Adaptive Networks for Physical Modeling

Nicolas SZILAS and Claude CADOZ

ACROE-CLIPS-IMAG LEIBNIZ-IMAG
INPG, 46 avenue Félix Viallet, 38000 Grenoble, France
Tel.: (33) 76 57 47 84
Fax: (33) 76 57 46 02
Email: Nicolas.Szilas@imag.fr

Abstract

This paper presents an original link between neural networks theory and mechanical modeling networks. The problem is to find the parameters characterizing mechanical structures in order to reproduce given mechanical behaviors. Replacing "neural" units with mechanically based units and applying classical learning algorithms dedicated to supervised dynamic networks to these mechanical networks allows us to find the parameters for a physical model. Some new variants of RTRL (Real-Time Recurrent Learning) are also introduced, based on mechanical principles.

The notion of interaction during learning is discussed at length and the results of tests are presented. Instead of the classical {machine learning system, environment} pair, we propose to study the {machine learning system, human operator, environment} triplet.

Experiments have been carried out in simulated scenarios and some original experiments with a force-feedback device are also described.

Keywords. Recurrent networks; physical modeling; active learning.

1 An original approach in modeling

We are concerned with solving the identification problem for unknown mechanical structures. Usually, mechanical system identification is performed using two separate classes of techniques: "parametric" or "non-parametric" estimation [19]. In the first case, we have a physical model of the structure, but the parameters are unknown. Thus, we use identification techniques to establish these parameters. But in many cases, the physical model is not known in advance, and thus the non-parametric techniques are appropriate as they allow us to build a model of the structure. Neural networks are usually used as non-parametric identifiers [14, 19]. But for non-parametric techniques, the resulting model is often far from a description of the physical nature of the identified system: for example it could be Fourier coefficients or "synaptic" weights, which do not provide information concerning stiffness, damping, etc.. Such a model is often described as a "black box".

The general idea that we would like to introduce in this paper is as follows: instead of using "neural" units that apply a non-linear function to the sum of the inputs [15, 13], could it be possible to have a mechanically based unit, i.e. a unit that *represents* a material entity? A network with such units would thus represent a material object and not a black box.

Indeed, physical modeling networks do exist, and in our work we make use of networks of masses linked with damped springs (see *Fig. 1*) [7]. These nets are composed of specific units (see next section) and have been shown to be capable of modeling any linear mechanical system when represented as a weighted sum of sinusoidal oscillators [9]. Such networks are used both for simulating vibrating structures, like musical instruments [7], or three-dimensional objects, like robots and soils [16].

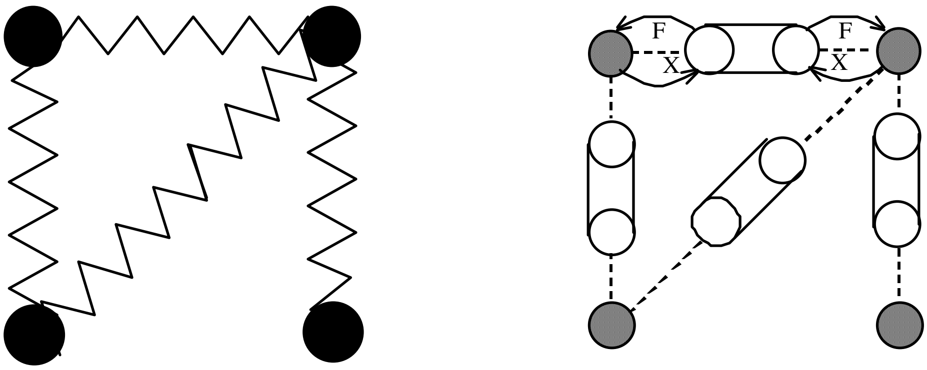

Fig. 1. Mechanical and network representations of a physical modeling network

Our short-term goal is to add some *adaptive* properties to these networks [17] and thus to close the gap between physical non-adaptive networks and non-physical adaptive networks. As a part of this work, we have applied neural network methods (as defined in [15]) to the problem of parametric identification.

Our long-term goal concerns the identification of musical instruments: even if we know the geometrical and physical properties of a musical instrument, it is quite difficult to deduce the physical parameters of a modeling network, and an adaptive system would solve this problem. However, the approach proposed here is quite general, and could be applied to identification of soils, in robotics, of animated structures [10], etc..

The originality of this approach is principally that we put at the same level both the physical representation and the learning mechanisms, whereas the classical approach considers the learning as a higher level cognitive function than that of the physical and analogical representation.

2 Application of Real-Time Recurrent Learning (RTRL) to physical modeling

The graph of mechanical networks, that depends on the simulated object (a cord, a membrane, etc.) is never unidirectional, because automata are always connected in both direction (see *Fig. 1*). Thus, we are interested in recurrent networks. Furthermore, as for any identification problem, the network is given one or several pairs of input-output sequences, and must adapt its parameters in order to perform the input-output mapping. Thus, two families of algorithms are interesting for our purpose: the Back-Propagation Through Time [5, 13] and the Real-Time Recurrent Learning algorithm [2]. Because the sequences which are processed are very long (for example more than 10000 samples) and because we want to perform a high speed real-time identification (1000 adaptations per second), we chose the RTRL algorithm (although some experiments with BPTT have been carried out, but are presented elsewhere [7]). Because of this speed constraint we also did not use sophisticated improvements, like second order methods; we kept the algorithm as simple as possible.

