TORQUE CONTROL WITH RECURRENT NEURAL NETWORKS

Guillaume Jouffroy
Artificial Intelligence Laboratory, University Paris 8, France

Keywords: Joint constraint method, oscillatory recurrent neural network, generalized teacher forcing, feedback, adaptive systems.

Abstract: In the robotics field, a lot of attention is given to the complexity of the mechanics and particularly to the number of degrees of freedom. Also, the oscillatory recurrent neural network architecture is only considered as a black box, which prevents from carefully studying the interesting features of the network's dynamics. In this paper we describe a generalized teacher forcing algorithm, and we build a default oscillatory recurrent neural network controller for a vehicle of one degree of freedom. We then build a feedback system as a constraint method for the joint. We show that with the default oscillatory controller the vehicle can however behave correctly, even in its transient time from standing to moving, and is robust to the oscillatory controller's own transient period and its initial conditions. We finally discuss how the default oscillator can be modified, thus reducing the local feedback adaptation amplitude.

1 INTRODUCTION

Central Pattern Generators (CPG) are biological periodic oscillatory neural networks responsible for a wide range of rhythmic functions. They can be made of endogeneous oscillatory neurons connected to non oscillatory ones or from the sole interaction between non oscillatory neurons.

Particularly, they are a great source of inspiration in the robotics field, for the control of joints in locomotion. In general, an oscillatory network controls a joint angle in both directions, and the phase relationships needed between all joints arise from the coupling between the different networks.
The needed parameters for an artificial Recurrent Neural Network (RNN) to have a periodic oscillatory behavior cannot be measured experimentally. This network is most of the time a relatively simplified model of its biological counterpart when available. Only clinical temporal data of joints kinetics and kinematics can be of use, where however it is difficult to isolate the real control of a particular joint from the influence of the others.

In the case of non endogeneous oscillatory neurons, in the litterature, parameters are thus mainly determined empirically or with genetic algorithms (Buono and M.Golubitsky, 2001), (Ghigliazza and P.Holmes, 2004), (Ishiguro et al., 2000), (Kamimura et al., 2003), (Taga, 1994), (Ijspeert, 2001), comparatively to relatively few learning methods (Mori et al., 2004), (Tsung and Cottrell, 1993), (Weiss, 1997). Though this is useful with large networks in complex mechanical models, there are two drawbacks. It is very difficult in general to isolate the resulting dynamics of the different networks, and to understand their interaction to each other and with the mechanical system dynamics. Also, it is not clear how to modify such networks in an adaptive context, e.g. in the case of a permanent constraint change on a joint due to injury.

Based on this considerations, we apply a *generalized* formulation of the so called *teacher forcing* gradient descent-based learning algorithm, to create an oscillatory RNN as a *torque controller* for an interesting vehicle with one single degree of freedom, the Roller Racer. The RNN is put in a closed loop with the Roller Racer, such that the vehicle can be freely controlled, where the RNN can be modified permanently.

The paper is structured as follows. In section 2, we briefly present the Roller Racer model and we show how it can be controlled with a torque input. In section 3 we describe the control system. The subsection 3.1 presents the generalized formulation of the teacher forcing learning algorithm with which we build the oscillatory RNN as a basic torque controller

Jouffroy G. (2008).
TORQUE CONTROL WITH RECURRENT NEURAL NETWORKS.
In *Proceedings of the Fifth International Conference on Informatics in Control, Automation and Robotics - RA*, pages 109-114
DOI: 10.5220/0001501401090114

for the Roller Racer vehicle. In subsection 3.2 we describe the local feedback control that can be built so that the vehicle direction can be controlled, limiting the effect of its transient state. We discuss how the basic oscillatory system can be In section 4, we give concluding remarks and discuss how the basic oscillatory system can be modified to better fit the needs of the Roller Racer, thus reducing the local feedback adaptation amplitude.

2 VEHICLE MODEL

The Roller Racer is a toy-vehicle with one single degree of freedom which is the handlebar. The direction wheels are shifted back from the the axis. Thus, oscillating the handlebar from side to side, a component of the reaction force on the ground which points backward is created, moving forward the vehicle.

