

A Walking Hexapod Controlled by a Neuro-Fuzzy System

F. Berardi, M. Chiaberge, E. Miranda and L.M. Reyneri
Dipartimento di Elettronica, Politecnico di Torino
C.so Duca degli Abruzzi, 24 - 10129 TORINO - ITALY
e.mail marcello@polimage.polito.it, fax: ++39 11 564 4099

Abstract

¹ This paper describes DANIELA, a Neuro-Fuzzy system for control applications. The system is based on a custom neural device that can implement either Multi-Layer Perceptrons, Radial Basis Functions or Fuzzy paradigms. The system implements intelligent control algorithms mixing neuro-fuzzy algorithms with finite state automata and is used to control a walking hexapod.

1 Introduction

Real-time control of non-linear plants [1, 2] is often a hard and computationally intensive task. Neural networks and fuzzy systems (neuro-fuzzy systems, in general) are raising more and more interest in the field of real-time control thanks to their superior performance [1, 2, 3]. Advantages of neuro-fuzzy techniques are their non-linear characteristics, the capability of learning from examples, the human friendly description language, the adaptation capability, etc.

Furthermore the availability of dedicated neuro-fuzzy processors [4, 5, 6] permits a very fast implementation of neuro-fuzzy algorithms, and therefore it allows to use these methods in real-time applications, where a quite high sampling rate (namely, $>1\text{kHz}$) is required.

In this paper we described DANIELA (Digital Analog Neuro-fuzzy Interface for Enhanced Learning Applications), a real time neuro-fuzzy controller that implements intelligent control

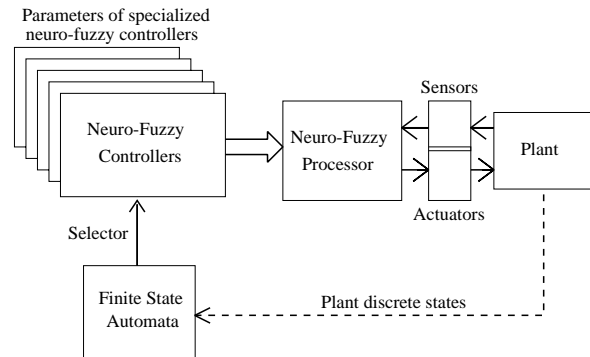


Figure 1: Working principles of DANIELA.

algorithms mixing neuro-fuzzy paradigms with Finite State Automata (FSA) (see figure 1). The FSA tracks the state of the plant and selects the parameter (e.g. weights) of the controller accordingly.

This mixing allows the system to change the control strategy according to the actual state of the plant which leads to the use of a number of very simple controllers, instead of a more complex single one which covers all possible plant conditions. For instance, our system applied to a six legged walking robot uses specialized controllers for every different plant state (for instance, forward and backward steps, curves, etc.).

In this project we have made extensive use of neuro-fuzzy control methods and finite state automata. The main reasons for the use of those methods instead of more traditional controllers are four-folds: i) we wanted to test the validity of neuro-fuzzy control techniques; ii) we had a lot of “human knowledge” on the problem and fuzzy logic seemed to be the most straightforward way to include it into the design of the controller; iii) we used this application as a testbed for the tight integration

¹This work has been partially supported by the EEC MEPI initiative DIM-109, *Neural Network Chip for Embedded Real-Time Intelligent Control*, and the Italian National Research Center’ project, “Sistemi microelettronici a basso consumo per apparecchiature portatili”

of neuro-fuzzy controllers with finite state automata; iii) we had available a neuro-fuzzy computing engine (DANIELA) with good real-time performance which was more suited to implement neuro-fuzzy than other algorithms.

We have also developed the theory of *Fuzzy State Automata* (not described here due to length limitations) as a side result of the project. Such automata derive from the tight integration of neuro-fuzzy systems and more traditional FSA, but they have better performance and are more suited than FSA to the specification and control of smooth trajectories and sequences of steps. In our case we have used Fuzzy State Automata to design the Motion Coordination Control and the Leg Control blocks of the walking hexapod, as a way to specify the cooperation between individual legs.

In order to build an experimental board of this system we have developed a custom neuro-fuzzy processor called AMINAH [4] that implements some neural and/or fuzzy paradigms. This chip, which is the evolution of the previous CINTIA neural processor [5], uses Coherent Pulse Width Modulation (CPWM) for low-power computation [10].