We are now going to describe the physical modeling networks. There are two types of automata, punctual masses and link elements, that exchange two types of signals: forces and positions. The propagation formulas are established by discretizing the laws of mechanics:

- a punctual mass is an automaton receiving forces and yielding one position, according to the fundamental principle of dynamics:

$$X_i(n) = 2X_i(n-1) - X_i(n-2) + \frac{1}{m_i}\Big[Fe_i(n-1) + \sum_{j\in C(i)} F_{ji}(n-1)\Big] \tag{1}$$

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 $Fe_i(n-1)$ the external force applied to the mass i.

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

$$F_{ij}(n) = k_{ij}\big(X_i(n)- X_j(n)\big) + z_{ij}\Big[\big(X_i(n)- X_i(n-1)\big)-\big(X_j(n)- X_j(n-1)\big)\Big] = -F_{ji}(n) \tag{2}$$

where k_{ij} is the stiffness of the spring between the masses i and j, and z_{ij} its damping factor.

Note that in this case, the evolution of the network is one-dimensional, instead of three-dimensional, because we are interested in vibrating structures.

Let $Xd_i(n)$ denote the desired position of the mass i at time n, and A the set of indices where this desired behavior is known.

Then define the error E_{tot} between the actual behavior and the desired behavior by:

$$E_{tot} = \sum_{n=1}^{N} E(n) = \sum_{n=1}^{N} \sum_{i\in A} e_i(n)$$

where $e_i(n) = [X_i(n) - Xd_i(n)]^2$ if $i\in A$, $e_i(n) = 0$ if $i\notin A$ and N is the total length of the simulation.

The parameters, that is masses m, stiffness and damping factors k and z are adapted according to the gradient descent algorithm:

$$\Delta m_i^{-1} = -\alpha \frac{\partial E_{tot}}{\partial m_i^{-1}} = \sum_{n=1}^{N} \Delta m_i^{-1}(n), \text{ with } \Delta m_i^{-1}(n) = -\alpha\frac{\partial E(n)}{\partial m_i^{-1}} \tag{3}$$

$$\Delta k_{ij} = -\beta \frac{\partial E_{tot}}{\partial k_{ij}} = \sum_{n=1}^{N} \Delta k_{ij}(n), \text{ with } \Delta k_{ij}(n) = -\beta\frac{\partial E(n)}{\partial k_{ij}} \tag{4}$$

$$\Delta z_{ij} = -\mu \frac{\partial E_{tot}}{\partial z_{ij}} = \sum_{n=1}^{N} \Delta z_{ij}(n), \text{ with } \Delta z_{ij}(n) = -\mu\frac{\partial E(n)}{\partial z_{ij}} \tag{5}$$

As for RTRL, one needs to define additional quantities, which represent the sensitivity of each force and position to the parameters of the network:

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

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

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

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

$$S_i^l(n) = 2S_i^l(n-1) - S_i^l(n-2) + \frac{1}{m_l}\sum_{j \in C(l)} R_i^{jl}(n-1) + \delta_{il}\sum_{j \in C(l)} F_{jl}(n-1) \tag{6}$$

$$R_i^{jl}(n) = (k_{jl}+z_{jl})(S_i^j(n) - S_i^l(n)) - z_{jl}(S_i^j(n-1) - S_i^l(n-1)) \tag{7}$$

$$\sigma_{ij}^l(n) = 2\sigma_{ij}^l(n-1) - \sigma_{ij}^l(n-2) + \frac{1}{m_l}\sum_{h \in C(l)} \rho_{ij}^{hl}(n-1) \tag{8}$$

$$\begin{aligned}\rho_{ij}^{hl}(n) = {} & (k_{hl}+z_{hl})(\sigma_{ij}^h(n) - \sigma_{ij}^l(n)) - z_{hl}(\sigma_{ij}^h(n-1) - \sigma_{ij}^l(n-1)) \\ & + \delta_{\{i,j\}\{h,l\}}(X_i(n) - X_j(n))\end{aligned} \tag{9}$$

$$\tau_{ij}^l(n) = 2\tau_{ij}^l(n-1) - \tau_{ij}^l(n-2) + \frac{1}{m_l}\sum_{h \in C(l)} \chi_{ij}^{hl}(n-1) \tag{10}$$

$$\begin{aligned}\chi_{ij}^{hl}(n) = {} & (k_{hl}+z_{hl})(\tau_{ij}^h(n) - \tau_{ij}^l(n)) - z_{hl}(\tau_{ij}^h(n-1) - \tau_{ij}^l(n-1)) \\ & + \delta_{\{i,j\}\{h,l\}}\left[(X_i(n)-X_i(n-1)) - (X_j(n)-X_j(n-1))\right]\end{aligned} \tag{11}$$

where δ is the Kronecker symbol.

One finally can calculate:

$$\Delta m_i^{-1}(n) = 2\sum_{l \in A}(X_l(n)-Xd_l(n))S_i^l(n) \tag{12}$$

$$\Delta k_{ij}(n) = 2\sum_{l \in A}(X_l(n)-Xd_l(n))\sigma_{ij}^l(n) \tag{13}$$

$$\Delta z_{ij}(n) = 2\sum_{l \in A}(X_l(n)-Xd_l(n))\tau_{ij}^l(n) \tag{14}$$

According to Eqs. (3) to (5), one should sum the •$m_i^{-1}(n)$, •$k_{ij}(n)$ and •$z_{ij}(n)$ over time and perform the adaptation at the end of the sequence. But for on-line learning, this is no longer the case: The parameters must be adapted at each time step or periodically, which prevents from following the true gradient direction [2, 5, 14]. In our case, we chose to adapt at each time step.

The RTRL algorithm is described by Eqs. (1) and (2) for single propagation, and Eqs. (6) to (14) for learning.