In (Jouffroy and Jouffroy, 2006), we revisited and synthetized a mathematical model of the Roller Racer from the original work of Krishnaprasad and Tsakriris (Krishnaprasad and Tsakriris, 1995). The input control was the angle of the axis. Here, we will describe the torque input control formalization.

Recall the state of the Roller Racer vehicle is $\mathbf{x} \triangleq (\theta_r, x_r, y_r, p, \theta_c, \dot{\theta}_c)^T \in \mathbb{R}^6$, and its dynamics is $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u)$, where

$$\mathbf{f}(\mathbf{x},u) \triangleq \begin{pmatrix} \frac{1}{\Delta(\theta_c)}\left(\sin\theta_c p - \delta(\theta_c)\dot{\theta}_c\right) \\ \frac{\cos\theta_r}{\Delta(\theta_c)}\left(\chi(\theta_c)p - \gamma(\theta_c)\sin\theta_c\dot{\theta}_c\right) \\ \frac{\sin\theta_r}{\Delta(\theta_c)}\left(\chi(\theta_c)p - \gamma(\theta_c)\sin\theta_c\dot{\theta}_c\right) \\ \left[A_1(\theta_c)\dot{\theta}_c - C_1(\theta_c)\right]p + \\ \qquad \left[A_2(\theta_c)\dot{\theta}_c + C_2(\theta_c)\right]\dot{\theta}_c \\ \dot{\theta}_c \\ u \end{pmatrix}, \quad (1)$$

θ_c is the angle of the handlebar, p is the momentum of the vehicle and (x_r, y_r) are the rear coordinates respectively to the global reference space. Friction constraint is built in the model through the functions $C_1(\theta_c)$ and $C_2(\theta_c)$. Thus one does not need to deal with mechanical aspects, leaving focus on the control strategy, and on the learning aspects of the RNN.

Here we control the Roller Racer using the torque input control T_c with the following equation

$$\ddot{\theta}_c = u = B_1(\theta_c)\theta_c p + \dot{B_2}(\theta_c)\dot{\theta}_c^2 + B_3(\theta_c)T_c \quad (2)$$

The right hand side of the equation replaces u in (1). The parameters are defined as

$$\begin{aligned} B_1 &\triangleq -\frac{A_2(\theta_c)}{\Delta_1(\theta_c)} \\ B_2 &\triangleq \frac{m_1\gamma(\theta_c)\sin\theta_c}{\Delta(\theta_c)\Delta_1(\theta_c)}[\gamma(\theta_c)\cos\theta_c + d_1\delta(\theta_c)] \\ B_3 &\triangleq \frac{\Delta(\theta_c)}{\Delta_1(\theta_c)}, \end{aligned}$$

where

$$\Delta_1 \triangleq I_1 I_2 \sin^2\theta_c + m_1(I_1 d_2^2 + I_2 d_1^2 \cos^2\theta_c),$$

and the other parameters are as defined in (Jouffroy and Jouffroy, 2006).

3 DESIGN OF THE OSCILLATORY CONTROLLER

3.1 The Oscillatory Recurrent Neural Network Torque Controller

Let us consider the RNN system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{W}), \quad (3)$$

with $\mathbf{x} \in \mathbb{R}^n$ is the state vector of the network, $\mathbf{W} \in \mathbb{R}^{n\times n}$ is the matrix of the weight connexions, w_{ij} to be considered as the weight from the neuron i to the neuron j. We consider a *fully* connected RNN, which means all neurons are interconnected and self-connected ($w_{ii} \neq 0$). For the neuron model we use the rate based neuron model of the simplest form

$$\mathbf{f}(\mathbf{x}, \mathbf{W}) = (\mathbf{I}\tau^{-1})(-\mathbf{x} + \mathbf{W}s(\mathbf{x})), \quad (4)$$

with $s(x)$ a squashing function such as $\tanh(x)$. $\mathbf{I}$ is the identity matrix and $\tau \in \mathbb{R}^n$ the time constant vector of the system.