2 Neuro-Fuzzy Controller

Figure 2 shows a block diagram of DANIELA. It is mainly composed of a neuro-fuzzy processor (AMINAH) [4] that interacts with a general purpose 68HC11 microcontroller and a memory in order to control the plant. Other important parts of the system are Analog to Digital converters, Configuration and Refresh block and a Host Interface block. DANIELA has also special Power Drivers used to drive the actuators in the plant.

Plant control is realized with a neuro-fuzzy algorithm computed by the neuro-fuzzy processor. Algorithm characteristics both for neural and fuzzy are defined by a matrix of *synaptic weights* and other *neural parameters* that are stored into the neuro-fuzzy processor.

In order to define different control strategies (optionally, one per each state of the FSA) there are different matrices of synaptic weights and neural parameters stored in the weight mem-

ory, where every matrix is stored in a different memory bank.

Only one bank can be active per time and its data are transferred to the neuro-fuzzy processor internal memory by the Configuration and Refresh Block every time the FSA changes states.

This memory organization allows to obtain different neural controllers by changing the active memory bank. Weight memory is also used to store tables of FSA states and outputs; these parameters are used by the microcontroller to implement the FSA controller.

The microcontroller performs the following tasks:

1. on-line training: by sampling neuro-fuzzy inputs and outputs (also hidden) and modifying matrices of synaptic weights stored into weight memory [1, 2, 8]. As the microcontroller is less powerful than the neural chip, learning is a relatively slow process, but this is not a problem in most learning controllers.
2. FSA controller: by tracking the discrete states of the plant and selecting a suitable memory bank in order to change the control strategy. The computational requirements to compute the FSA are less stringent, as in most plants state transitions are much less frequent than neural computation.
3. board self test: performing tests and diagnostic at power up.
4. interface with a host computer; which can either write data into the memory or read several debugging information from the microcontrollers (such as neural states, learning performance, actual FSM state, etc.)

The purpose of the Configuration and Refresh Block is to transfer a matrix of synaptic weights and the other neural parameters from the active memory bank to the neural processor every time the FSA changes state.

A prototype board of the DANIELA system is currently under manufacturing. This board uses AMINAH (see section 2.1) as neuro-fuzzy

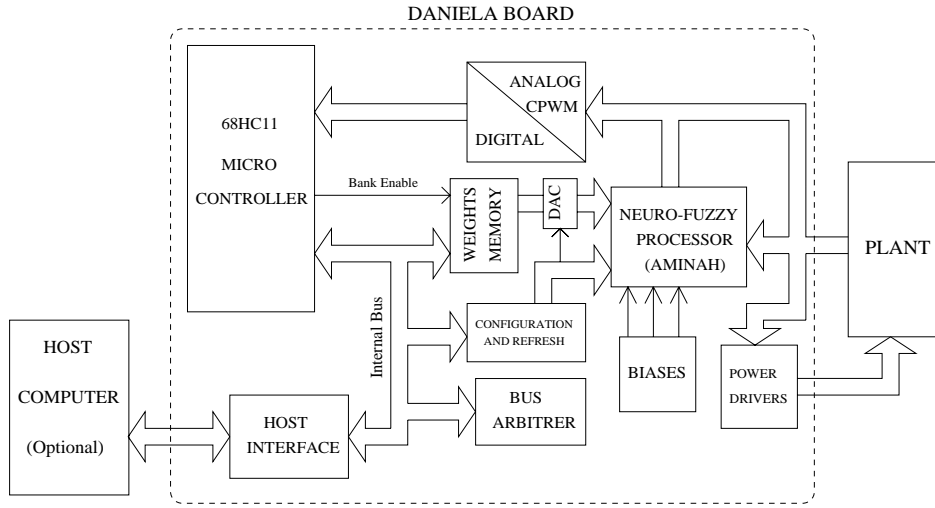


Figure 2: Block diagram of DANIELA.

| | AMINAH | NLX420 | 80170NW | ZISC036 |
|-------------------------------|--------|-----------------|----------|----------|
| Chip power diss. (mW) | 75 | 1500 | 1500 | 1000 |
| Computation energy (pJ) | 1000 | 5000 | 750 | 60000 |
| CPS ($10^6/s$) | 64 | 300 | 2000 | 16 |
| Response time (μs) | 6 | ≥ 200 | ≥ 2 | ≥ 4 |
| Clock Frequency (MHz) | 0.250 | 20 | 0.5 | 16 |
| Number of synapses | 256 | $16 \times 64K$ | 10240 | 2304 |
| Number of neurons | 24 | $16 \times 64K$ | 64 | 36 |
| Weight precision (No of bits) | 6 | 1,4,8,16 | 6 | 8 |