An interesting point of these formulas is that they can be physically interpreted. Eqs. (6) and (7) are very similar to (1) and (2), which describe the propagation of the activities. In fact, 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. (8) to (11). Thus, each adaptive parameter has its associated network, which is similar to the original one, excited in one unit, and one signal of which is used for adaptation (see *Fig. 2*). This interpretation is interesting because it shows that learning involves the same kind of processing as propagation: the learning network is entirely expressed in term of units and connections. 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. Note that this is also true for the classical RTRL, with the difference that, because our activation functions are linear, one term disappears in Eqs. (6) to (11): this term would amount to add connections between the (X , F) and (S , R) networks at *each* node.

At last, it is noteworthy that physically based units can be used to perform the adaptation: propagation and adaptation are described in a homogeneous way.

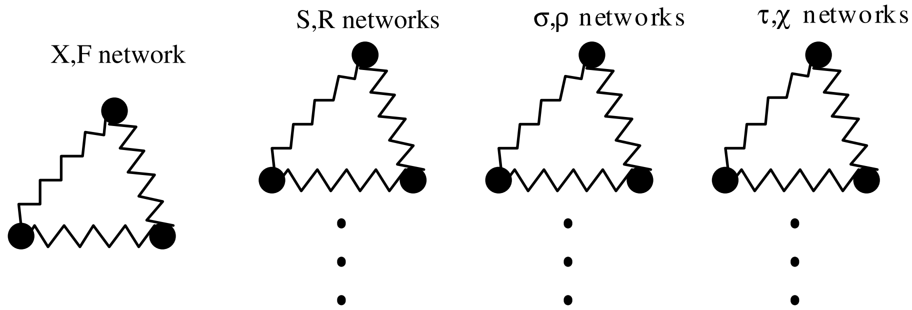

Fig. 2. Physical interpretation of the RTRL algorithm (inter-networks connections are not represented)

3 About interactions

In the physical world (as for many other situations), there does not exist any unidirectional communication: there are only interactions. At ACROE this principle of interactivity has been intensively developed for several years, not only inside the model, but also between the model and the physical environment. Physical models can be manipulated in real-time by the operator, not with a mouse or any unidirectional of communication tool *but* with transducers that can transmit both the gesture from the operator to the model and the forces from the model to the operator (we call them Force-Feedback Gestural Transducers: *FFGT*). This force-feedback is essential for the fine control of the physical models; it is therefore used in music performance, realistic computer animation, robotics or multimodal human-computer interaction. In our system, there are three entities that can communicate interactively: the structure to be modeled (the object), the operator and the simulated model itself (the network), as illustrated in *Fig. 3*.

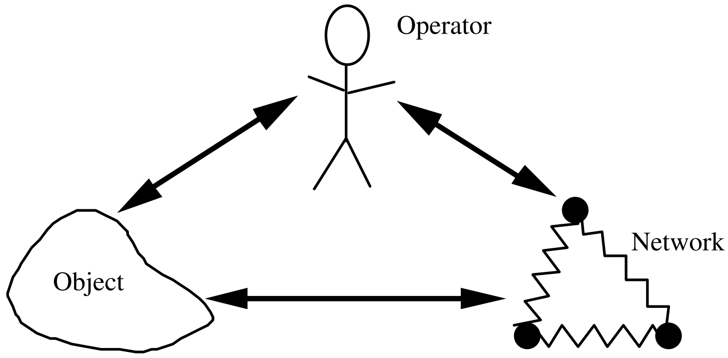

Fig. 3. Interactions during learning

3.1 Network-object interaction

In our formalism of physical objects description, interactions are performed through the *link elements*, the damped springs, which connect two masses. The idea for interactive learning in physical models is consequently to put a *coupling spring* between the object and the model network. To allow the communication between this (virtual) spring and the (real) object, a *FFGT* is inserted. *Fig. 4* illustrates the whole system, where the dashed line represents the interface between the simulated world (masses, springs) and the physical environment.

Before further developing the consequence of such a coupling, we would like to compare this situation with the classical "teacher forcing" techniques [2, 8, 14], whose principle is to replace the actual state of the network by the desired state. In [8], "teacher forcing" is compared with a father guiding his son, who learns how to bike, by holding the bicycle. But this analogy is wrong, since the father is not a rigid machine: the movements of the learning child interact with the father's arms, whose movements are modified, and it can be supposed that this interaction helps learning. Thus, the interaction scheme that we propose may exist in natural learning. However, we are far from really modeling the interactive human motor teaching.

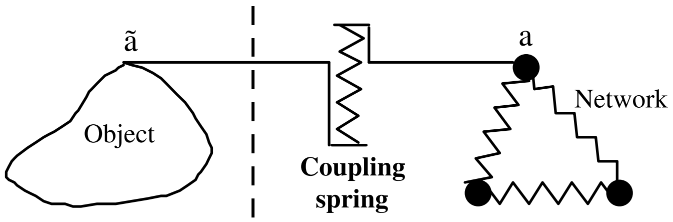

Fig. 4. Interaction between an object to be identified and the adaptive model

Let us now derive the mathematical consequences of coupling the object and its network model.

First, to derive the Eqs. (12) (13) and (14), we have considered that the desired signal is independent of the network parameters. But if the network influences the object to be modeled, this is no longer the case. For example, in the case of masses adaptation, one has:

$$\Delta m_i^{-1}(n) = \sum_{l \in A} \frac{\partial (X_l(n)-Xd_l(n))^2}{\partial m_i^{-1}} = 2\sum_{l \in A} (X_l(n)-Xd_l(n)) \left[S_i^l(n) - \frac{\partial Xd_l(n)}{\partial m_i^{-1}} \right]$$

The last term not only has a non-zero value, but cannot be calculated, because it depends on the object itself, which is unknown by definition. This problem is evoked in [15], but no solution is proposed. We simply put this last term to zero, thus keeping equations (12) (13) and (14) unchanged, which is an approximation.

