A STRATEGY FOR MODULAR IMPLEMENTATION OF PHYSICS-BASED MODELS

Matthias Rath

Dipartimento di Informatica
Università degli studi di Verona, Italy
rath@sci.univr.it

ABSTRACT

For reasons of practical handling as well as optimization of the processes of development and implementation, it is desirable to realize realtime models of sound emitting physical processes in a modular fashion that reflects an intuitively understandable structure of the underlying scenario. At the same time, in discrete–time algorithms based on physical descriptions, the occurance of non–computable instantaneous feedback loops has to be avoided. The latter obstacle prohibits the naive cross-connection of input–output signal processing blocks. The following paper presents an approach to gain modularity in the implementation of physics-based models, while preventing non–computable loops, that can be applied to a wide class of systems. The strategy has been realized pratically in the development of realtime sound models in the course of the *Sounding Object* [1] European research project.

1. INTRODUCTION

One common question in the development and implementation of sound generating algorithms based on models of physical processes is that of possible modularity. It is often desirable to represent distinct independent objects in a physical scenario to be modelled, a string, a hammer, a bow..., in according distinct algorithms that might be developed independently and combined in a second step. Ideally, implementational representations of physical objects or processes should be combinable, i.e. interconnectable, finally on a higher level of implementation, e.g. at runtime through a graphical interface, according to the intended overall physical scenario. E.g., in mechanical reality solid objects such as strings and hammers may be combined in various ways, "a hammer striking a string", "a hammer sliding along a string", "two hammers colliding"..., while their inner structure and properties stay generally unchanged (assumed that involved forces are not excessive, not causing lasting deformation). For practical use (in particular for musicians or non–experts) it would be of strong value to preserve this "modularity" of physical reality, throughout the process of modelling and implementation in such a way that software objects representing physical units (strings, hammers, interaction processes...) might be connected in a way analog to mechanical combination. Furthermore, development and implementation may be optimized when algorithms representing physical objects can be reused in various combinations. Usually, in implementations of sound synthesis based on physical models, specific closed physical systems are handled and changes in the underlying physical system require the repetition of the whole process of development (discretization of differential equations) and implementation.

The obstacle that is responsible for the difficulty to achieve modularity in implementations of physical models is the need to avoid the occurance of non–computable instantaneous loops. Such non–computabilities occur e.g. when naively connecting two signal processing blocks with no additional delay in such a way that the input, internal state and output vector of one block instantaneously depends on the contemporaneous values of the other block and vice versa. The most simple reaction to the problem, the insertion of an ad–hoc delay somewhere in the non–computable chain necessarily introduces errors that are hard to control. Thus, unproblematic modularity is usually only achieved when cross–dependencies of input/output blocks are excluded. The latter is e.g. the case when fixed force profiles are assumed for the excitation of resonators, such as for contacts of solid objects (e.g. [2], [3]); this however, is a rather restricted model of physical reality where in fact interaction forces and the contemporaneous states of interacting objects instantaneously mutually depend. During periods of interaction indeed, generally distinct, independent physical objects form a common system.

The occurance of non–computable, instantaneous loops is usually avoided during the process of discretization. One approach that allows the handling also of non–linearities (and which forms a basis for the strategy presented in this paper) has been developed by Borin et al. under the name of "*K-method*" [4]. As a side effect of such strategies to avoid instantaneous loops in discrete-time algorithms already at continuous–time level, before and during the process of discretization, eventual structurization of an initial physical scenario, e.g. into distinct solid objects, is generally not passed on to the final algorithm since describing continuous–time equations are merged in a first step. The whole development process then has to be repeated for varied physical scenarios, e.g. exchanges of objects or different modes of interaction.

In the following I present an approach to combine independent discrete–time algorithms in a modular fashion without introducing non–computable instantaneous loops or artificial ad–hoc delays. The key ideas are inspired by the *K-method* as described extensively in [4] and some final technical details are taken over identically and not handled here again; I instead refer to [4] at respective points. The following approach however attacks the problem directly on discrete–time level; it thus allows the integration also of algorithms that were derived from continuous–time differential equations through different methods of discretization or modelling techniques (finite–elements approximations, digital waveguides).

