ANALYSIS TECHNIQUES FOR PHYSICAL MODELING NETWORKS

Abstract: We are concerned with the development of an analysis tool dedicated to the mass-spring networks approach in sound synthesis by Physical Models. Thanks to an analogy between mechanical networks and neural networks, some algorithms to adapt the parameters of the mechanical network are proposed. The interaction between the model, the instrumentalist and the instrument to be modeled is discussed and tested. At last, symmetries and differences between analysis and synthesis are discussed, in the case of Physical Models.

INTRODUCTION

Physical Models have proven to be a powerful sound synthesis method whose advantages are, among others, richness of sounds, simplicity of control parameters, natural communication between the performer and the computer (Hiller and Ruiz 1970 *in* Borin 1992; Cadoz, Luciani and Florens 1981; Smith 1992; Borin 1992).

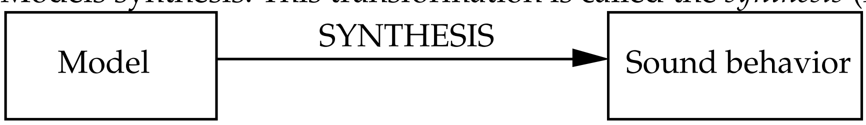
If several methods for Physical Models synthesis have been developed so far (Roads 1994), the design of the corresponding analysis tools remains a difficult issue. In this paper we will especially study the analysis in one kind of Physical Models approach: the mass-spring models. The experimental results remain theoretical, and mainly serve the purpose of validating the algorithms. Concrete musical application is possible, based on the proposed approach, if it is assumed that some technological constraints are manageable.

Our process is driven by the synthesis system we are dealing with, namely the CORDIS-ANIMA system (Cadoz, Luciani and Florens 1993) developed at ACROE. In the first section, four features of synthesis in CORDIS-ANIMA are listed, which will serve as guidelines for the design of an analysis tool. Then, a parallel between mass-spring networks and connectionist (neural) networks is proposed, which allows us to derive some analysis algorithms. Then the issue of the interaction in analysis is discussed and tested. The last section compares synthesis and analysis from a broader point of view.

From Synthesis to Analysis

There are many different methods for producing sounds with a computer. These methods differ in the type of *representation* that is used by the computer. This representation is usually created by the user, inside a given representation framework. Then, this representation, called a model, is processed by the computer to produce the synthetic sound.

From an abstract point of view, one can consider this execution as the transformation from a model to a sound behavior. A sound behavior is simply a sound signal in case of classical synthesis, while it corresponds to the response to an excitation (a gesture) in the case of Physical Models synthesis. This transformation is called the *synthesis* (Fig. 1).



<Fig. 1. Synthesis>

This synthesis would be sufficient for the purpose of musical creation if the user had a very good knowledge of the relation between a model and the sound behavior this model yields. Unfortunately, this is not the case : One cannot directly build a model that produces a desired sound, existing or imaginary. Thus the sound synthesis system should be provided with a tool that could automatically design such a model, by transforming a given sound behavior into the corresponding model. This transformation is called the *analysis*.

<Fig. 2. Analysis>

From a functional point of view, the analysis is strictly the reverse of the synthesis (see Fig. 2). Furthermore, in several sound synthesis techniques, this symmetry can also be observed into the mechanisms themselves: For example, in the case of additive synthesis, equations for deriving the parameters of the model from a sound strongly look like the equations for creating the sound from the model; They are respectively called Fourier analysis and Fourier synthesis.

Unfortunately, for other sound synthesis techniques, the analysis is not directly derived from the synthesis equations, and is radically different from it (see the discussion in the last section of the paper). The topic of this paper is the analysis techniques dedicated to Physical Models approach, in particular mass-spring models (Cadoz, Luciani and Florens 1981; Roads 1994), that is networks of masses, linked together with viscoelastic links. This work has two objectives :

- finding some analysis techniques for mass-spring models, which share some properties with the synthesis,
- understanding the relationship between synthesis and analysis in the case of Physical Models.

The first step of this work is thus to list the features of the synthesis technique which constitutes our starting point.

Whereas many sound synthesis techniques use a model of a signal, the Physical Models approach uses a model of the origin of the sound, that is a model of a music instrument. The modeling of the phenomenon is replaced by the modeling of the cause (Cadoz 1990). This approach has many advantages (Borin *et al.* 1992), like a better comprehension of the model, more intuitive interaction with it, possibility of assembling sub-structures.

The CORDIS-ANIMA system, which constitutes our basis has the four following features (Cadoz, Luciani and Florens 1993):

<u>1- Elementary modularity:</u> Because mass-spring models are indeed networks of automata, any model can be decomposed into smaller networks, each of which can be decomposed into still smaller networks, and so on to the smallest basic elements (the automata) : the matter point and the link element. Also, any model can be merged with another model to build a more complex model. This modularity has many advantaged :

- simplicity of handling,
- unified language of description, whatever the level of complexity,
- possibility to easily put models together,
- efficiency of implementation.

2- Bidirectionnality of communication: Matter points and link element are always exchanging forces and displacements in both direction, in order to follow the mechanical principle of action-reaction: any object *A* that mechanical acts on another object *B* necessarily receives a reaction from the object *B*.

3- Real-time handling: In order to obtain more expressive sounds, the instrumentalist acts in real-time on the simulated instrument. This implies a simulation speed constraint : the calculation must be performed 40000 times per second.

4- Force-feedback: This action, in order to be similar to the natural action of the instrumentalist, must be associated with a reaction, according to the bidirectionnality mentioned above. Thus, the instrumentalist not only sends forces to the simulated instrument but also receives some forces from this instrument, thanks to a force-feedback device, called a *Force-Feedback Gestural Transducer* (Cadoz, Lisowski and Florens 1990).

In the present study, we intend to design analysis techniques that would retain these four properties. Here, in first approach, only linear vibrating structures are considered.

From Cognitive to Physical Networks

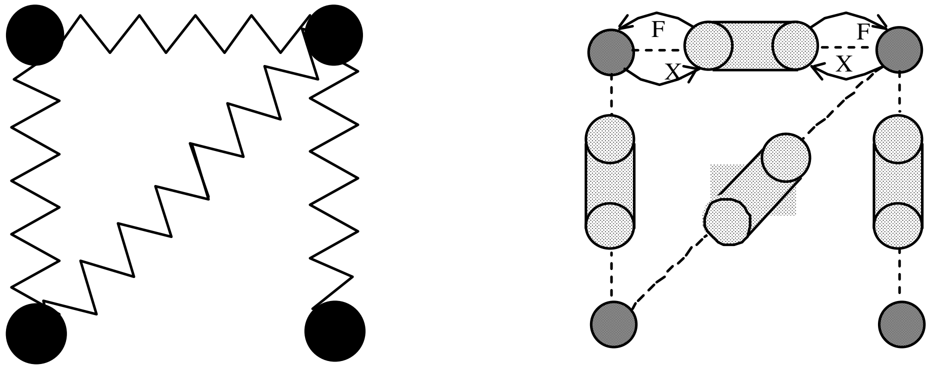
Our approach is based on an analogy between neural and mass-spring networks. This section describes how algorithms can be derived from the former to the later, and proposes en experimental validation of the efficiency of the algorithms.

