Plates*. CRC Press, 2004.
114. Viscount International S.p.A. Physis Organs. `http://www.physisorgans.com/index.shtml`.
115. Viscount International S.p.A. Physis Piano. `http://physispiano.com/`.
116. J.C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations*. Society for Industrial Mathematics, 2004.
117. Dassault Systemes. Simulia Finite Element Analysis Software. `http://www.3ds.com/products/simulia`.
118. Applied Acoustic Systems. Cromaphone Creative Percussion Synthesizer. `http://www.applied-acoustics.com/`.
119. T. Tolonen, V. Välimäki, and M. Karjalainen. Evaluation of modern sound synthesis methods. Technical Report, Helsinki University of Technology, 1998.
120. L. Trautmann and R. Rabenstein. *Digital Sound Synthesis by Physical Modeling Using the Functional Transformation Method*. Kluwer Academic/Plenum Publishers, 2003.
121. L. Trautmann and R. Rabenstein. Multirate Simulations of String Vibrations Including Nonlinear Fret-String Interactions Using the Functional Transformation Method. *Eurasip Journal on Applied Signal Processing*, pages 949–963, 2004.
122. PP Vaidyanathan. Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial. *Proceedings of the IEEE*, 78(1):56–93, 2002.
123. V. Välimäki, J. Pakarinen, C. Erkut, and M. Karjalainen. Discrete-time modelling of musical instruments. *Reports on progress in physics*, 69:1, 2006.
124. C.P. Van Den Doel. *Sound synthesis for virtual reality and computer games*. PhD thesis, The University of British Columbia (Canada), 1999.
125. K. van den Doel, P.G. Kry, and D.K. Pai. FoleyAutomatic: physically-based sound effects for interactive simulation and animation. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 537–544. ACM New York, NY, USA, 2001.
126. K. van den Doel, DK Pai, T. Adam, L. Kortchmar, and K. Pichora-Fuller. Measurements of perceptual quality of contact sound models. In *Proceedings of the International Conference on Auditory Display (ICAD 2002), Kyoto, Japan*, pages 345–349, 2002.
127. S.A. Van Duyne and J.O. Smith. Developments for the commuted piano. In *Proceedings of the 1995 International Computer Music Conference, Banff*, pages 335–343. International Computer Music Association, 1995.
128. G. Var Rossum. Python Programming Language. `http://www.python.org/`.
129. University of Verona VIPS Lab. Sound Design Toolkit (SDT). `http://www.soundobject.org/SDT/`.
130. Y. Visell, F. Fontana, B.L. Giordano, R. Nordahl, S. Serafin, and R. Bresin. Sound design and perception in walking interactions. *International Journal of Human-Computer Studies*, 67(11):947–959, 2009.

131. M. Wright, A. Freed, and A. Momeni. Opensound control: state of the art 2003. In *Proceedings of the 2003 conference on New interfaces for Musical Expression (NIME)*, pages 153–160. National University of Singapore, 2003.
132. C. Yeh and A. Röbel. Adaptive noise level estimation. In *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-06), Montreal, Canada*, pages 145–148, 2006.
133. S. Zambon. Un modello per la simulazione in tempo reale di un riverbero a piastra. In *Proceedings of the XVII Colloquio di Informatica Musicale, Venezia*. Associazione Informatica Musicale Italiana, 2008.
134. S. Zambon. Physis Piano Project Documentation. Internal Technical Report, Viscount International S.p.A., 2012.
135. S. Zambon, E. Giordani, F. Fontana, and B. Bank. Sistema per riprodurre il suono di uno strumento a corde. Deposited Italian Patent.
136. S. Zambon, H.M. Lehtonen, and B. Bank. Simulation of piano sustain-pedal effect by parallel second-order filters. In *Proc. Int. Conf. on Digital Audio Effects (DAFx-08), Espoo, Finland*, 2008.
137. C. Zheng and D.L. James. Rigid-body fracture sound with precomputed soundbanks. *ACM Transactions on Graphics (TOG)*, 29(4):69, 2010.
138. C. Zheng and D.L. James. Toward high-quality modal contact sound. In *ACM Transactions on Graphics (TOG)*, volume 30, page 38. ACM, 2011.
139. O.C. Zienkiewicz and R.L. Taylor. *The finite element method. 1. Basic formulation and linear problems*. McGraw-Hill, 1989.
140. O.C. Zienkiewicz and R.L. Taylor. *The finite element method. 2: Solid and fluid mechanics dynamics and non-linearity. repr*. McGraw-Hill, 1998.
141. U. Zölzer, editor. *DAFX-Digital Audio Effects, 2nd Edition*. John Wiley & Sons, New York, NY, USA, 2011.