2. INITIAL SETTING, FORMULATION AND SCOPE

The implementational strategy described in this paper was developed during the work on realtime physics-based sound models of scenarios of contacting solid objects that was done as part of the *Sounding object* (*SOb*) European research project [1]. Behind the goal of modularity in implementation lies the central consideration, that the solid objects in the intended scenarios ("hitting", "bouncing", "rolling", "rubbing"...) show a charateristic, individual inner behavior which can be described independently from the different ways in which these objects may interact — impact and friction are at the core of the cases looked at in the course of *SOb* project. The same decomposition into objects of fixed independent characteristics and a specific way of interaction can surely be applied to many other sound–emitting processes, also such of non–mechanical, e.g. electro-magnetic nature. For this reason, the developed technique is assumed to be of potential wide use in the field of modelling of physical systems for sound synthesis.

Relevant for the interaction of the distinct objects in our cases of interest, is usually only a limited configuration, not their complete internal state. Furthermore, internal properties of interacting (i.e. here: contacting) objects can be described in different ways (such as in terms of resonant modes), independently from the interaction, that does not induce permanent changes to the objects. On the other hand, information is exchanged only via the objects' configurations in the areas of contact; the entire state of the objects can generally not be deduced from their behavior in one, or some, limited contact areas. A structure of implementation is therefore of interest that allows to develop independently, computational algorithms representing distinct objects and processes of interaction, and to freely connect such algorithms without the need of further adaptation. In the following, I refer to representations — of whatever nature, discrete–time (mostly) or continuous–time, of independent objects in the explained sense as "*resonators*", and representations of processes of interactions as "*interactors*". The term *resonator* here aims solely at the presence of some sort of memory, i.e. some internal state that is relevant for the subsequent, future behavior. This notion of an internal state is reflected through a differential operator in continuous–time representations, while we have some temporally changing state vector, $\mathbf{w}$, in the discrete–time algorithms, with a "state–update" algorithm, $\mathbf{w}(n) \rightarrow \mathbf{w}(n+1)$; no general a priori specifications, e.g. concerning linearity, are given at this point. For simplicity, *interactors* are here assumed to be memory-less, i.e. instantaneous relations; this assumption may be bypassed [1] but is unproblematic and simplifies the description.

Resonators and *interactors* are connected through input and output vectors that can most easily be thought of as forces $\mathbf{f}$ (coming from the interactor) and spatial position–velocity configurations $\mathbf{x}$, as in our concrete case; the complete state $\mathbf{w}$ of the *resonator* is generally not passed to the *interactor*. Figure 1 gives a sketch of the intended modular structure as described, and the connected problem; only one *resonator* is depicted here, which has no influence on the validity of the following argumentations. In the most strict sense, "modularity" would mean here, that discrete–time realizations of *resonators* and *interactor* formulas can be exchanged and "plugged" at this discrete–time level without any further knowledge about the internal algorithms or their origins, such as an underlying continuous–time model or the used technique of discretization (such as bilinear transform, Euler backward differencing...) whatsoever. Discrete *resonators* should be handable as "black boxes" that produce output vectors at every time step depending on their contemporaneous input vector and the hidden state-vector. It is seen that this goal conflicts with the instantaneous cross-relationship given by the *interactor*: in Figure 1, $\mathbf{f}(n)$ would be computed from $\mathbf{x}(n)$, which in turn depends on $\mathbf{f}(n)$; this loop can not be resolved without additional information about the internal structure of the *resonator*, i.e. without "breaking the black box". A non-computable instantaneous feedback-loop occurs.

[1]A model of friction interaction has been developed and implemented, that makes use of the structure presented here, and, indeed, a friction *interactor* with internal memory.

3. MODULARITY USING "LABELED BLACK BOXES"

The described problem is solved and modularity is reached in the development and interconnection of the *resonators* and *interactors* through a "labeled–black–box" approach. It is clear that discrete–time *resonators*, in the situation of Figure 1, cannot be handled as strict black boxes, in the sense of passing input to output vectors without additional information. However, as we will see now, it is on the other hand not necessary to reveal the origin and complete internal structure of the *resonator* algorithm, nor to reconstruct the whole computational structure for each change of objects or interaction. Under certain presumptions on the *resonator* algorithm we are able to resolve the instantaneous feedback loop, with the help of a "label" attached to the black box, representing minimum information about its hidden internal structure. The important point here is the exact specification of these presumptions on the *resonator* and of the "minimum information necessary", and the derivation of a uniform representation and interconnection procedure. The developed solution, that is now presented in detail, is inspired by, and closely related to, the *K-method*; we however work directly and only on the discrete–time level without referring back to (possible) continuous–time origins of the discrete algorithms. I will finally apply the techniques inherited from the *K-method*, that are not explained in detail again; at the point I refer to [4].