Each component x_i^* of the teacher vector $\mathbf{x}^*$ is of the form

$$x_i^* = \sin(t + \phi_i) \quad (5)$$

The learning is achieved when an error criterion E, $E \in \mathbb{R}^n$ is less or equal than a minimum $\varepsilon \in \mathbb{R}$, $\varepsilon \simeq 0$

$$E = \frac{1}{2}(\mathbf{x} - \mathbf{x}^*) \circ (\mathbf{x} - \mathbf{x}^*) < \varepsilon, \quad (6)$$

the operator $\circ$ being the Hadamard product.

The weight matrix $\mathbf{W}$ is ajusted according to the following gradient rule

$$\dot{w_{pq}} = -\eta \sum_{i=1}^{n} \frac{\partial E}{\partial x_i} z_{pq}^{i}, \tag{7}$$

with $\eta \in \mathbb{R}$ is the learning rate. $\mathbf{z} \in \mathbb{R}^{n \times n^2}$ is the sensitivity of the state of the system with respect to a weight w_{pq}, which can be written in the matrix form

$$\frac{d\mathbf{z}}{dt} = (\mathbf{I}\tau^{-1})(\mathbf{J}_{\mathbf{f_x}}(M)\mathbf{z} + \mathbf{J}_{\mathbf{f_w}}(M)), \tag{8}$$

where $\mathbf{J}_{\mathbf{f_x}}(M)$ and $\mathbf{J}_{\mathbf{f_w}}(M)$ are the jacobian matrices of the function $\mathbf{f}$ respectively to $\mathbf{x}$ and $\mathbf{w}$ at the point M. In the teacher forcing case (8) reduces to

$$\frac{d\mathbf{z}}{dt} = (\mathbf{I}\tau^{-1})(-\mathbf{I}\mathbf{z} + \mathbf{J}_{\mathbf{f_w}}(M)), \tag{9}$$

For convenience $\mathbf{z}$ is of the form

$$\begin{bmatrix} z_{11}^{1} & \cdots & z_{nn}^{1} \\ \vdots & \ddots & \vdots \\ z_{11}^{n} & \cdots & z_{nn}^{n} \end{bmatrix} \tag{10}$$

Providing a target signal(s) x_i^* only for some neuron(s) i, letting $\mathbf{J}_{\mathbf{f_x}} = \mathbf{A}$, the sensitivity equation (8) can be written as

$$A_{ij} = -a_{ij} + b_{ij} w_{ji} \frac{\partial s(x_j)}{\partial x_j}, \tag{11}$$

with $a_{ij} = 1$ when $i = j$, 0 otherwise, $b_{ij} = 1$ when i is not a forced neuron, 0 otherwise. $\mathbf{J}_{\mathbf{f_w}} = \mathbf{B}$ is of the same form as $\mathbf{z}$ and its elements are such that $B_{pq}^{i} = 0$ when $i \neq q$, $B_{ij} = s(x_p^*)$ if p is a forced neuron, $B_{ij} = s(x_p)$ otherwise.

With this algorithm we build a 4 neurons fully connected RNN. Teacher signals have an amplitude of 1, and we choose the phase difference vector ϕ^* as $\{0; \pi/3; 2\pi/3; \pi\}$. To generate the torque control we use the output of the neurons 1 and 4 which are in opposite phase, to control each direction of the handlebar angle using the transformation

$$T_c = pos(x_1) - pos(x_4) \tag{12}$$

The use of 3 neurons could have been the minimum acceptable to solve the phase difference of π between the output neurons. But in the scope of recover, with at least 4 of them if one break, we have the opportunity to start the algorithm again and obtain the needed oscillator. The data obtained for the weight matrix $\mathbf{W}$ of the estimated oscillator needed is

$$\mathbf{W} = \begin{bmatrix} 0.493 & -0.18 & -0.673 & -0.493 \\ 0.958 & 0.882 & -0.076 & -0.958 \\ 0.465 & 1.062 & 0.597 & -0.465 \\ -0.493 & 0.18 & 0.673 & 0.493 \end{bmatrix}$$

Convergence is reached in at most 300 timesteps, with $\eta = 0.1$.