Description of the Algorithms

Basic principles

From an algorithmic point of view, analysis consists in finding an automata network that reproduces a given sound behavior. One field of computer science is especially dedicated to such a problematic: neural or connectionist networks. In particular, *supervised* networks (Rumelhart and McClelland 1986) adapt themselves according to an external given desired behavior. The networks are not determined in one shot, but according to an iterative process, starting from an initial network and reaching the final network. This process is called *learning*. These learning networks, originally inspired from biological considerations, can be adapted in the case of mechanical networks (Szilas and Cadoz 1993).

Connectionist nets suitable for our purpose must be supervised, recurrent (the graph of connectivity contains circuits), with hidden units (where no desired behavior is given) and real valued (not binary). Some known learning algorithms for such networks exist, able to adapt their parameters given a desired dynamic behavior (structure adaptation is not considered). The main ones are *Back-Propagation Through Time (BPTT)* (Williams and Peng 1990) and *Real-Time Recurrent Learning (RTRL)* (Williams and Zipser 1989). These algorithms are described below.



<Fig.3 Mechanical and CORDIS-ANIMA automata representations of a mass-spring network>

The mass-spring networks contain two types of automata (see Fig. 3), described by the following equations:

- the punctual mass is an automaton receiving forces and yielding one position signal; its equations are obtained by discretizing the fundamental principle of dynamics, which states that the sum of forces on a system is equal to the product of its mass by its acceleration:

$$Fext_i(n-1) + \sum_{j \in C(i)} F_{ji}(n-1) = m_i\left[\left(X_i(n) - X_i(n-1)\right) - \left(X_i(n-1) - X_i(n-2)\right)\right]$$

where $X_i(n)$ is the position of the mass i at time n, m_i the value of the mass i, $C(i)$ the set of indices of masses connected to the mass i, $F_{ji}(n-1)$ the force applied to the mass i by the spring located between the masses i and j, and $Fext_i(n-1)$ the external force applied to the mass i. Therefore, the equation of the punctual mass is:

$$X_i(n) = 2X_i(n-1) - X_i(n-2) + m_i^{-1}\left[\sum_{j \in C(i)} F_{ji}(n-1) + Fext_i(n-1)\right] \tag{1}$$

Note that some other discretizations are possible; This one gives the simpler equation but is not necessarily the most precise.

- the link element is an automaton receiving two positions and yielding two opposite forces, obeying the law of damped springs:

$$F_{ij}(n) = -F_{ji}(n) = k_{ij}\left(X_i(n) - X_j(n)\right) + z_{ij}\left[\left(X_i(n) - X_i(n-1)\right) - \left(X_j(n) - X_j(n-1)\right)\right] \quad (2)$$

where k_{ij} is the stiffness of the spring between the masses i and j, and z_{ij} is its damping factor. Once again, more sophisticated equations can be used. For visual simplicity, the link element is represented by a simple spring in Fig. 3 and following, although it also contains damping.

Let us define the error *Etot* between the actual behavior and the desired behavior by:

$$Etot = \sum_{n=1}^{N} E(n) = \sum_{n=1}^{N} \sum_{i \in A} \left(e_i(n)\right)^2$$

$$\begin{cases} e_i(n) = X_i(n) - Xd_i(n) \text{ if } i \in A \\ e_i(n) = 0 \text{ if } i \notin A \end{cases}$$

where A is the set of masses where a desired behavior is provided, $Xd_i(n)$ is the desired position of mass i and N is the total length of the simulation.

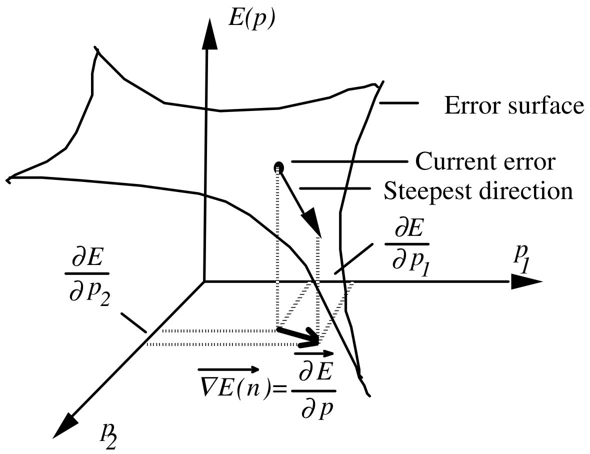
Parameters m_i^{-1}, k_{ij}, z_{ij} are adapted according to the gradient algorithm: the error *Etot* is a function of the parameters, which can be represented by a *surface* in a multidimensional space (fig. 4); the current error (with the current parameters) is a point on this surface; the principle of the algorithm is to move this point to a local minimum of the surface, by following the direction in which the slope is the steepest, that is the *gradient* of the error; Consequently, the parameters are modified according to:

$$\Delta m_i^{-1} = -\alpha \frac{\partial Etot}{\partial m_i^{-1}} = \sum_{n=1}^{N} \Delta m_i^{-1}(n) \text{ where } \Delta m_i^{-1}(n) = -\alpha \frac{\partial E(n)}{\partial m_i^{-1}} \quad (3)$$

$$\Delta k_{ij} = -\beta \frac{\partial Etot}{\partial k_{ij}} = \sum_{n=1}^{N} \Delta k_{ij}(n) \text{ where } \Delta k_{ij}(n) = -\beta \frac{\partial E(n)}{\partial k_{ij}} \quad (4)$$

$$\Delta z_{ij} = -\gamma \frac{\partial Etot}{\partial z_{ij}} = \sum_{n=1}^{N} \Delta z_{ij}(n) \text{ where } \Delta z_{ij}(n) = -\gamma \frac{\partial E(n)}{\partial z_{ij}} \quad (5)$$

The positive numbers α,β and γ, are the learning steps; the $\partial X / \partial Y$ notation denotes the partial derivatives of X according to Y.



<Fig.4 Illustration of the gradient descent algorithm; $p=(p_1,p_2)$ are the parameters and $E(p)$ the error to be minimized. Gradient descent consists in following the steepest direction>

The two following algorithms are obtained by deriving Eqs. (3) to (5) in two different ways. For more details on the mathematical basis justifying the way equations are derived, reefer to (Piché 1994). Basically, equations are obtained by applying the theorem of derivation of composed functions.

Back-Propagation Through Time

By applying the derivatives composition law on Eq. (3), one obtains:

$$\Delta m_i^{-1} = -\alpha \sum_{\eta=1}^{N} \frac{\partial Etot}{\partial X_i(\eta)} \frac{\partial X_i(\eta)}{\partial m_i^{-1}} = -\alpha \sum_{\eta=1}^{N} \frac{\partial Etot}{\partial X_i(\eta)} \left(\sum_{j \in C(i)} F_{ji}(\eta-1) + Fext_i(\eta-1) \right)$$

This equation shows that the variation of m_i^{-1} depends on values calculated for all time steps η.