Table 1: Performance Comparison between AMINAH and other commercial devices.

processor and a commercial 68HC11 as general purpose processor. A 16Kbyte RAM is used as weight memory and contains 16 memory blocks; each block consists of 512 bytes of synaptic weights (8 bit/weight) and other neural parameters (about 500 bytes), resulting in 1Kbyte per weight matrix. Most of the other logic circuits are implemented on an ALTERA EPM7128. This prototype board includes also Analog to Digital converters and power switching amplifiers for motor driving.

2.1 Neural Processor Chip

AMINAH [4] chip has two cascaded layers called hidden and output layer respectively. The architecture of each layer of AMINAH is based on an $N_{inputs} \times M_{outputs}$ synaptic array and a vector of M neurons. The hidden layer has $N=8$ and $M=16$, while the output layer has $N=16$ and $M=8$.

Output signals are coded using Coherent

Pulse Width Modulation (CPWM), a particular Pulse Stream technique [5, 10]. Inputs signal (X_i) can either be analog or CPWM; in the former case, they are converted using internal Analog to CPWM converters.

Each synapses uses two weights (excitatory and inhibitory) to compute the synaptic contribution; weights are stored internally in AMINAH using two capacitive memories per synapsis with an unlimited weight retention time (due to internal-refresh).

Synaptic contributions depend also on the chosen neural paradigm; this can either be a *Multi-Layer Perceptron (MLP)*, a *Radial Basis Function (RBF)*, or a *Weighted Radial Basis Function (WRBF)* of order 1, and therefore the system can also emulate *Fuzzy Systems* [9]. AMINAH dissipates 75mW at 166kHz sampling rate (equivalent to $292\mu W$ per synapses).

Table 1 shows a performance comparison between our neuro-fuzzy processor and some



Figure 3: Photograph of the walking hexapod capable of walking on rough surfaces.

other commercial neuro-fuzzy systems. Notice that NeuraLogix NLX420 and IBM ZISC036 are DSP-based systems while AMINAH and Intel 80170NW are analog devices.

3 Walking Hexapod

This section describes an application of DANIELA in the field of control. The aim is to build and control an autonomous six-legged walking robot (Figure 3).

Legged robots, or more in general walking machines, have been widely studied in the past years [7, 11], because they represent a better solution with respect to the classical wheeled or tracked robots when they must move over corrupted surfaces. Legged motion can easily avoid large obstacles on the path and any kind of direction change can be performed more quickly in less space. They can also move sideways and can represent a better approach for moving in environments, where the surface has less adherence (on the moon for instance, where the lower gravity causes a friction reduction).

On the other hand it is a hard task to let an hexapod move, because of the complexity of the robots kinematics and dynamics and the complex coordination between legs for each gait. Furthermore legged robots have to be designed to handle with obstacles, hill, stairs, and leg coordination in such environments is a tough task. If the speed of movement is not too fast, the system can be considered in static equilibrium, that means that the robots has always

some legs in contact with the ground, differently from the more complex condition of dynamic equilibrium, where this assumption is not respected.

The development phases of the project have been the following:

1. modeling individual legs and the complete hexapod system. We have purposely chosen a simple model mainly to test the capability of the neuro-fuzzy controller to deal with approximate models.
2. identification of the kinematics and the dynamics of the robot (individual legs and complete robot).
3. design and implementation of a neuro-fuzzy controller for individual legs.
4. design of the motion coordination system (gait control and obstacle avoidance).
5. investigating the ability of the neural controller to autonomously learn to move on uneven grounds and to deal with obstacles. This phase has not been started yet.

3.1 Hexapod Leg Description

Each leg is made of a shinbone and a thighbone both 50 cm long; it has three degrees of freedom and is equipped with three current-controlled motors. To obtain high efficiency, switching amplifiers are employed, which are perfectly matched to the CPWM encoding used in the DANIELA system. All the actuators are directly controlled by the power drivers, as explained in section 2. One of the actuator must carry a significant fraction of the hexapod weight (depending on the number of legs that support the body) and so its driver has a maximum output current of 20A (namely, 480W), while the other two drivers have only 3A (namely, 72W).