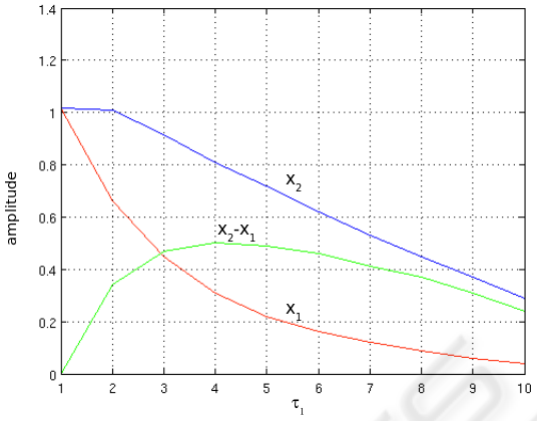

Figure 1: Effect of the modification of the time constant τ_1 on the state x_1 in the oscillatory neural network. τ_2 is kept fixed($\tau_2 = 1$).

3.2 Feedback Design of the System

The torque amplitude of the oscillatory controller, considering its frequency, may be too large for the needs of the Roller Racer. Beside, the starting energy for a vehicle is generally different from when it is at full speed. So is the Roller Racer with the torque control. It needs a transient oscillatory input which might depend on friction biases and inertia. Thus we create an angle feedback from the Roller Racer to the oscillatory controller, which purpose is to constrain the angle within some limits. We control the vehicle in the forward direction which means the average angle of the handlebar should be π (see (Jouffroy and Jouffroy, 2006)).

To apply a correction to the torque generated, we use the feedback to modify the time constants τ_i of the oscillatory controller's output neurons. In Figure 1 we illustrate the effect on the amplitude of the output neurons state when changing only one time constant (τ_1 on the figure). As might be expected, the state of each neuron output decreases. However the correction applies more to x_1, and the amplitude difference is maximum at $\tau \approx 4$. It is not really really desirable to have one output which becomes zero as it prevents the Roller Racer to get energy. Therefore we should not have τ_i being set too high.

We now define the transfer function, that constrains the angle within the boundaries $[\pi - 1; \pi + 1]$ and with a correction applied when $\tau_i < 5$, considering the effect on the amplitude reduced above this limit. Note that the frequency is relatively not modified in this range (a frequency modification would take place if all τ_i where equally changed). The transfer function $\mathbf{g}$ for the feedback is defined as

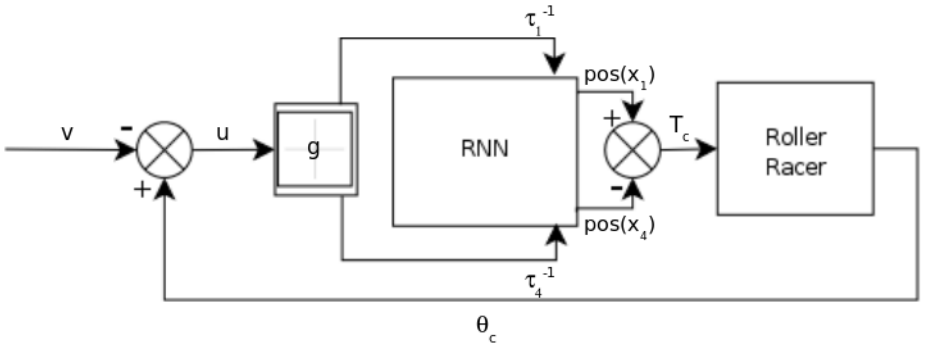

Figure 2: Control architecture of the Roller Racer. T_c is the torque control applied to the vehicle. v is the angular input which is obtained from the user direction control δ.

$$\mathbf{g}(u) = \begin{bmatrix} g_1(u) \\ g_4(u) \end{bmatrix}, \tag{13}$$

with

$$g_1(u) = \frac{6}{1+e^{-(6u-4.5)}}, \tag{14}$$

g_1 being the transfer function for the feedback to the neuron 1, and $g_4(u) = -g_1(u)$ to the neuron 4.