In discrete time, the most general *resonator* consists of a discrete–time state vector $\mathbf{w}$ and some "time-step" or "update" function R such that

$$\mathbf{w}(n) = R(\mathbf{w}(n-1), \mathbf{e}(n), \mathbf{f}(n)) \ , \tag{1}$$

where $\mathbf{f}$ is the output ("force") vector of the *interactor* (see Figure 1) and the vector $\mathbf{e}$ represents some external influence on the *resonator*, that is *independent from* the *interactor*. [2] In plain words, with each time step, the *resonator* state is updated from the previous state vector and the contemporaneous external input vectors $\mathbf{f}$ and $\mathbf{e}$, coming from the *interactor* resp. some independent external source. Further on, the *resonator* shows a representing configuration vector $\mathbf{x}(n)$ to the "outside world", on which in turn $\mathbf{f}(n)$ depends. In this application here, a vibrating solid object is accessed through its "configuration", position and velocity, in a certain contact point (or area). $\mathbf{x}(n)$ "represents" (to the outside, in particular the *interactor*) the *resonator* in its state $\mathbf{w}$ via some function S:

$$\mathbf{x}(n) = S(\mathbf{w}(n)) \ . \tag{2}$$

Combining equations (1) and (2), we can see $\mathbf{x}(n)$ as a function of $\mathbf{f}(n)$ and the vectors $\mathbf{w}(n-1)$ and $\mathbf{e}(n)$, that are known from the previous time step resp. an external input:

$$\mathbf{x}(n) = S(R(\mathbf{w}(n-1), \mathbf{e}(n), \mathbf{f}(n))) \ . \tag{3}$$

[2]For clarity of the picture, $\mathbf{e}$ is not depicted in Figure 1 as not relevant for the general idea and unproblematic.

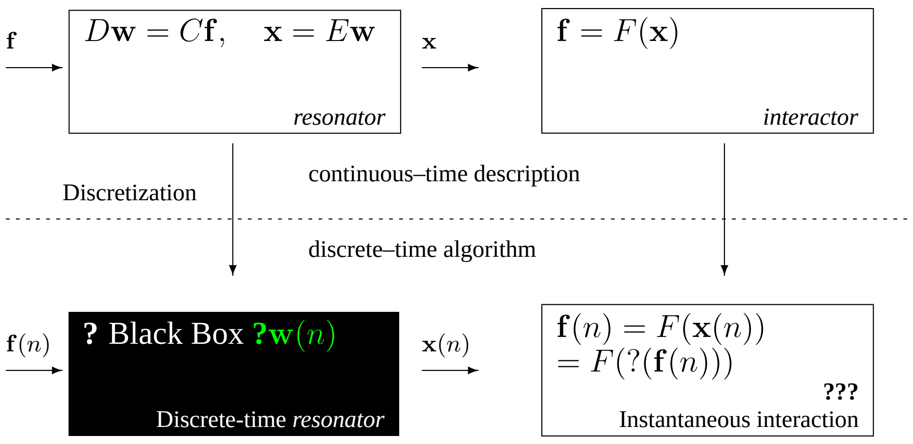
$\mathbf{f}$
$D\mathbf{w} = C\mathbf{f}, \quad \mathbf{x} = E\mathbf{w}$
resonator
$\mathbf{x}$
$\mathbf{f} = F(\mathbf{x})$
interactor
Discretization
continuous–time description
discrete–time algorithm
$\mathbf{f}(n)$
? Black Box ?$\mathbf{w}(n)$
Discrete-time *resonator*
$\mathbf{x}(n)$
$\mathbf{f}(n) = F(\mathbf{x}(n))$
$= F(?(\mathbf{f}(n)))$
???
Instantaneous interaction

Figure 1: *A sketch of the goal of "modularity" in implementation and the connected problem. It is desirable to represent objects involved in modeled scenarios as "black boxes" that generate output from input values, without the necessity of further information (thus "black") about their origins or inner structures. This goal conflicts with the instantaneous cross-dependency of values, due to the description of the interaction. Such non-computabilities are usually excluded already at the continuous–time level (as done in the original* K-method*), which generally destroys the independence of* resonators *and* interactors*, that is the second main goal behind the term "modularity".*