Let us then define:

$$D_i(\eta) = \frac{\partial Etot}{\partial X_i(\eta)}$$

One obtains:

$$\Delta m_i^{-1} = -\alpha \sum_{\eta=1}^{N} D_i(\eta) \left(\sum_{j \in C(i)} F_{ji}(\eta-1) + Fext_i(\eta-1) \right) \tag{6}$$

For stiffness and damping factors, one has likewise:

$$\Delta k_{ij} = -\beta \sum_{\eta=1}^{N} d_{ij}(\eta) \big(X_j(\eta) - X_i(\eta) \big) \tag{7}$$

$$\Delta z_{ij} = -\gamma \sum_{\eta=1}^{N} d_{ij}(\eta) \Big[\big(X_j(\eta) - X_j(\eta-1) \big) - \big(X_i(\eta) - X_i(\eta-1) \big) \Big] \tag{8}$$

where:

$$d_{ij}(\eta)=\frac{\partial Etot}{\partial F_{ji}(\eta)}-\frac{\partial Etot}{\partial F_{ij}(\eta)}$$

Equations for obtaining signals $D_i(\eta)$ and $d_{ij}(\eta)$ can be calculated by deriving the expression of *Etot* by *X* and *F* respectively :

$$\begin{aligned}D_i(\eta)&=2e_i(\eta)+2D_i(\eta+1)-D_i(\eta+2)\\&\quad+\sum_{l\in C(i)}(k_{li}+z_{li})d_{li}(\eta)-z_{li}d_{li}(\eta+1)\end{aligned}\tag{9}$$

$$d_{ij}(\eta)=\frac{D_i(\eta+1)}{m_i}-\frac{D_j(\eta+1)}{m_j}\tag{10}$$

Expressions (9) and (10) are still correct for $\eta=N-1$ and $\eta=N$, if the initial conditions are settled as following:

$$D_i(N+1)=D_i(N+2)=\delta_{ij}(N+1)=0\tag{11}$$

The BPTT algorithm is described by Eqs. (6) to (10), with Eq. (11) for initialization. Equations (9) and (10) show the specificity of the algorithm : signals $D_i(\eta)$ and $d_{ij}(\eta)$, that are necessary for the adaptation of parameters (Equations (6) to (8)), cannot be calculated during the simulation, but have to be calculated backward, because the signals at time η depends on signals at time $\eta+1$ and $\eta+2$.. Typically, one has to store all signals of the simulation (positions *X* and forces *F*) and then adapt the parameters.

Real-Time Recurrent Learning

One defines additional signals, that represent the sensitivity of each position and force to the parameters of the network:

$$S_i^j(n)=\frac{\partial X_j(n)}{\partial m_i^{-1}}\qquad R_i^{lj}(n)=\frac{\partial F_{lj}(n)}{\partial m_i^{-1}}$$

$$\sigma_{ij}^l(n)=\frac{\partial X_l(n)}{\partial k_{ij}}\qquad \rho_{ij}^{hl}(n)=\frac{\partial F_{hl}(n)}{\partial k_{ij}}$$

$$\tau_{ij}^l(n)=\frac{\partial X_l(n)}{\partial z_{ij}}\qquad \chi_{ij}^{hl}(n)=\frac{\partial F_{hl}(n)}{\partial z_{ij}}$$

By differentiating Eqs. (1) and (2), with respect to the parameters, one obtains:

$$S_i^l(n)=2S_i^l(n-1)-S_i^l(n-2)+\sum_{j\in C(l)}R_i^{jl}(n-1)+\delta_{il}\sum_{j\in C(l)}F_{jl}(n-1)\tag{12}$$

$$R_i^{jl}(n)=\left(k_{jl}+z_{jl}\right)\left(S_i^j(n)-S_i^l(n)\right)-z_{jl}\left(S_i^j(n-1)-S_i^l(n-1)\right)\tag{13}$$

$$\sigma_{ij}^l(n)=2\sigma_{ij}^l(n-1)-\sigma_{ij}^l(n-2)+\frac{1}{m_l}\sum_{h\in C(l)}\rho_{ij}^{hl}(n-1)\tag{14}$$

$$\rho_{ij}^{hl}(n) = (k_{hl} + z_{hl})(\sigma_{ij}^{h}(n) - \sigma_{ij}^{l}(n)) - z_{hl}(\sigma_{ij}^{h}(n-1) - \sigma_{ij}^{l}(n-1)) + \delta_{hl,ij}(X_i(n) - X_j(n)) \tag{15}$$

$$\tau_{ij}^{l}(n) = 2\tau_{ij}^{l}(n-1) - \tau_{ij}^{l}(n-2) + \frac{1}{m_l}\sum_{h\in C(l)} \chi_{ij}^{hl}(n-1) \tag{16}$$

$$\chi_{ij}^{hl}(n) = (k_{hl} + z_{hl})(\tau_{ij}^{h}(n) - \tau_{ij}^{l}(n)) - z_{hl}(\tau_{ij}^{h}(n-1) - \tau_{ij}^{l}(n-1)) + \delta_{hl,ij}((X_i(n) - X_i(n-1)) - (X_j(n) - X_j(n-1))) \tag{17}$$

where δ is the Kronecker symbol:

$\delta_{ij} = 1$ iff $i = j$

$\delta_{ij,hl} = 1$ iff($i = h$ and $j = l$) or ($i = l$ and $j = h$)

One can finally calculate:

$$\Delta m_i^{-1}(n) = -2\alpha \sum_{l\in A} (X_l(n) - Xd_l(n)) S_i^l(n) \tag{18}$$

$$\Delta k_{ij}(n) = -2\beta \sum_{l\in A} (X_l(n) - Xd_l(n)) \sigma_{ij}^l(n) \tag{19}$$

$$\Delta z_{ij}(n) = -2\gamma \sum_{l\in A} (X_l(n) - Xd_l(n)) \tau_{ij}^l(n) \tag{20}$$

The RTRL algorithm is described by Eqs. (12) to (20). Contrary to BPTT, It can be computed forward, but involves more computation, because of the sensitivity signals that are added for the adaptation.