Second, as there is a new damped spring in the system, the derivatives of the force produced by this spring according to all the adaptive parameters in the network must be calculated, and used in Eqs. (6), (8) and (10). We define, for all a A, the index $\tilde{a}$, which refers to the access point linked to the mass a by the corresponding coupling spring (see *Fig. 4*). Thus, we have $\tilde{a}$ $C(a)$. It can easily be found that :

$$R_i^{\tilde{a}a}(n) = \frac{\partial F_{\tilde{a}a}(n)}{\partial m_i^{-1}} = -(Kc_a+Zc_a)S_i^a(n) + Zc_a S_i^a(n-1) \tag{15}$$

$$\rho^{\tilde{a}a}_{ij}(n) = \frac{\partial F_{\tilde{a}a}(n)}{\partial k_{ij}} = -(Kc_a+Zc_a)\sigma^{a}_{ij}(n) + Zc_a\sigma^{a}_{ij}(n\text{-}1) \tag{16}$$

$$\chi^{\tilde{a}a}_{ij}(n) = \frac{\partial F_{\tilde{a}a}(n)}{\partial z_{ij}} = -(Kc_a+Zc_a)\tau^{a}_{ij}(n) + Zc_a\chi^{a}_{ij}(n\text{-}1) \tag{17}$$

This coupling algorithm is thus described by the same learning equations as RTRL, i.e. Eqs. (6) to (14), plus Eqs. (15) to (17). Furthermore, Eq. (2) is applied to the coupling spring and in Eq. (1), the point $\tilde{i}$ is added to *C(i)* (and the point *i* is added to $C(\tilde{i})$ when the object is simulated).

We give the simple following physical interpretation of these three equations: each additional network we mentioned in Section 2 is attached to the soil at the masses *a* *A*, thanks to a spring with a stiffness *Kc* and a damping factor *Zc* (see *Fig. 5*).

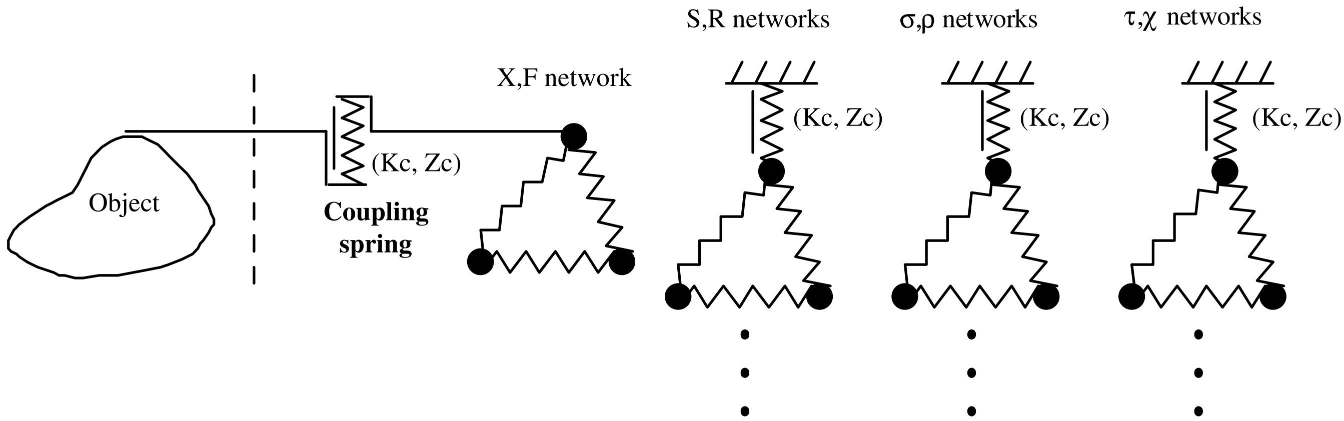

Fig. 5. Physical network for adaptation in the coupling case

Because the action towards the environment implies a truncation in the adaptation formulas, we also propose a variant of coupling, which we call the *semi-coupling*. In this case, which is no longer physically based, the spring between the model and the object sends forces to the model only, not to the object. The equations are the same as in the real coupling case, except that the model sends no force to the object (and when the object is simulated, $i \notin C(\tilde{i})$).

For comparison, we have also implemented teacher forcing, a variant of RTRL which consists, for every *a* *A*, in setting $X_a(n)$ equal to $Xd_a(n)$, and $S^{i}_{a}(n)$ to zero (for all *i*) after all adaptations at time *n* are performed. Note that semi-coupling shares some features with the variant of teacher forcing proposed in [8].

As for the teacher forcing techniques, it is not demonstrated that coupling and semi-coupling work and improve the convergence. In particular, even if we know that if the error between the actual and desired signals during learning is exactly zero, the model is reproducing the object, a small difference between the two signals *during learning* can correspond to a high error, when the teacher forcing or the (semi) coupling is removed.

3.2 Interaction with the operator

Usually, in system identification two components are present: the system to be identified and the computer. We introduce here a third one: the operator. This is justified by the fact that we are interested in physical structures as objects *handled by the operator*. Furthermore we believe that the human touch has its importance in learning. A consequence of this approach is that the computer is not autonomous for acting on the physical objects because the human operator intervenes.