The condition that is now imposed on the *resonator*, is for the concatenation $S \circ R$ of the functions R and S to split into two summands, one of which depends only on the known vectors $\mathbf{w}(n-1)$ and $\mathbf{e}(n)$, and another depending *linearly* only on $\mathbf{f}(n)$. (The symbol "$\overset{!}{=}$" is used in the following equation to indicate that as a necessary precondition for the applicability of the following approach this relation with some T and L as above has to exist):

$$\begin{aligned}&(S \circ R)(\mathbf{w}(n-1), \mathbf{e}(n), \mathbf{f}(n))\\ &= S(R(\mathbf{w}(n-1), \mathbf{e}(n), \mathbf{f}(n)))\\ &\overset{!}{=} T(\mathbf{w}(n-1), \mathbf{e}(n)) + L(\mathbf{f}(n)), \quad L \text{ linear.}\end{aligned} \tag{4}$$

This condition is fulfilled in particular if both R and S are linear, as in our case of modal description with "pick up"points, or if both functions split in the described way. It is however thinkable that the condition holds neither for R nor S, but for the concatenation $S \circ R$, i.e. that non-linearities "cancel out". L as a linear mapping between finite-dimensional vectors can also be seen as a matrix $\mathcal{L}$ whose dimensions are the dimensions of $\mathbf{f}$ resp. $\mathbf{x}$ and we may write $\mathcal{L} \cdot \mathbf{f}$ instead of $L(\mathbf{f})$. If we now define $\mathbf{p}(n) \triangleq T(\mathbf{w}(n-1), \mathbf{e}(n))$, combine equations (4) and (3) to

$$\mathbf{x}(n) = \mathbf{p}(n) + \mathcal{L} \cdot \mathbf{f}(n) \tag{5}$$

and recall the definition of the *interactor*

$$\mathbf{f}(n) \triangleq F(\mathbf{x}(n)) \ , \tag{6}$$

we finally receive the crucial equation that determines $\mathbf{f}(n)$:

$$\mathbf{f}(n) = F(\mathbf{p}(n) + \mathcal{L} \cdot \mathbf{f}(n)) \ . \tag{7}$$

It is underlined again, that here $\mathbf{p}(n)$ does not depend on $\mathbf{f}(n)$, i.e. can be computed before $\mathbf{f}(n)$, and an implicit relation $\mathbf{p}(n) \mapsto \mathbf{f}(n)$ has been found, that completely coincides with the situation in [4], Section C. This implicit relation (7) may be transformed into an explicit mapping — under the conditions of the *implicit mapping theorem* — or solved through an approximation; I refer to [4] for the detailed discussion that is not repeated here. It has to be noted from equation (5), that $\mathbf{p}(n)$ coincides with $\mathbf{x}(n)$ if $\mathbf{f}(n)$ is zero:

$$\text{If } \mathbf{f}(n) = 0, \text{ then } \Rightarrow \mathbf{x}(n) = \mathbf{p}(n) \tag{8}$$

In plain words, $\mathbf{p}(n)$ is equal to the output vector of the *resonator* under some fictitious "*pseudo-update*" with zero input (force). As a result, we finally see that the non-computable loop in Figure 1, $\mathbf{f}(n) = F(?(\mathbf{f}(n)))$, can be turned into a resolvable implicit relation, equation (7), if the black box of the *resonator* is equipped with
1. a label containing $\mathcal{L}$ and
2. a *pseudo-update* functionality, that delivers the "simulated" *resonator* output with zero input, without de–facto updating the internal state.
The dimensions of $\mathcal{L}$ have already been mentioned as being of a similar order as those of $\mathbf{f}$ and $\mathbf{x}$; exactly, $\mathcal{L}$ contains $\dim(\mathbf{f}) \times \dim(\mathbf{x})$ elements. If the vectors $\mathbf{f}$ and $\mathbf{x}$ are of orders below 10 — we most often deal with force and position/velocity vectors in one- to three-dimensional space — passing $\mathcal{L}$ whenever necessary, i.e. when *resonator* or *interactor* or any of their attributes (such as modal parameters or the point of interaction for impact or friction) are exchanged, is thus a negligible overhead in comparison with the processing of the in- and output vectors $\mathbf{f}$ and $\mathbf{x}$ that have to be passed with each time step, i.e. usually 44100 times per second. In particular is the size of $\mathcal{L}$ often small compared to the state vector of the *resonator*: the internal state vector of a *digital waveguide* e.g., can easily reach dimensions of the order of [3] 10000 while its representing external configuration would usually be of dimension 2 (position and velocity in a point...).