The formalization of these transfer functions has been chosen so that they can be used as "feedback" neurons, with the same neuron model as in (3) , replacing f_i by g_i.
The signal $u \in \mathbb{R}$ is the actual feedback signal which is the difference between the angle of the handlebar θ_c (in radians) and the desired average angle control $v \in \mathbb{R}$. For the forward direction u is actually set as

$$u = \theta_c - v = \theta_c - (\pi + \delta), \tag{15}$$

where $\delta \in \mathbb{R}$ is the user direction input, which is actually the continuous component control of the RNN.

The feedback information thus obtained is used to modify the time constants τ_1 and τ_4

$$\tau_1^{-1} = \frac{1}{1+g_1(u)} \text{ and } \tau_4^{-1} = \frac{1}{1+g_4(u)} \tag{16}$$

The whole architecture is summarized in Figure 2.

3.3 Results

We present here the results of two different trials. In both of them the purpose is to have a straight trajectory along the x axis of a physical space reference. The architecture is of course able to freely control the vehicle in all directions but the simulations are not shown.
In the first trial we start the RNN with a little energy given to a neuron. The neuron 3 in the following simulation has the initial condition $x_3 = 0.1$.

In Figure 3 left, is plotted the angle of the handlebar θ_c which continuous component is π, for the forward direction. It clearly shows the transient time when the system is extracting itself from reaction forces, until it reaches its permanent speed at around 40 timesteps. The transient period has an amplifying oscillation around π because the RNN is also in its transient state with little energy.

This transient activity of the RNN, is shown by the very weak correction applied from the feedback $g(u)$ in Figure 3 right, and the low speed along the x axis in Figure 3 bottom. One can clearly see on this graphics of the right side of the figure, a little deviation during this time. This is only the drawback of the transient state of the RNN which does not provide a symmetric gain, even if the angle is within the boundaries $[\pi - 1; \pi + 1]$.

The correction applied once the vehicle is in its permanent state shows that the RNN's own oscillation is not optimal and that a relearning could fix this. The trajectory however becomes quite straight.

In the second trial we initialize the RNN with a strong gain to see how the feedback behaves during the transient period. We set $x_3 = 1$.

In Figure 4 left we can see that the amplitude of the transient state of the RNN has been pushed too high. The angle of the handlebar θ_c does not show anymore an amplifying oscillatory behavior, and reach the boundaries we have specified. The correction from the feedback apply a high gain correction (see Figure 4 right).

After $t \approx 40$, as in the first trial, the symmetric oscillations are recovered, and the correction reduces to the steady-state time in Figure 3 right. Interestingly the correction has constrained the deviation well, which is not higher than in the first trial, except in the transient period. The vehicle also gets speed earlier and the trajectory is also straight (Figure 4 bottom).