As previously, this interaction is performed by simulated springs. The scheme of interaction is depicted in *Fig. 6*. Two *FFGT* are needed: one for handling the network, and one for handling the object; the two *FFGT* are linked with a simulated spring so that the action from the human is equally distributed in the physical object and the model. Note that in this case, the term $Fe_i(n)$ of Eq. (1) is removed.

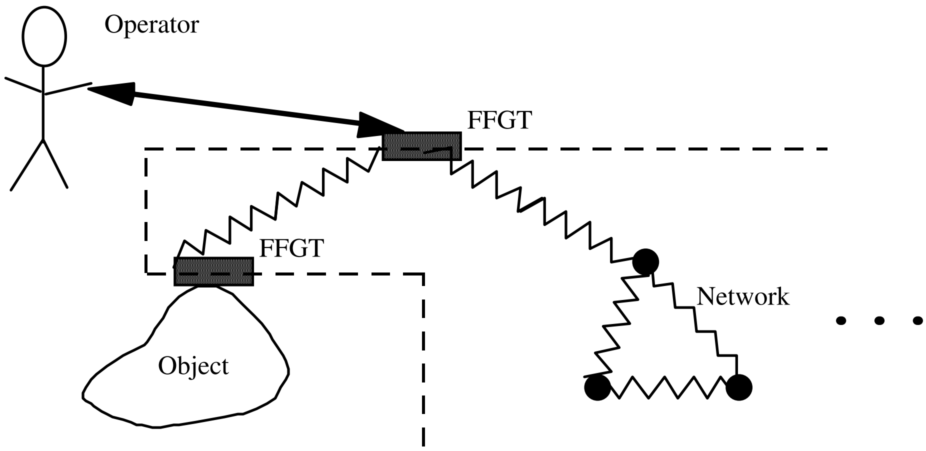

Fig. 6. Interaction scheme between the operator, the object to be identified and the model (the adaptive network)

The idea about the action of the operator is that it can play a useful role in learning, by grading the difficulty of the learning task [1]. This leads us to shortly review the concept of active learning in neural networks (for an exhaustive review, see [1, 6]).

In fact, active learning is generally neglected in neural networks which are usually fed with random samples from a training set. Some studies however consider the way of selecting examples within a training set [18] or by queries [20] in order to speed up learning. In this case, the most informative patterns are selected. However, an alternative technique would consist in selecting patterns in order to begin with a simple problem and gradually raise the difficulty. Such a progressive learning has been studied in the case when an a priori criterion for estimating the difficulty does exist [11], but general algorithms are still unknown.

Therefore, back to our identification problem, we realize that the action for progressive learning could be performed by the operator, instead of being automatic. Thus, we would use the cognitive skill of human to facilitate learning.

Hence, an additional interaction takes place, besides the simple physical one: it occurs at a higher level, with the human choosing according to his intelligent capacities the movement towards both the object and the model, in order to grade the difficulty. The information that he or she uses is first of all mechanical (he or she physically feels, in its hands, what happens), but it can be also a visual or sound information, or even numeric information about the learning in progress.

4 Experiments

4.1 Fixed length simulations

In a first stage, the purpose is to systematically try out the algorithms that have been proposed above. Thus, we have limited ourselves to *simulated* scenarios, where the physical object is simulated with the same kind of network as the model. This unusual choice in Neural Networks (it consists of using a neural network for the identification of a neural network) allows us to control the task: if the two networks (the learning one and the static one) are chosen with the same topology, then we know that complete identification is possible. All the experiments reported in this section belong to this category. Furthermore, the networks are limited to five masses (including the soil, which is an infinite mass) entirely connected, and the set *A* is a singleton *{a}*, i.e., there is only one mass in the network where a desired signal is provided. Both networks receive a single unit external force at time *0* and the length of the simulation (*N*) is 50. In order to interpret the similarity between the actual and desired behaviors, we choose a criterion different from 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:

$$\mathrm{MRD} = \frac{\sum_{n=1}^{N} \left| X_a(n) - Xd_a(n) \right|}{N\left(\max_{n=1,\dots,N} \{X(n)\} - \min_{n=1,\dots,N} \{X(n)\} \right)}$$

There are three types of parameters to be adapted: masses *m* (more precisely inverted masses *1/m*), stiffness and damping factors *k* and *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 into a certain interval, then training the network until the *MRD* goes below a given threshold. Ideally, the two values become equal. Note that, coarsely speaking, the wider the intervals of parameters, the harder the task.

For teacher forcing and 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. Other studies about teacher forcing mention this point [12]; in order to go beyond this acknowledgment, we propose to overtrain the network until the desired *MRD* is reached. 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.

Four algorithms are tested: RTRL, teacher forcing (TF), semi-coupling (SC) and real coupling (RC). Two series of experiments have been carried out.

First, we had to check that the algorithm is able reach a minimum of the error surface, when the network's initial state is near this minimum (*local convergence*). Thus, two networks with close initial states must perfectly fit after learning. Such a local convergence is mathematically demonstrated for the basic off-line RTRL only, hence the necessity of these experiments. Note that we are only interested in the fraction of successful cases over the trials; if this fraction is *100 %*, then the local convergence test is passed.

Second, we compared the four algorithms on a more realistic task, which we call *difficult task*. The reference is the performance of RTRL: we choose a problem that is badly solved by RTRL, and examine the performance of the other algorithms.

For all experiments, twenty pairs {learning net, static net} are tested. 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).

Note that the quantitative comparison is slightly distorted by the fact that some parameters are arbitrarily set up for each experiment by trial and error (there is no guarantee that the choice is optimal): learning step, stiffness and damping of the coupling, and the actual threshold on *MRD*.