3.2 Control Hierarchy

Since the control task is complex, the idea is to build a hierarchy of neuro-fuzzy controllers and FSA. Neuro-fuzzy controllers have been chosen because of their good behaviour in the presence of non linear systems and for this intrinsic

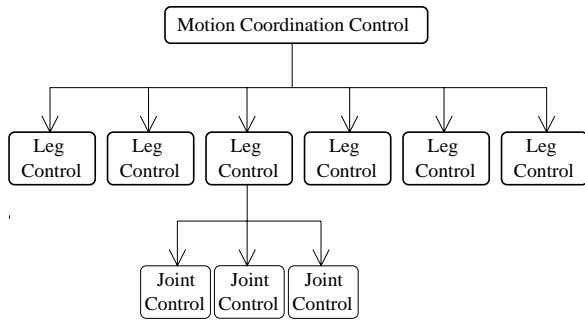


Figure 4: Robot control hierarchy

generalization capability. The control system is organized in three different levels (figure 4):

- motion coordination control. It acts like a "central" controller, to give the legs the right "high-level" control signals (such as: robot speed, robot height from ground, radius of trajectory) to let the hexapod execute properly each gait. Obstacle avoidance is done by giving proper trajectory information to the individual Leg Controls. Only large obstacles are avoided, as small ones (namely roughnesses of the ground) are handled by each Leg Control.
- leg controls. They are like a "local" control that define the trajectory and the sequence of movements of each leg. They also recover and modify leg trajectories when small local obstacles are encountered.
- controls of joint positions. They receive the angular positions and translate them into control signals for positioning servos.

Further developments will integrate higher control levels with different kinds of sensors (like vision, touch, etc.) to drive the robots in complex environments.

3.3 Leg Control

Locomotion control is distributed evenly among the six legs. Independently from the different gait, each leg cycles over four different phases:

- power phase: the leg leans on the ground where it supports and propels the body, moving backward to the posterior extreme position.

- lift phase: the leg rises from the posterior extreme position and loses its support function. Lift phase ends when the leg is high enough to swing forward.
- return phase: leg swings forward to the anterior extreme position. As soon as it reach this point is ready to the next phase.
- contact phase: leg lowers down to the ground. During this phase it starts again to support the body weight.

Each leg controller must interact with the Motion Coordination Control to synchronize leg phases and to perform each different gait.

3.4 Motion Coordination Control

The local leg controllers run simultaneously; however they are not independent of one other. To let the robot move and deal with obstacles, an inter-leg coordination is needed. For instance, when a leg is trying to step over an obstacle, it can ask the supporting legs to momentarily stop, to raise the body, or to move the robot backwards, when the obstacle is too large to step over. To achieve this inter-leg coordination, each Leg Control communicates with the Motion Coordination Control.

3.5 Fuzzy State Automata

The Motion Coordination and the Leg Controls have been developed using a number of custom Fuzzy State Automatas (FFSA), and in particular one for each leg plus one for the Motion Coordination Control.

As shown in fig. 5, an FFSA looks like a finite state automaton which tracks the discrete states of the system (for instance, power, lift, return, and contact phases). Each state is associated with one of a set of different controllers (C_j), each one tailored to the specific system state.

Such an approach is easier to design than a single controller which can handle all possible system conditions, but it has two major drawbacks: state transitions are abrupt, thus control is passed immediately from one controller to the next one, and the resulting trajectory

cannot be smooth, unless the controllers are designed accordingly (but such modifications are not easy to design and tune). Furthermore, having crisp state transitions in an otherwise fuzzy system might seem antithetic.

The main difference of an FFSA with respect to the above described method is that transitions are not triggered by crisp events, but by fuzzy variables (all labels on state transitions in fig. 5 are fuzzy events), and state transitions are fuzzy as well. It immediately results that, at any time, the whole system is not necessarily in one well-defined state, but it may well be in more states, each one with its own *state membership value*.

State transitions are therefore smoother and slower, even if the controllers described in each state are not designed to smooth them. As a consequence, all the individual controllers may become as simple as a traditional PID, each one with a different target (for instance, when the leg is carrying the body weight, the PID should keep body height and speed constant, while in the backward step it should move as quick as possible towards a target point). Smoothing of transitions between partial local trajectories is then taken care by the FFSA, which is much simpler to design and tune than a controller with trajectory smoothing, as smoothing is intrinsic in the FFSA working principles.