Mechanical interpretation of the algorithms

Classical comparisons between BPTT and RTRL involve two features:

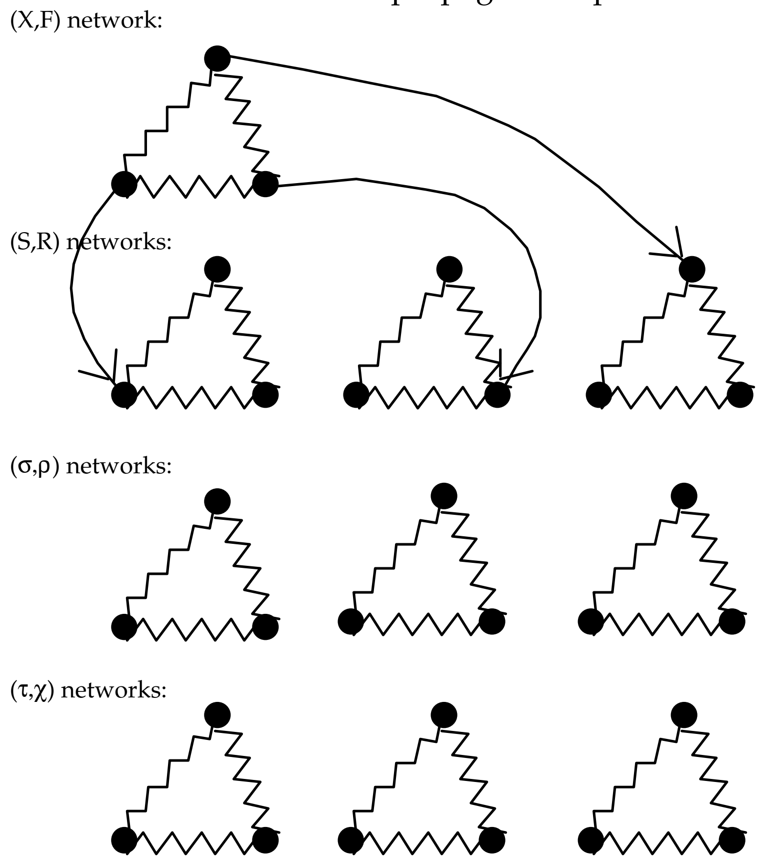
- computational complexity: BPTT is less costly than RTRL in terms of number of computations, but needs more storage;

- spreading of calculation in time: BPTT must store the sequence and then adapt the parameters whereas RTRL can perform continuously.

But it is also interesting, in our case, to establish whether the calculation can be interpreted in terms of mechanical networks similar to the ones used for synthesis (Eqs. (1) and (2)).

For BPTT, Eqs. (9) and (10), under a certain variable transformation, are equivalent to a mass-spring network simulated backwards through time. Unfortunately, the temporal storage involved by Eqs. (6) to (8) cannot be physically interpreted.

In the case of RTRL, it is noteworthy that according to Eqs. (12) and (13), S_i and R_i are positions and forces of a mass-spring network that is excited at point i. The same interpretation can be given to Eqs. (14) to (17). Thus, each adaptive parameter has its associated mechanical network, which is similar to the original one, one unit of which is excited, and one signal of which is used for adaptation. These networks are represented in Fig. 5, on which, for clarity, inter-networks excitation signals are only represented from the (X,F) network to the (S,R) networks (curved arrows). This interpretation is interesting because it shows that learning involves the same kind of processing as propagation: the learning network is entirely expressed in terms of units and connections, that is to say analysis is also performed by simulating a mass-spring network. Furthermore, it demonstrates that if the network is implemented in parallel then one learning update takes approximately the same time as does one propagation update.

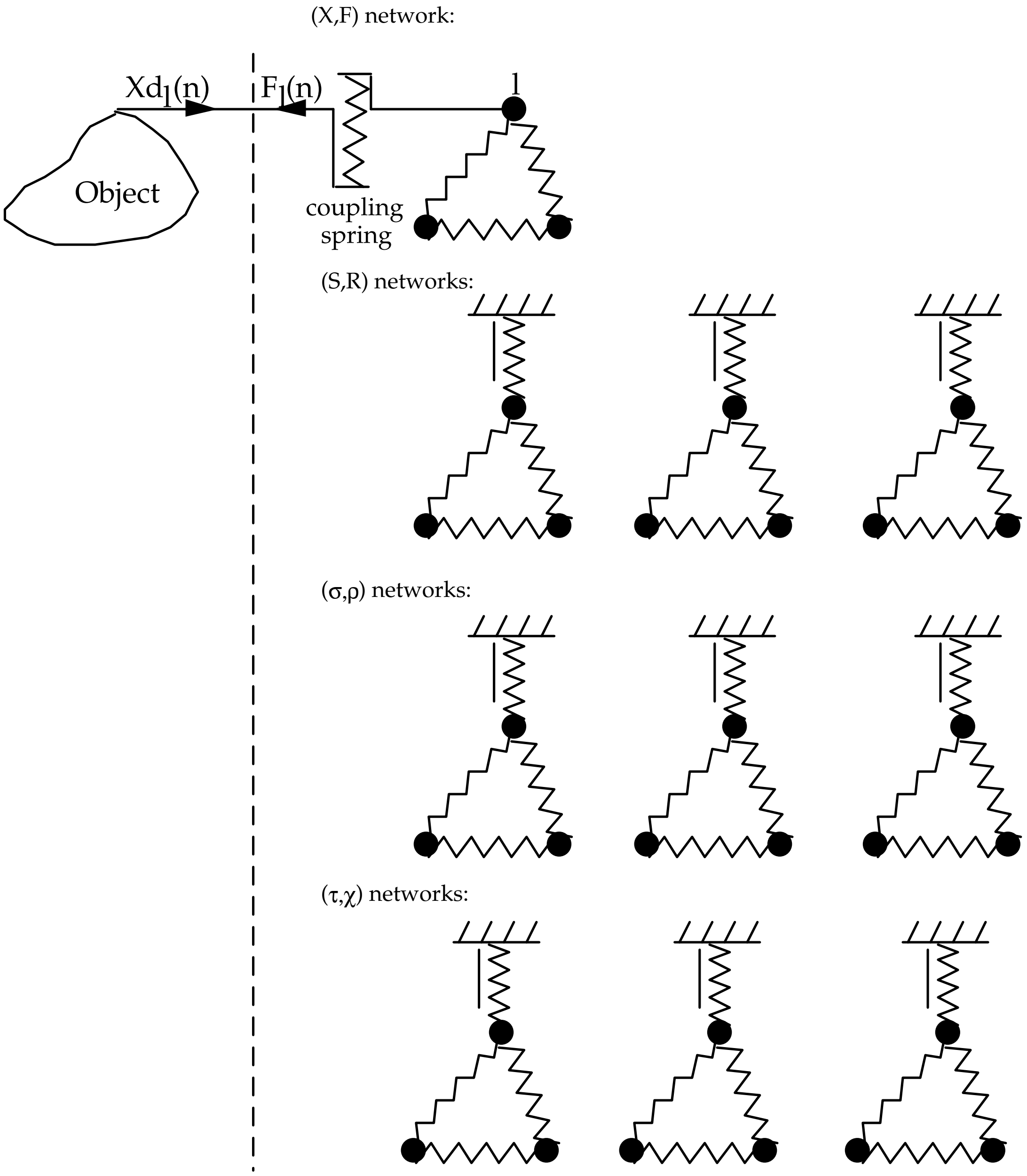


<Fig. 5. Mechanical interpretation of RTRL: the equations for adapting the parameters can be interpreted as a set of mass-spring models.>

However, the way the model and the instrument communicate does not respect the bidirectionnality of information exchange : the sound behavior is just fed into the adaptive network. In order to make this interaction mechanical, we introduce a viscoelastic link (a *coupling* damped spring) at each access point. Unfortunately, the fact that the model modifies the behavior of the instrument makes the true gradient calculation impossible ; the derivative of the position of the point in the instrument that is connected via the coupling spring to the access point must be set to zero, which is an approximation (see (Szilas 1995) for details).