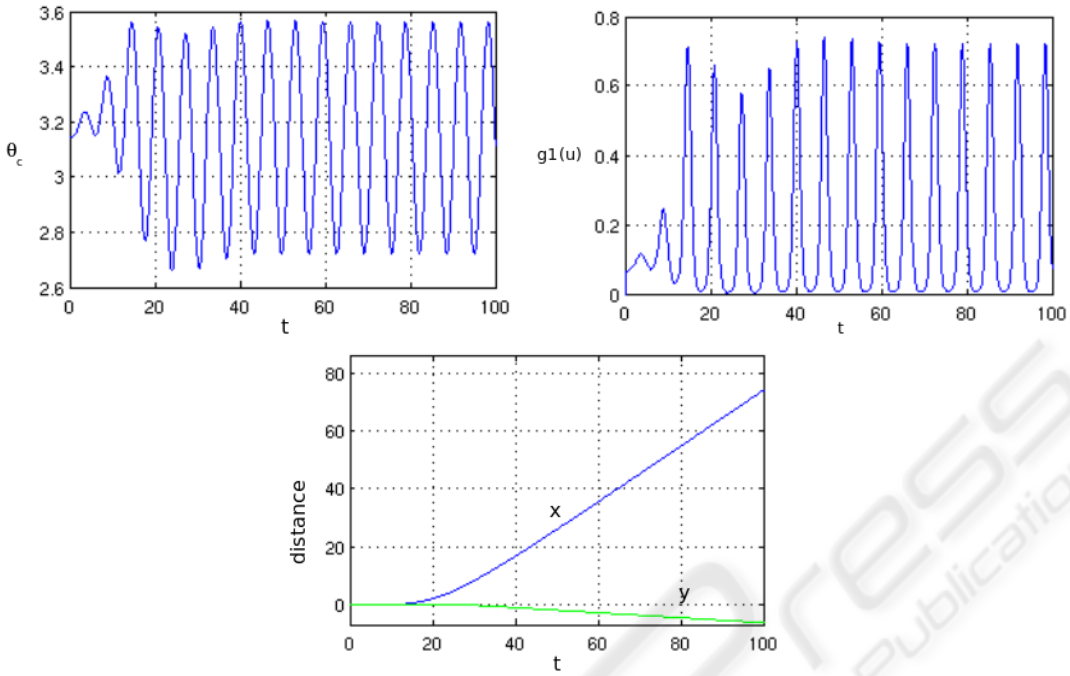

Figure 3: Simulation results when the RNN is initialized with a weak energy. Left: angle of the handlebar θ_c. Right: feedback correction $g_1(u)$. Bottom: trajectory evolution. During and after the transient times, little correction is applied.

4 CONCLUSIONS AND DISCUSSION

Nature is a great source of inspiration for engineers who deal with autonomous robots. Evolution has found optimally-designed solutions for robustness and adaptability in a changing environment, which are exciting to discover. However, most of the research in the control aspects of robots with neural networks tackle the question of complex mechanics with many degrees of freedom, and massive neural architectures, which appear as "black boxes" designed by genetics algorithms. This hides the dynamics of the neural system and correlatively the opportunity to constitute adaptive strategies.

In this work, we present a generalized version of the teacher forcing learning algorithm, to build up an estimated oscillatory controller for a vehicle with one degree of freedom, the Roller Racer. We create an angular feedback such that the degree of freedom is constrained within some boundaries. The purpose is to prevent the vehicle to go out of control during its transient state when it starts moving, as a consequence of the oscillator being not adapted to this particular moment.

Our simulation results show that the feedback makes the vehicle to behave relatively well during transient state, when the oscillator is initialized with a weak energy or even a strong one. The deviation also stays little. When the steady-state period is reached, the vehicle moves in a straight line as expected.

From a design point of view, the correction applied by the feedback system to the RNN, never completely vanishes. This shows that a more optimal oscillatory behavior can be obtained, though the "default" one does not critically affect the system with the help of a *mutual entrainment* between the vehicle dynamics and the controller, as described first by (Taga, 1994).

We are currently studying how an adaptive process or observer, can modify permanently the RNN when the average correction is too high. The outputs of the network, with the help of the feedback, could be the desired targets for a second network which could thus be trained in parallel, with a partial teacher forcing. However this is highly computationally expensive, and not biologically viable.

The teacher forcing principle is made such that an oscillatory behavior can be obtained with a gradient descent algorithm. However, forcing the outputs of the network means disconnecting it, and thus loosing the interesting desired target obtained with feedback. Algorithms without direct gradient descent evaluation techniques may be more appropriate (for e.g. (Kailath, 1990)).