Unfortunately FFSA have a drawback, that is, as the system is often in more than one state, it has to process more than one controller at any time. This increase of computing time often counterbalances the higher speed that can be achieved through the use of simpler controllers. Globally, an FFSA has approximately the same computing complexity of an FSA controlling Neuro-fuzzy controllers, but it is easier to design (especially for complex systems) and produces smoother trajectories.

Fig. 5 shows a simplified FFSA for the Leg Control. It has provisions for the normal leg step (namely, walking on smooth surfaces without obstacles) and to recover from obstacles encountered during leg movements. The actual FFSA is more complex, as it includes provisions for: i) modification of hexapod attitude in presence of complex obstacles (e.g. to walk inside low tunnels), ii) modification of leg trajectory

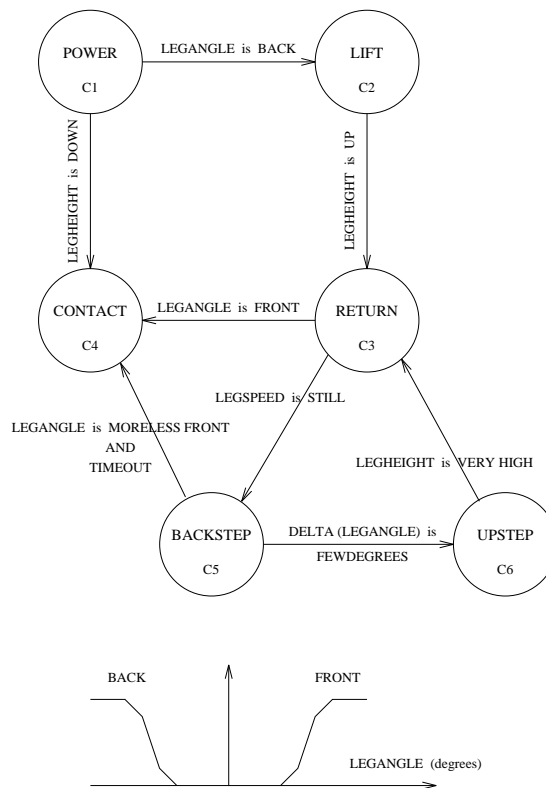


Figure 5: FFSA of the Leg Control.

when walking sideways, iii) control of alternative paces for dynamic gaits (e.g. galloping, or climbing staircases); iv) other special cases a leg might have to face.

3.6 Leg Identification and Neuro-Fuzzy Controller

Kinematic and dynamic identification of the leg model has been performed with different neural paradigms such as MLP, FUZZY, RBF. Each leg motor was driven by an independent net. Training has been done with several linear trajectories (60 cm movement amplitude) at different heights (10, 20 and 30 cm) of the robot body from the ground. The test has been performed on a completely different parabolic trajectory with a slightly wider range to test the generalization ability of the net. So far only simulations have been done, as the real hexapod is available only since few days. Results for all the different paradigms are comparable, as shown in table 2. As can be seen, an MLP network with only 3 hidden neurons is sufficient to generate a trajectory with good precision (max. RMS error is ≈ 4 mm for 80 cm

| Paradigm | Training Set (mm) | | | | Test Set (mm) | | | |
|----------|-------------------|-------|------|------|---------------|-------|------|-------|
| | x | y | z | av. | x | y | z | av. |
| MLP (3) | 0.17 | 0.41 | 0.43 | 0.35 | 2.46 | 2.77 | 4.57 | 3.39 |
| MLP (6) | 0.73 | 0.62 | 0.36 | 0.59 | 2.71 | 5.50 | 2.77 | 3.88 |
| LIN | 6.09 | 19.26 | 1.13 | 11.6 | 19.08 | 49.96 | 1.49 | 30.88 |
| POLY | 0.23 | 0.47 | 1.77 | 1.06 | 3.19 | 4.02 | 7.38 | 5.18 |
| RBF | 0.42 | 0.14 | 0.32 | 0.31 | 2.70 | 0.51 | 5.38 | 3.48 |

Table 2: Positioning errors for different neural paradigms used for the kinematic identification of an hexapod leg.

movement range). A similar accuracy can be obtained with a polynomial (POLY) or with a RBF networks.