A variant of this coupling is the semi-coupling : the coupling damped spring does not send any force to the object but only to the model. In this case, the true gradient can be calculated, because the derivative mentioned above are effectively zero, but semi-coupling is no longer physical.

In both cases, the introduction of this coupling damped spring means that when the mass i is an access point, the set $C(i)$ of indices of masses connected to mass i must include the point to which the coupling spring is attached. Thus, in Equations 12-17, signals R, ρ and χ must be computed at the coupling spring. These alterations amount to linking the additional networks represented in Figure 5 by damped springs attached to a fixed point (see Figure 6).



<Fig. 6. Mechanical interpretation of the coupling algorithm. The additional networks are now attached to the ground. >

Computer Simulations

The algorithms previously described use the gradient descent principle: they are not guaranteed to converge to a satisfying solution, because the error surface depicted in Figure 4 is usually complex : it contains local minima (learning is stucked), flat area (learning evolves vey slowly), or areas where the derivative is much higher in one direction than th other (learning is unstable). It is thus necessary to experimentally test them, in a systematic manner. Experiments have been limited to *simulated* scenarios, where the physical object is simulated with the same kind of networks as the model. This allows to perfectly control the task: if the two networks, the learning one and the static one, are chosen with the same topology, then one knows that a perfect solution does exist. All the experiments displayed here belong to this category. Furthermore, the networks are limited to five masses (including the ground, which is an infinite mass) entirely connected, and there is only one desired output in the network.

In order to estimate the similarity between the actual and desired behaviors in a meaningful way, we choose a different criterion than the squared error used by the algorithms: this new criterion should be independent of the excitation magnitude. We thus define the Mean Relative Difference (MRD), which is only one criterion among other possibilities, as:

$$MRD = \frac{\sum_{n=1}^{N} |X_a(n)-Xd_a(n)|}{N(\max\limits_{n=1,\ldots,N} \{Xd_a(n)\} - \min\limits_{n=1,\ldots,N} \{Xd_a(n)\})}$$

where a is the index of the mass where a desired behavior is given.

There are three types of parameters to be adapted: masses m (more precisely inverted masses m^{-1}), stiffness factors k and damping factors z. The protocol of experiment consists in taking for each type of parameter two values, one for the learning network and one for the static network, randomly within a certain interval, called the interval of initialization, then training the network until the MRD goes below a given threshold. Ideally, the two values will become equal. Note that, coarsely speaking, the wider the interval of initialization, the harder the task.

For coupling techniques, the problem is that the MRD during learning is different from the *real MRD*, measured when the model and the objet evolve freely. Thus, we are obliged to manually tune an *actual threshold on MRD* lower than the difference that we wish to reach, in order that at the end, the networks (in average) have really gone below the desired threshold.

For all experiments, twenty pairs made up a learning net and a static net are tested. Each net contains 5 masses fully connected (one of which is the ground) by viscoelastic links; there are 24 parameters to adapt. The learning stops either when the threshold on MRD is reached (case of success), or if the number of adaptations is higher than 50000, or if the parameters explode (bigger than 10^6 for inverted masses and 100 for stiffness and damping factors).

Four algorithms are tested: BPTT, RTRL, Real Coupling (RC) and Semi Coupling (SC). Two tests are applied, one where the intervals of initialization are quite small ([1.5,1.7] for masses, [0.09,0.11] for stifness and [0,0.05] for damping factors), in order to check that the algorithms converge when the starting point in near the optimal solution (local convergence), and the second where the interval is larger ([0.05,0.3] for stifness, while masses and damping factors are fixed at 1 and 0 respectively), which correspond to a difficult task. At last, those tests are applied in two conditions, the first with short sequences (50 samples) repeated many time (or epochs in neural networks terminology) and the second with a unique continuous sequence.

Thus we have sixteen experiments, each involving twenty different runs.

As a result, we found that the four algorithms passed the local convergence test, in the case of short sequences. In the case of a long sequence, the number of successful trials are shown in table 1.

		RTRL	BPTT	RC	SC
Long sequences	# of successful trials (out of 20)	7	13	19	19

<Table 1. Experimental results of the analysis algorithms for the local convergence test in the case of long sequences: there is a cyclic impulsion, on a single long sequence.>

RTRL performed particularly bad, because all unsuccessful runs "exploded". This explosion phenomenon is known for recurrent networks trained on long sequences (Doya 94). It corresponds to the fact that a very small variation on parameters has a great influence on the signal many samples after. Therefore, RTRL was not tested further.

For the difficult test, results are reported in Table 2. Note that we are mainly interested in the relative performance between algorithms, because the absolute values depend on the intervals of initialization. For the sake of clarity, all parameters (step size, characteristics of the coupling link) are omitted here, but detailed in (Szilas 1995); they have been manually tuned in order to optimize the number of successful runs. For the long sequences, because the stopping criterion on the MRD is calculated on-line on the last 50 samples, it differs from the MRD calculated off-line on the first 50 samples; thus, after learning, we tested the MRD on the 20 nets, which gave us a different number of successful runs; this number is reported between brackets in Table 2.

		RTRL	BPTT	RC	SC
Short sequences	# of successful trials (out of 20)	9	15	15	14
	# of epochs	267	95	139	228
Long sequences	# of successful trials (out of 20)	Not tested (see text)	14 (15)	13 (14)	9(15)
	# of adaptations	Not tested (see text)	2596	16594	17872

<Table 2. Experimental results of the analysis algorithms, for the difficult test. In the case of short sequences, a 50 sample long simulation is repeted many time; in the case of long sequences, there is one single long sequence. Thus, an epoch represents 50 adaptations.>

Those experiments have validated the adaptation equations described above, for the BPTT, RC and SC algorithms, but RTRL cannot be used for long sequences. In this case of long sequences, BPTT also have difficulty for local convergence, whereas its results for the difficult test are quite good in term of speed of convergence, better than SC and RC.