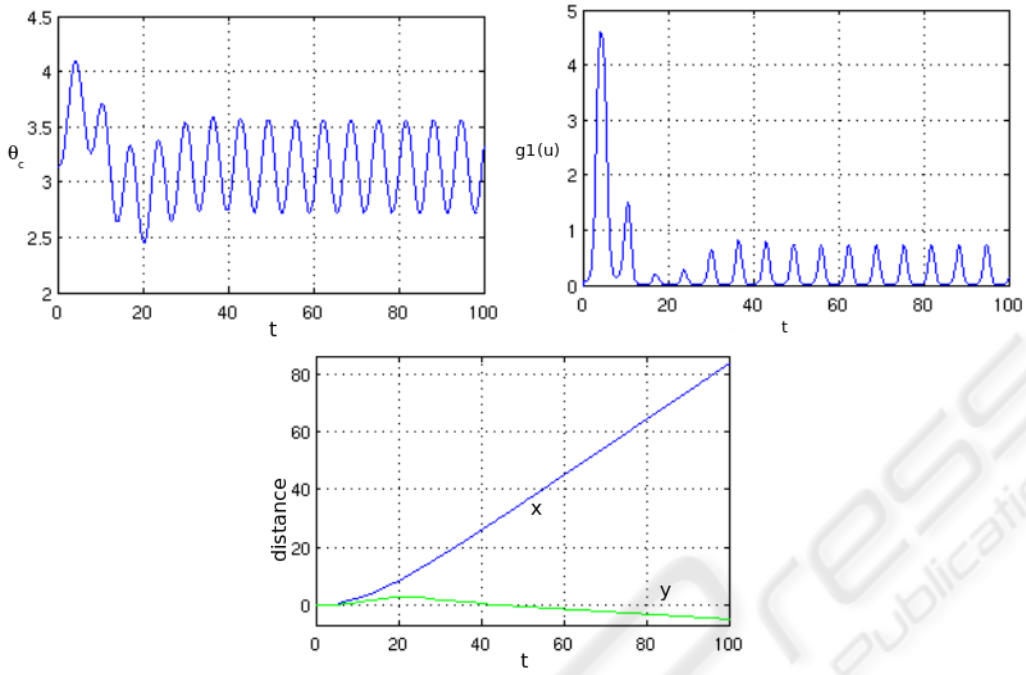

Figure 4: Simulation results when the RNN is initialized with a stronger energy. The correction is high during the transient state. The deviation is not higher than in the first trial which shows the effective action of the feedback.

Beside, the *constraint method* we used in this article can have some interest when studying the coupling of oscillatory neural networks, for e.g. to synchronize different joints. When we attach an oscillatory RNN to another one which has a constraining feedback, we can find coupling parameters which do not yield an increase of the correction in the feedback. This constraint method thus helps to reduce the space to search for suitable coupling parameters, and to better match the desired phase relationship.

REFERENCES

Buono, P. and M.Golubitsky (2001). Models of central pattern generators for quadruped locomotion. *Journal of Mathematical Biology*, 42:291–326.

Ghigliazza, R. and P.Holmes (2004). A minimal model of a central pattern generator and motoneurons for insects locomotion. *SIAM Journal on Applied Dynamical Systems*, 3(4):671–700.

Ijspeert, A. (2001). A connectionnist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander. *Biological Cybernetics*, 84:331–348.

Ishiguro, A., Otsu, K., Fujii, A., Uchikawa, Y., Aoki, T., and Eggenberger, P. (2000). Evolving and adaptive controller for a legged-robot with dynamically-rearranging neural networks. In *Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior*, Cambridge, MA. MIT Press.

Jouffroy, G. and Jouffroy, J. (2006). A simple mechanical system for studying adaptive oscillatory neural networks. *IEEE International Conference on Systems, Man and Cybernetics*, pages 2584–2589.

Kailath, A. D. . T. (1990). Model-free distributed learning. *IEEE Trans. Neural Networks*, 1(1):58–70.

Kamimura, A., Kurokawa, H., Yoshida, E., Tomita, K., Murata, S., and Kokaji, S. (2003). Automatic locomotion pattern generation for modular robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 714–720.

Krishnaprasad, P. and Tsakriris, D. (1995). Oscillations, se(2)-snakes and motion control. New Orleans, Louisiana.

Mori, T., Nakamura, Y., Sato, M., and Ishii, S. (2004). Reinforcement learning for cpg-driven biped robot. *Nineteenth National Conference on Artifical Intelligence*, pages 623–630.

Taga, G. (1994). Emergence of bipedal locomotion through entrainment among the neuro-musculo-skeletal system and the environment. *Physica D*, 75:190–208.

Tsung, F. and Cottrell, G. (1993). Phase-space learning for recurrent networks. Technical Report CS93-285, Dept. Computer Science and Engineering, University of California, San Diego.

Weiss, M. (1997). Learning oscillations using adaptive control. *International Conference on artifical Neural Networks*, pages 331–336.