Local convergence

We chose the following intervals for parameters:

	Minimum	Maximum
Mass	1.5	1.7
Stiffness	0.09	0.11
Damping	0	0.05

Table 1. Choice of parameters interval for local convergence

For these intervals, the mean of *MRD* over the 20 cases *before* learning is 5 10^{-2}. The threshold on *MRD* has been set up to 10^{-2}. It visually gives a very good fitting between the two curves (see *Fig. 7*). The results are summarized in *Table 2*.

Before learning: *MRD* = 4.5 10^{-2} After 54 epochs: *MRD* = 0.98 10^{-2}

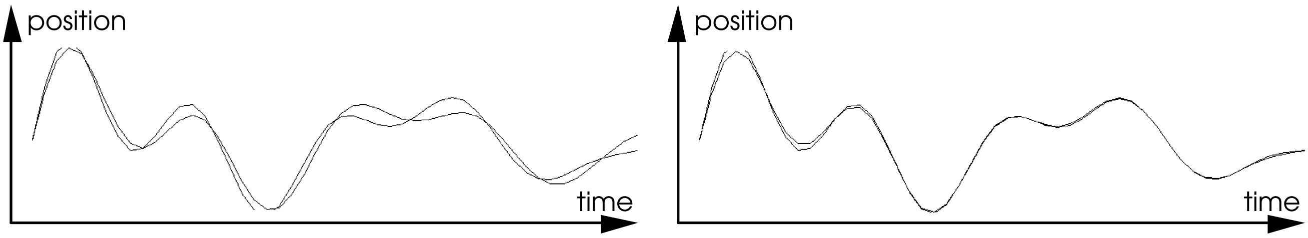

Fig. 7. An example of the temporal sequences for the local convergence test

Algo.	Step size	Kc	Zc	Actual threshold on *MRD* (x 100)	number of successes	Mean of real *MRD* on successful trials (x 100)	ımber of epochs			
							Min.	Max.	Mean	Std. dev.
RTRL	2 10^{-5}	-	-	-	20	-	14	203	92	51
SC	2 10^{-5}	0.1	0	0.7	20	0.99	2	213	59	61
RC	2 10^{-5}	0.1	0	0.65	20	0.99	22	760	313	210
TF	2 10^{-5}	-	-	1	20	5	1	1	1	0
TF	10^{-3}	-	-	0.2	4	2.2	20	302	138	103
TF	10^{-3}	-	-	0.1	0	-	-	-	-	-

Table 2. Results for the local convergence test

As a result, all the algorithms passed the test of local convergence, except teacher forcing. Indeed, although you read 20 successes out of 20 for the first experiment with teacher forcing, the mean of *real MRD* is 5 10^{-2}, which means that during simulation, *MRD* was below 10^{-2}, but the *real MRD*, estimated without teacher forcing was much higher. If the threshold on *MRD* is diminished, the rate of convergence is low (4 out of 20) and the mean of real *MRD* still higher than 10^{-2}. We have carried out even easier local convergence tests, but the teacher forcing failed again.

However, it must be pointed out that the comparison may be unfair, because some variants of teacher forcing, proposed in [8], have not been tested.

Difficult task

Now the masses are set equal to 1, the damping factors to zero, while the stiffness is in the interval [0.05,0.3]. Only the stiffness is adaptive. The mean of *MRD* over the 50 cases *before* learning is 24.6 10^{-2} and the threshold on *MRD* is 5 10^{-2}. This value gives a good fitting between the two curves (see *Fig. 8*). The results are summarized in *Table 3*.

Before learning: *MRD* = 22 10^{-2} After 355 epochs: *MRD* = 4.97 10^{-2}

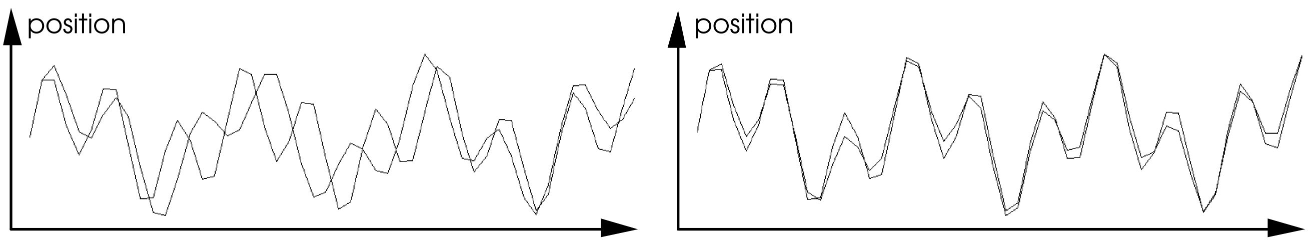

Fig. 8. An example of the temporal sequences for the "difficult task"

Algo.	Step size	Kc	Zc	Actual threshold on *MRD* (x 100)	number of successes	Mean of real *MRD* on successful trials (x 100)	ımber of epochs			
							min.	Max.	Mean	Std. dev.
RTRL	10^{-5}	-	-	-	19	-	43	815	246	188
SC	$2\ 10^{-4}$	0.05	0.5	1.5	29	4.36	8	846	211	227
RC	10^{-3}	0.3	0.6	0.58	38	4.16	10	992	169	203

Table 3. Results for the "difficult task"

Once again, teacher forcing failed ; results are not reported here. The main result is that semi-coupling and real coupling perform well. This is of particular interest for the real coupling algorithm, whose equations have been truncated (see Section 3.1). The results are significatively better than RTRL : semi-coupling succeeds 29 times out of 50, against 19 times for RTRL, which is significant at 5 % (χ^2=4.0 > 3.84) ; real coupling succeeds 38 times out of 50, against 19 times for RTRL, which is significant at 1% (χ^2=14.7 > 6.64).