Dynamic identification of the leg [1, 2] has been performed using the scheme shown in figure 6(a). For each coordinate a neural net has been trained to evaluate the acceleration of the leg caused by the application of the $u(t)$ stimulus that represents the input current to the motor drivers. The leg position is represented by $s(t)$.

The scheme used to train the neural controller for a single leg is shown in figure 6(b). It is a case of inverse control [2], where the neural model of the leg (i) is required to back propagate the training errors to the controller (c). The system works at 1 kHz sample frequency and this is equivalent, for the MLP (6), to 72000 CPS (connections per second) for a single leg controller. This rate can be achieved only with high computational power DSP or with a dedicated hardware implementation such as DANIELA.

4 Conclusion

This paper has described an autonomously walking hexapod controlled by means of Fuzzy State Automata, which have been developed starting from a mixture of Finite State Automata and Neuro-Fuzzy controllers. The proposed method has demonstrated interesting characteristics, as a way to develop non-linear controllers capable to handle complex sequences and coordination functions. The system is currently under integration and testing, but preliminary simulation results showed that the proposed approach provides good perfor-

mance.

References

- [1] *W.T.Miller, R.S.Sutton, P.J.Werbos editors*, "Neural Networks for Control" 1990.
- [2] *D.A.White D.A.Sofge editors*, "Handbook of Intelligent Control", Van Nostrand Reinhold 1992.
- [3] L. Wang, "Adaptive Fuzzy Systems and Control", *Prentice Hall*, Englewood Cliffs, New Jersey, 1994.
- [4] M.Chiaberge, E.Miranda, L.M. Reyneri, "A Pulse Stream System for Low-Power Neuro-Fuzzy Computation", *IEEE Transaction on Circuits and Systems*, Vol. 42, No. 11, Nov 1995, pp. 946-954.
- [5] L.M. Reyneri, M. Chiaberge, L. Zocca, "CINTIA: A Neuro-Fuzzy Real Time Controller for Low Power Embedded Systems", *IEEE MICRO, special issue on Hardware for Artificial Neural Networks*, June 1995, pp. 40-47.
- [6] The 80170NX Electrically Trainable Analog Neural Network (ETAN) Chip, Data Sheet, Intel Corporation 1992.
- [7] C.Ferrel, "Robust and Adaptive Locomotion of an Autonomous Hexapod", *From Perception to Action Conference Proceedings*, Lausanne, 1994.
- [8] S. Haykin, "Neural Networks: A Comprehensive Foundation", *Mc Millan College Publishing Company*, New York, 1994.
- [9] L.M. Reyneri, "Weighted Radial Basis Functions for Improved Pattern Recognition and Signal Processing", *Neural Processing Letters*, Vol. 2, No. 3, May 1995, pp. 2-6.
- [10] L.M. Reyneri, "A Performance Analysis of Pulse Stream Neural and Fuzzy Computing Systems", *IEEE Trans. on Circuits and Systems*, Vol. 42, No. 10, October 1995.

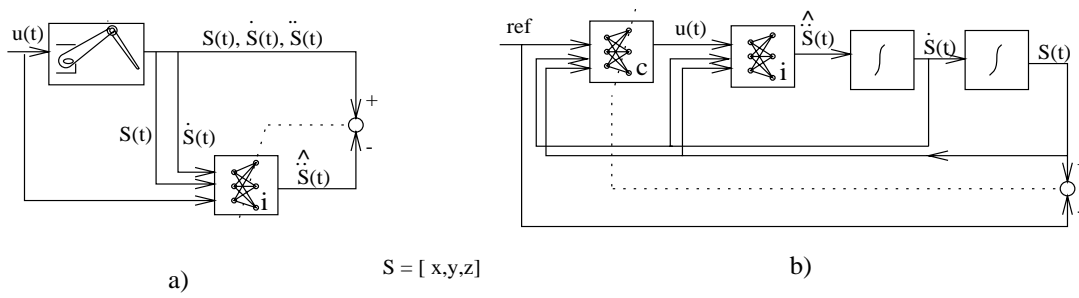


Figure 6: Training scheme for leg dynamic identification (a) and leg neuro-fuzzy control (b)

- [11] D. Bassani, M. Chiaberge, D. Del Corso, G. Genta, F. Zanetti, "Simulation of a Hexapod Walking Machine Controlled by Neural Networks", *Proceedings of the Third International Symposium on Measurement and Control in Robotics*, Sept 1993.