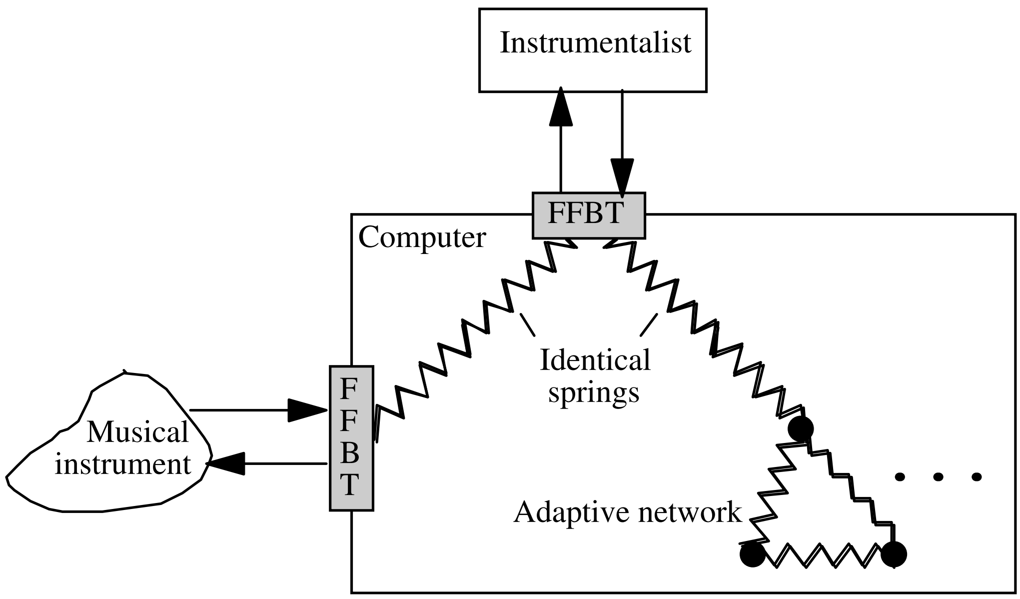
The fact that the coupling algorithms gives good experimental results is the most important result, because they constitute analysis techniques which share the two first features of the synthesis with the system we are based on: elementary modularity and bidirectionnality. Two explanations can be given for the good performance of coupling techniques. First, coupling limits the phase difference between the object to be analyzed and the model : this prevents the sensitivity parameters to explode. Second, when damping is high, the coupling element filters the high frequencies ; this simplifies the adaptation.

The following section discusses the two last features mentioned above, real-time handling and force-feedback.

Interactive Learning

Basic Principles of Interactive Learning for Physical Models

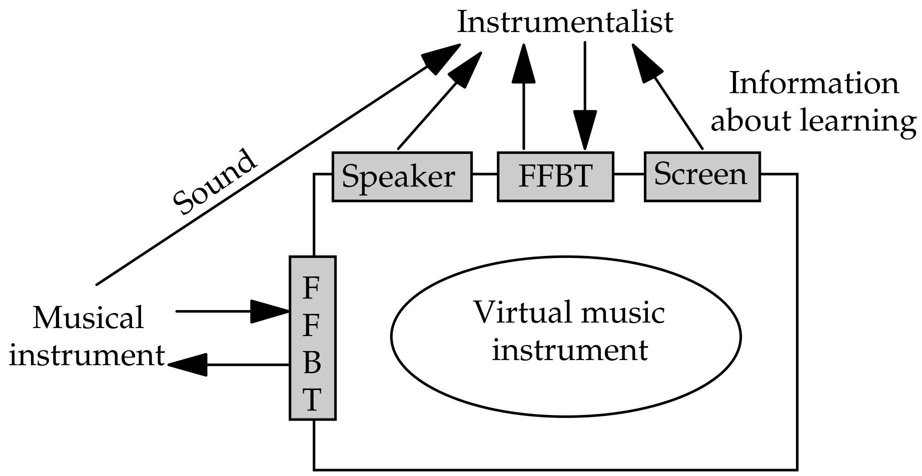
If we now intend to design an analysis system where the instrumentalist mechanically interacts in real-time with the simulated instrument, a system such as the one shown in Fig. 7 must be developed. This system involves two force-feedback transducers (FFBT): one between the instrumentalist and the computer, and one between the computer and the instrument to be analyzed. In this system, the instrumentalist interacts both with the instrument (via two transducers) and the model.



<Fig. 7. Interaction scheme between the instrumentalist, the instrument of reference and the model (the adaptive network). Two force-feedback transducers are necessary.>

In synthesis, the instrumentalist acts towards the virtual instrument according to musical and aesthetic criteria. Thus, in analysis, the action should also be based on those criteria, since the virtual instrument obtained by learning will be used in this way. However, the action of the instrumentalist could also be used as a way to facilitate learning. Indeed, learning algorithms based on gradient descent are subject to a couple of difficulties, like the problems of local minima or flat areas (Le Cun 87), which heavily diminish the chances of success. One of the empirical techniques to avoid those problems consists in modifying the data learned by the network, in order to achieve a progressive learning: start with the easy data and go on with the more difficult ones (Elman 1993; Cloete and Ludik 1993; Szilas and Ronco 1995). This teaching strategy is worth being applied in the context of physical modeling networks. We thus propose the action of the instrumentalist to be guided by both aesthetic and *pedagogical* criteria.

Note that whereas aesthetic action mainly needs haptic and sound feedback from the instrument, pedagogical action, or cognitive action, needs other information from the virtual instrument; this information is related to the progress of learning: how far are the real and the virtual instruments, how fast the parameters evolve, etc. The way such information would be displayed to the instrumentalist is an interesting human-computer interaction design problem. Figure 8 sketches out such a multimodal real-time interactive learning system.



<Fig. 8. Analysis as a multimodal real-time interactive system. Auditive, visual and haptic perception is involved.>

The fact that aesthetic and pedagogical components of the instrumentalist action come from different motives does not mean, however, that they are not of mutual interest. Indeed, some aesthetic effects like crescendi or pizzicati could be of pedagogical interest and, symmetrically, pedagogical action may be interesting from a musical point of view.

Experiments on Real-Time Coupling

In the following experiments, the object to be identified is not simulated but real, and the model is adapted in real-time under the human's hand excitation.