An interesting and not predicted result is that good convergence results are obtained with a quite high damping factor in the spring coupling. In additional experiments, we have noticed that a pure damping gives much better results than a pure stiffness.

4.2 Continuous and interactive simulations

In the previous experiments, the learning is segmented into several sessions, consisting of the 50 samples long sequences, representing the impulsional response of the object. This is artificial relative to the real case where the object is handled. In particular, the length of the successive simulations (50 in our case), which strongly influences the learning performance, must be chosen before the experiment, by a trial and error strategy. We are now considering the case where there is *one* simulation, whose length can be arbitrarily long. In such a situation, instead of applying impulsional forces to the object, we introduce the "human touch" by letting the operator *act on real-time on the object*, according to *Fig. 3* and *Fig. 6*.

The introduction of the operator into the identification process completely disturbs our experimental and validation techniques. Different measures should be performed in varying not only the weights initialization but also the human trials (each action is different) and the human operators, according to the methods used in experimental psychology. This has not been carried out here, because the experiments reported above are preliminary. Thus, the results are rather qualitative.

The model chosen is a simple one: a mass connected by a damped spring to a "soil" (an infinite mass). This model is able to reproduce damped sinusoidal signals. As an object, we simply attached one *FFGT* to a fixed stand with rubber bands; note that the identified system includes the *FFGT* itself, which has its own viscosity and inertia. The algorithm is the "Real Coupling" (see previous sections). The experiments have been carried out with a single operator and the simulations are performed at 1 kHz (1000 adaptations per second) on specific real-time machines. The stiffness and damping factor of the coupling are 10^{-5}.

We use two methods to estimate the accuracy of the model.

Firstly, in the interaction scheme of *Fig. 6* we define the following criterion to estimate the current difference between the object and its model:

$$d_T(n) = \frac{1}{T} \sum_{\eta = n-T+1}^{n} \left| X_a(\eta) - X_{\tilde{a}}(\eta) \right|$$

that we visualize in real-time when testing the network. Theoretically, without any force applied to the object and the network, all positions *X* should be at zero, as well as $d_T(n)$. However, a non zero value is obtained, since there is noise and mechanical friction. Let us denote $d0_T$ this value and define:

$$\Delta d_T = d_T(T) - d0_T$$

Δd_T is a measure of the error of the network. Since this error depends on the gesture of the operator, the values given below are indicative.

Two series of three experiments have been carried out (see *Table 4*) corresponding respectively to different parameters initializations and human handling trials. In all these cases, the error was highly reduced in about 30 seconds of handling (there were no precise stop criterion).

m_{init}	k_{init}	z_{init}	α	β	μ	initial Δd_{10000}	trial number	final Δd_{10000}	number of adaptations
2	10^{-6}	$3\ 10^{-5}$	10^{-3}	10^{-8}	10^{-5}	0.14	1	0.03	36495
							2	0.02	26244
							3	0.02	27756
1	10^{-6}	0	$5\ 10^{-4}$	10^{-8}	10^{-5}	0.2	1	0.02	40102
							2	0.02	40032
							3	0.04	27302

Table 4. Results for continuous time simulations (m_{init}, k_{init} and z_{init} are the initial values of the mass, stiffness and damping factor of the network)

Secondly, we performed a qualitative estimation of the learning abilities of the proposed algorithms. The operator compares the object and the network by handling them *separately* via a *FFGT* and a simulated damped coupling 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 (adaptation of the m, k, z parameters), the difference does not seem perceptible, which is, at the end, our goal.

5 Discussion

We have introduced here the notion of adaptive mechanical networks. Beyond the translation from one formalism to the other - which is not straightforward - these mechanical recurrent networks open the way to some original extensions to recurrent networks.

The teacher forcing technique can be replaced by semi-coupling or coupling. Our simulations show the advantage of these two techniques over the classical RTRL algorithm and the teacher forcing (that was not successful in our problem). However, this is by no means a proof that it is the case for other types of units (like sigmoid units).

Coupling can not be implemented in every situation, and this is one of the reasons why we have developed the idea of semi-coupling, which can be used in the case where we have a fixed training set of sequences. In our simulations, coupling and semi-coupling gave similar results. Thus, the role of such an action towards the environment remains an open question.

In our interactive simulations, sequences contained a very high number of samples (for example 40000), whereas it is shown than since parameters are adapted on-line, RTRL is no longer valid for long sequences, except if the learning step (α, β or μ in our case) is very mall [2, 5]. However, this condition is unsatisfactory in practice, since for infinite sequences, an infinitely small step would be required. One solution that has been proposed is to reset to zero from time to time the sensitivity variables ($S,R,\sigma,\rho,\tau,\chi$ in our case) [3]. If such resetting had to be performed in a mechanical way, a solution would be to *damp* the mechanical network corresponding to the sensitivity variables. This is exactly what is done by the (Kc Zc) link elements of *Fig. 5*, if the value of Zc is high. Thus, semi-coupling and coupling could help for long sequences and we have effectively observed that having a high damping factor in the spring coupling is beneficent for learning (see the end of section 4.1).