Summing up, the update-cycle at each time-step n for the complete discrete–time system consists of the schedule given in Figure 2.

[3]A simple two-directional *waveguide* with a minimal frequency of 10 Hz at a sample-rate of 44100 Hz, contains at least two delay lines of 4410 samples each.

1. Read in external variables to the *resonator(s)*, such as additional external forces or related signals ($\mathbf{e}(n)$ in the notation above).
2. *Pseudo-update* of the *resonator(s)* from previous state $\mathbf{w}(n-1)$ and $\mathbf{e}(n)$, without de–facto update of the internal *resonator(s)* state. $\mathbf{p}(n)$ is passed to the *interactor*.
3. Calculation of $\mathbf{f}(n)$ from $\mathbf{p}(n)$. The mathematical technique for this step depends on the *interactor* function F. In the concrete cases here of impact an explicit formulation can be used in the (piece-wise) linear case, while the nonlinear relation is solved through *Newton–Raphson* approximation [5].
4. After $\mathbf{f}(n)$ has been computed and passed to the *resonator(s)*, the internal *resonator* states are updated, $\mathbf{w}(n-1) \mapsto \mathbf{w}(n)$.

Figure 2: *The update schedule at each time–step (sample cycle).*

4. EXAMPLE APPLICATIONS

The previously described strategy has been applied succesfully in the implementation of several models of contacting solid objects. Combined can be *resonators* in general modal description, realized in discrete time using discretization by bilinear transform, and the simple case of an inertial point mass. A digital waveguide *resonator* is being developed. These representations of solid objects are integrated through *interactors* modelling interaction in contacts based on impact or friction. The resulting models have been implemented in *C* as modules for the realtime sound software *pd* [6]. Due to the surrounding architecture of *pd*, the different *resonators* and *interactors* are linked statically at compilation time, but dynamical linkage would be straightforward in a suitable software environment. The modular approach presented above strongly minimized development efforts. All details of the development process, from the physical description to computational algorithms and practical handling can be found in [7] (more exactly [8], [9] and [10]).

5. CONCLUSION

In this paper, an approach for the modular implementation and development of computational algorithms based on physical models for sound generation has been described. The approach, that has been derived and practically realized during the work of implementation in the course of the European project *The Sounding Object* [1], has been presented in a general form and can be applied to a wide range of sound-producing scenarios.

6. ACKNOWLEDGEMENTS

This work has been supported by the European Commission under contract IST-2000-25287.

7. REFERENCES

[1] "*The Sounding Object (SOb)*," European research project (IST-25287, `http://www.soundobject.org`) as part of the *Disappearing Computer (DC)* proactive initiative (`http://www.disappearing-computer.org/`).

[2] K. van del Doel, *Sound synthesis for virtual reality and computer games*, Ph.D. dissertation, University of British Columbia, 1998.

[3] K. van del Doel, P. G. Kry, and D. K. Pai, "Foleyautomatic: Physically-based sound effects for interactive simulation and animation," in *Proc. ACM Siggraph 2001*, Los Angeles, Aug. 2001.

[4] G. Borin, G. D. Poli, and D. Rocchesso, "Elimination of delay-free loops in discrete-time models of nonlinear acoustic systems," *IEEE Trans. on Speech and Audio Processing*, vol. 8, no. 5, pp. 597–605, September 2000.

[5] F. Avanzini and D. Rocchesso, "Modeling Collision Sounds: Non-linear Contact Force," in *Proc. Conf. on Digital Audio Effects*, Limerick, December 2001, pp. 61–66.

[6] "Pure data, *pd*," `http://www.pure-data.org`

[7] D. Rocchesso and F. Fontana, Eds., *The Sounding Object.* Firenze, Italy: Mondo Estremo, 2003, `http://www.soundobject.org`

[8] D. Rocchesso and F. Avanzini, "Impact," in *The Sounding Object*, D. Rocchesso and F. Fontana, Eds. Firenze, Italy: Mondo Estremo, 2003, pp. 125–129.

[9] ——, "Friction," in *The Sounding Object*, D. Rocchesso and F. Fontana, Eds. Firenze, Italy: Mondo Estremo, 2003, pp. 129–136.

[10] ——, "Discrete-time-equations," in *The Sounding Object*, D. Rocchesso and F. Fontana, Eds. Firenze, Italy: Mondo Estremo, 2003, pp. 124–125.