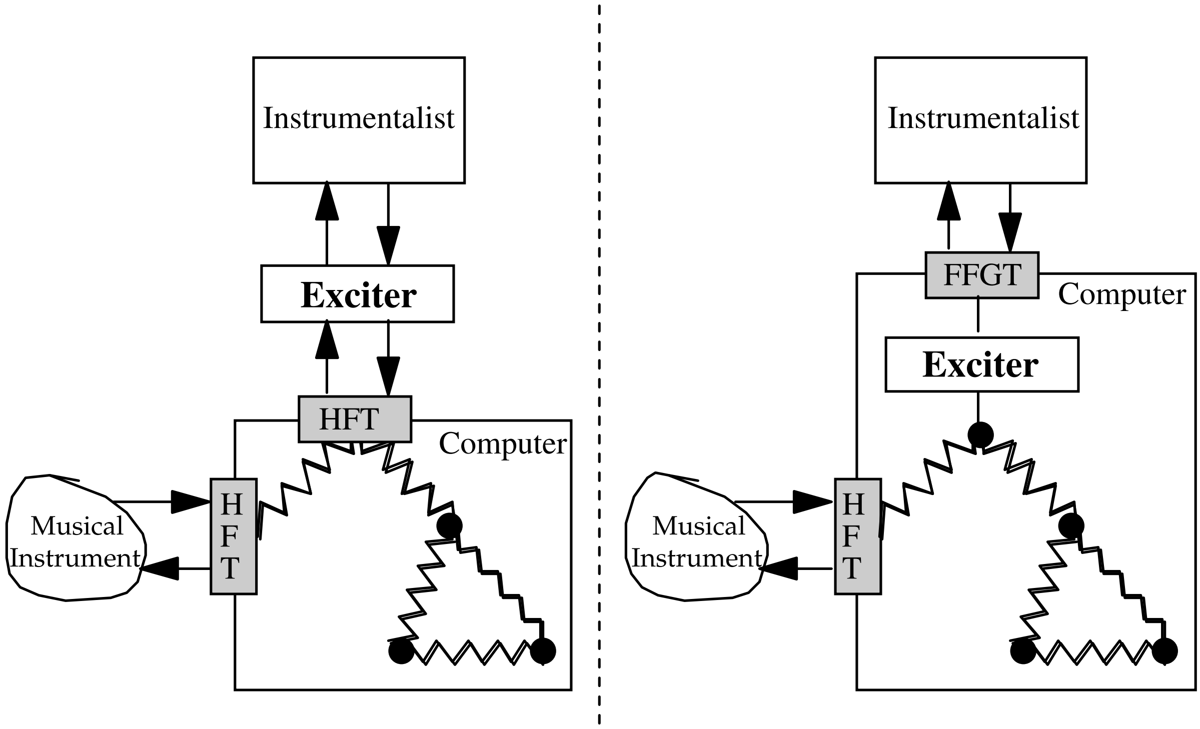
As for synthesis, we used force-feedback gestural transducers (FFGT) for the interaction between the model, the object and the operator. Because of the limited frequency of this device (1 kHz), and because of computational limitations, we limited those experiments to slow moving objects: Slow deformation behaviours replaced sound behaviours. Furthermore, because the FFGT is not designed for vibrating behaviours, the object was simply two rubber bands linked to a fixed stand. The corresponding model were a mass connected by a damped spring to a ground (an infinite mass). The experiment is shown on Fig. 9. Note that in fact, the object to be indentified is not composed of the rubber bands only, but also of the FFGT they are attached to.

<Fig. 9. Real-time coupling experimental device, from outside and inside, with the force-feedback gestural transducer (FFGT) developed at ACROE. The operator handles both the real and virtual objects, via the FFGT.>

The introduction of the operator into the identification process completely disturbs our experimental and validation techniques, because each human excitation is different, and results should be averaged through many trials. A first quantitative evaluation on six trials has shown that the difference between the model and the object can be highly reduced by the learning, in less than 50000 samples, that is less than 50 sec (see (Szilas 1995a) for details).

Besides, a qualitative evaluation has been performed. The operator compares the object and the model by handling them *separately* via a simulated damped spring. Before learning, one can clearly feel the difference between the model and the object, by gestural handling: Whereas the FFGT corresponding to the object is light but rather hard to move and quickly comes back to its initial position when moved away from it, the initial model is much easier to move and one can feel its inertia in the fingers. As a result, after learning, the difference does not seem perceptible, which is, at the end, our goal.

In the future, this system should be adapted for sound behaviour analysis, which has several implications. First, forty thousands adaptations per second must be performed, which actuates to simplify the algorithm and/or to optimize the programming. Second, instead of FFGT, high frequency transducers (HFT) must be used for the interaction between the musical instrument and the computer, as in (Caussé and Weinrach 1989). Between the computer and the performer, one can either have a HFT, which implies that he or she handles the excitater when he or she handles the device (as in (Boutillon and Guérard 1995), or a FFGT, which implies that the exciter is simulated (see Fig. 10). The second solution is preferred, because it allows to analyse a wider range of possible instruments, without changing the FFGT of the instrumentalist.



<Fig. 10. Two possibilities for the future system. The exciter is either real or simulated. >

Experiments on Progressive Learning

The aim of the following experiments is to show that the idea of pedagogical action proposed in Section 3.1 is feasible in the case of physical models.

In the experiments described in Section 2.2, only the Dirac excitation has been used (one single impulse at time 0). In fact, because this excitation reveals all the frequential components in a mechanical structure, it can be considered as the most difficult action.

Let us take this Dirac excitation as the target action, although we are aware that in the future, the target excitation should be based on the instrumentalist play.

Then, for finite length sequences, the following assumption can be made: the more impulses the signal contains, the easiest the action. This is only one assumption among others, and more frequency based algorithms could be tried, like lowpassing the Dirac excitation.

This leads to the following algorithm:

Begin

action ♦ Dirac excitation

adapt parameters for a N samples simulation

Repeat

If stagnation of MRD **Then** add an impulse to action

Else If MRD<desired-MRD **Then** action ♦ Dirac excitation

adapt parameters for a N samples simulation

Until MRD **<** desired-MRD **and** action = Dirac excitation

End

The stagnation occurs when MRD hardly changes from the previous calculation in the loop. Note that if the stagnation occurs several consecutive times, then the action signal contains more and more impulses: At the limit, it becomes a step function.

We applied the same evaluation protocol described in Section 2.2. As a result, the learning succeeded in 19 out of 20 trials, with 71 epochs in average. When compared with the results reported in Table 2, those results demonstrate the utility of pedagogical action.

This finding overcomes the field of Physical Models because such a pedagogical action could be used in other connectionist algorithms and more generally in identification algorithms.

4. Comparison between Analysis and Synthesis

As stated in the introduction, one of the aims of this article is a better understanding of the relation between the synthesis and the analysis, in the case of Physical Models. In particular, the following question is raised: up to which point analysis and synthesis could be symmetrical?

The proposed algorithms share the four main features of synthesis described in Section 1. However, analysis is computationally more costly than synthesis, on the one hand in term of algorithmic complexity per time step, in case of RTRL or storage in case of BPTT, and on the other hand in term of number of time steps needed for convergence. The fact that other analysis techniques have the same computational cost than their corresponding synthesis technique could let us believe that we missed a simpler analysis method. However, it appears that this difference is a direct consequence of the physical modeling approach.

Indeed, signal based sound synthesis techniques (e.g. additive synthesis) use elementary components that are signal themselves, should it be sinusoidal harmonics, wavelets, etc. So there is a direct mapping between the sound and the model basic parameters, all being based on a signal representation. For Physical Model approach however, the model represents the cause of the signal. Thus there is a certain distance between the model and the sound which implies additional processing during synthesis, based on mechanical calculations. For analysis, this additional processing must be inverted. This inversion is complicated because the physical models representation is distributed: in most cases, one can not assign a given behavior to a single parameters but to many of them. Thus, an iterative search through all the parameter space must take place, that is performed in our case thanks to the gradient information. In machine learning, this problem is called the "credit assignment problem" (@@), which is known as a difficult problem. Thus, analysis cannot be symmetrical to synthesis, in the case of Physical Models.

Note that if one deduces the modal representation from a given mechanical network (Djoharian 1993), which consists in a bank of elementary mass-spring oscillators encapsulated into two matrices multiplications, then the analysis becomes much easier, because the representation is no longer distributed (Szilas 1995).

In the case of waveguides models (Smith 1992), low cost analysis is possible (Välimäki, Karjalainen and Huopaniemi 1995) because the model does not explicitly handle the causal relations between physical components but is based on a signal representation, the waveguide. For more complex waveguides models, made up with elementary waveguides (Van Duyne and Smith 1993), the analysis, if it were performed, would be much more costly than synthesis, as in mass-spring models.