Beside all these algorithmic aspects, one of the most original aspects of this study could be considered to be the interaction between the operator and the model. Learning or adaptive devices are often designed to completely replace the human in different tasks, like tuning parameters, classification, etc. But is it reasonable to totally discard the human in the learning process? Should we not study *cooperative* learning instead of automatic leaning? We propose three reasons why this idea merits further study. Firstly, many algorithms that are said to be automatic in fact are not, since they require a subtle setting of parameters, that must be performed by a human; even an expert familiar with the algorithm is sometimes necessary; this is particularly the case for neural networks. Therefore, many learning algorithms are already *implicitly* cooperative. Secondly, since active learning can be used as a *progressive* learning, the action of the human is a very good way to *give a priori knowledge* to the networks describing what must be learned first, and what may be left until later. Thirdly, the human and the machine can be viewed as complementary for learning. For example, if the machine is able to modify one hundred parameters at the same time, which is impossible for the human, the human can have a very high level representation of the progress of learning; thus a combination of these two abilities is preferred to totally automated solutions.

In the experiments previously described, we have not yet fully exploited all the advantages of interaction during learning. A particular aspect to study is the way in which real-time information about the progress of learning is presented in a manner that can help the operator to act effectively.

Conclusion

Neural algorithms have been adapted and proposed for the identification of mechanical structures, based on mechanical modeling networks and interactive learning. The results that we have obtained show the feasibility of our approach, but are far from being immediately applicable to real problems, like the identification of a real musical instrument. Indeed, some crucial extensions must be developed for cases such as nonlinear structures, since almost any mechanical object contains some nonlinearity. Likewise, multi-dimensional and plastic structures have not yet been studied.

However, it must be pointed out that, in principle, the proposed method is general: all the algorithms can be adapted for nonlinear units and used for different types of physically based units. The only condition is that the transfer functions of the units must be derivable.

More generally, the transformation and interpretation of the RTRL algorithm into mass-spring like computations leads us to the notion of "adaptive matter", which is an interesting and elegant alternative for determining physical models from a given mechanical behavior.

Acknowledgments

This research is supported by the french *Ministère de la Culture*. The authors wish to thank Claude Uhl, Jamel Nouiri and Jean-Loup Florens for their assistance in real-time simulations. The authors would also like to thank Arash Habibi for helpful discussions, Mireille Bétrancourt and Luc Rodet for assistance in statitics and the reviewers for their fruitful comments and corrections.

References

[1] N. Szilas and E. Ronco, Action for learning in non-symbolic systems. *European Conference on Cognitive Science*, Saint-Malo, France (1995).

[2] R.J. Williams and D. Zipser, A Learning Algorithm for Continually Running Fully Recurrent Neural Networks, *Neural Computation*, 1 (1989) 270-280.

[3] T. Catfolis, A Method for Improving the Real-Time Recurrent Learning Algorithm, *Neural Networks*, 6(6) (1993) 807-821.

[4] C. Cadoz, L. Lisowski and J.-L. Florens, A Modular Feedback Keyboard Design, *Computer Music Journal*, 14(2) (1990) 47-51.

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

[7] N. Szilas, Apprentissage dans les réseaux mécaniques et étude de leurs interactions avec l'environnement, *PhD Thesis*, INPG, Grenoble, France, Dec. 1995.

[7] 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(1) (1993) 19-29.

[8] N. Toomarian and J. Bahren, Fast Temporal Neural Learning Using Teacher Forcing, *IJCNN*, Seattle, USA (1991) 817-822.

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

[10] J. Louchet, Identification évolutive de modèles physiques d'animation à base de masses-liaisons. *Séminaire du groupe de travail "Animation et Simulation"* , Lille, France (1994) 79-83.

[11] J.L. Elman, Learning and development: the importance of starting small, *Cognition*, 48 (1993) 71-99.

[12] F.-S. Tsung, Learning in recurrent Finite Difference Networks, in: D.E. Touretzky, J.L. Elman, T.J. Sejnowski & G.E. Hinton, eds., *Proc. of the 1990 Connectionist Models Summer School* (San Mateo, CA: Morgan Kaufmann, 1991) 124-130.

[13] D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning internal representations by error propagation, in: D.E. Rumelhart and J.L. McClelland, eds., *Parallel Distributed Processing,* vol. 1 (MIT Press, Cambridge, MA, 1986) 316-362.

[14] O. Nerrand, P. Roussel-Ragot, L. Personnaz and G. Dreyfus, Neural Networks and Nonlinear Adaptive Filtering: Unifying Concepts and New Algorithms, *Neural Computation*, 5 (1993) 165-199.

[15] R. Hecht-Nielsen, *Neurocomputing* (Addison-Wesley, Reading, MA, 1991).

[16] B. Chanclou and A. Luciani, Physical models and dynamic simulation of planetary motor vehicles with a great number of degrees of freedom, *Intelligent Autonomous Systems - 4*, Karlsruhe, Germany (1995).

[17] N. Szilas and C. Cadoz, Physical Models that learn, *International Computer Music Conference*, Tokyo, Japan (1993) 72-75.

[18] M. Plutowski and H. White, Selecting Concise Training Sets from Clean Data, *IEEE Transactions on Neural Networks*, 4(2) (1993) 305-318.

[19] S.F. Masri, A.G. Chassiakos and T.K. Caughey, Structures-unknown non-linear dynamic systems: identification through neural networks, *Smart Materials and structures*, 1 (1992) 45-56.

[20] L. Atlas, D. Cohn, R. Ladner, M.A. El-Sharkawi, R.J. Marks II, M.E. Aggoune and D.C. Park, Training Connectionist Networks with Queries and Sampling, in: D.S. Touretzky, ed., *Advances in Neural Information Processing Systems 2* (Morgan-Kaufmann, San Mateo 1990) 566-573.