In Table 3, the comparison between synthesis and analysis in Physical Models is summed up. This table holds for any sound techniques based on the computing of the interaction of various physically based components.

	Complexity (n D.O.F.)	Principles of calculation	Signals propagated	Type of processing	Type of action
Synthesis	n	fundamental principle of Dynamics and law of damped springs	forces and positions	deduction	aesthetic
Analysis	n^2	gradient descent	gradient information	induction	pedagogical

<Table 3. Comparison of synthesis and analysis in mass-spring models>

Conclusion

The possibility of deriving an analysis algorithm for mass-spring networks has been demonstrated here, both theoretically and experimentally. The proposed algorithms allow the analysis of linear mechanical structures; the extension to nonlinear structures is straightforward, if all functions are differentiable.

Beyond this algorithmic side, most important is how interactions between the model, the instrument of reference and the instrumentalist should be taken into account in analysis: analysis is not considered as a single automatic tool for constructing a model but as an interactive and real-time simulation, where the role of the instrumentalist is central.

However, one has still got a long way to go, before such a system being implemented, and this paper only lays the theoretical and experimental foundations of it: on-line learning, coupling, instrumentalist action, progressive learning and multimodal human-computer interaction. If simpler (not fully connected) networks are chosen, a speed up of four times is sufficient to perform the calculation at 40 kHz. Some improvements are also necessary in the transducers: HFT and FFGT must be combined (see Fig. 10), and their noise has to be highly reduced.

The original view of analysis developed in this study raises some sides of connectionist learning that are hardly tackled so far: cooperative learning on the one hand, progressive active learning on the other hand. This is one more illustration of how musical issues can have some repercussions on a much broader set of disciplines.

References

Borin, G., G. De Poli, and A. Sarti. 1992. "Algorithms and Structures for Synthesis Using Physical Models." *Computer Music Journal*, 16(4): 30-42.

Boutillon, X. and J. Guérard. 1995. "Hybrid Synthesis." In *Proceedings of the International Symposium on Musical Acoustics*, pp. 440-445, Dourdan, France.

Cadoz, C. 1990. "Simuler pour connaître / Connaître pour simuler - Réflexions sur la représentation, la Modélisation, la Simulation et la Création avec l'Ordinateur." In *proceedings of the colloque Modèles physiques création musicale et ordinateur*. Paris: Ed. de la maison des sciences de l'homme, vol. III, pp. 661-708.

Cadoz, C., L. Lisowski, and J.-L. Florens. 1990. "A modular Feedback Keyboard Design." *Computer Music Journal* 14(2): 47-51.

Cadoz, C., A. Luciani, and J.-L. Florens. 1993. "CORDIS-ANIMA: a modelling and simulation system for sound and image sysnthesis - the general formalism." *Computer Music Journal* 17(1): 19-29.

Cadoz, C., A. Luciani, and J.-L. Florens. 1981. "Synthèse musicale par simulation des mécanismes instrumentaux. Transducteurs Gestuels Rétroactifs pour l'étude du jeu instrumental." *Revue d'acoustique* 59: 279-292.

Caussé, R., and G. Weinrach. 1989. "Simulation and Further Experiments With Digital Bow." In *Proceedings of the 13° International Congress on Acoustics (ICA'89)*, Vol. III, pp. 83-86, Belgrade.

Cloete, I. and J. Ludik. 1993. "Increased Complexity Training." In *Proceedings: New trends in neural computation (IWANN'93)*, Lecture Notes in computer science n°686, Springer Verlag.

Djoharian, P. 1993. "Generating Models for Modal Synthesis." *Computer Music Journal* 17(1): 57-65.

Doya, K. 1994. "Bifurcations of Recurrent neural networks in Gradient Descent Learning". Submitted to *IEEE Transactions on neural networks*.

Elman, J.L. 1993. "Learning and development: the importance of starting small." *Cognition* 48: 71-99.

Le Cun, Y. 1987. "Modèles connexionistes de l'apprentissage." PhD thesis, Université Paris VI.

Piché, S.W. 1994. "Steepest Descent Algorithms for Neural Network Controllers and Filters". *IEEE Trans. on Neural Networks*, 5(2): 198-212.

Roads, C. 1994. "Physical modeling." *Keyboard* 20(9): 89-94 & 100-103.

Rumelhart, D.E., and J.L. McClelland (Eds.). 1986. *Parallel Distributed Processing,* volume 1. Cambridge, Mass.: MIT Press.

Smith, J.O. 1992. "Physical Modeling Using Digital Waveguides." *Computer Music Journal* 16(4): 74-87.

Szilas, N. 1995. "Apprentissage dans les réseaux récurents pour la modélisation mécanique et étude de leurs interactions avec l'environnement." PhD thesis, Institut National Polytechnique de Grenoble, France.

Szilas, N., and C. Cadoz. 1993. "Physical Models that learn." In *Proceedings of the 1993 International Computer Music Conference,* San Francsico: International Computer Music Association, pp. 72-75.

Szilas, N., and E. Ronco. 1995. "Action for learning in non-symbolic systems." In *Proceedings of the European Conference on Cognitive Science*, pp. 55-63, Saint-Malo, France.

Välimäki, V., M. Karjalainen and J. Huopaniemi 1995. "Measurement, estimation, and modeling of acoustic tubes using DSP techniques." In *Proceedings of the International Congress on Acoustics (ICA'95)*, Vol. III, pp. 513-516, Trondheim (Norway).

Van Duyne, S.A., and J.O. Smith. 1993. "Physical Modeling with the 2-D Digital Waveguide Mesh." In *Proceedings of the 1993 International Computer Music Conference,* San Francsico: International Computer Music Association, pp. 40-47.

Williams, R.J., and J. Peng. 1990. "An efficient gradient-based algorithm for on-line training of recurrent network trajectories." *Neural Computation* 2: 490-501.

Williams, R.J., and D. Zipser. 1989. "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks." *Neural Computation*, 1: 270-280.

Figure captions

1: Synthesis.

2: Analysis.

3: Mechanical and CORDIS-ANIMA automata representations of a mass-spring network.

4: Illustration of the gradient descent algorithm; $p=(p_1,p_2)$ are the parameters and $E(p)$ the error to be minimized. Gradient descent consists in following the steepest direction.

5: Mechanical interpretation of RTRL: the equations for adapting the parameters can be interpreted as a set of mass-spring models.

6: Mechanical interpretation of the coupling algorithm. The additional networks are now attached to the ground.

7: Interaction scheme between the instrumentalist, the instrument of reference and the model (the adaptive network). Two force-feedback transducers are necessary.

8: Analysis as a multimodal real-time interactive system. Auditive, visual and haptic perception is involved.

9: Real-time coupling experimental device, from outside and inside, with the force-feedback gestural transducer (FFGT) developed at ACROE. The operator handles both the real and virtual objects, via the FFGT.

10: Two possibilities for the future system. The exciter is either real or simulated.