A Chaotic Neural Network as Motor Path Generator for Mobile Robotics

Michele Folgheraiter, Giuseppina Gini

***Abstract*— This work aims at developing a motor path generator for applications in mobile robotics based on a chaotic neural network. The computational paradigm inspired by the neural structure of microcircuits located in the human prefrontal cortex is adapted to work in real-time and used to generate the joints trajectories of a lightweight quadruped robot. The recurrent neural network was implemented in Matlab and a software framework was developed to test the performances of the system with the robot dynamic model. Preliminary results demonstrate the capability of the neural controller to learn period signals in a short period of time allowing adaptation during the robot operation.**

Keywords: Recurrent Neural Network RNN, Dynamic Neural Network, Control Path Generator, Lightweight Quadruped Robot, Neurodynamics.

I. INTRODUCTION

In vertebrate the deambulation represents a fundamental skill that is necessary to move in the environment in order to find food, to reach locations that offer better life conditions, to escape eventual predators, and to find partners for reproduction. The motor signals that control each muscle and joint of the legs are generated by the animal's nervous system. In particular, specialized neural circuits called Central Pattern Generators (CPGs) are able to form rhythmic patterns without the need of any sensory feedback [1]. The CPG paradigm was successfully used in robotics since the early 1990's, extensive work regarding the locomotion of hexapod robots was done by Randall Beer and colleagues [2], following different other works were inspired by the locomotion of insects [3][4][5]. In [6] the neural circuit of a salamander is reproduced in order to obtain both aquatic and terrestrial gates.

The concept of CGP can also be adapted to mobile robots that are equipped with wheels, in [7] a controller was implemented able to perform different concurrent behaviors, like reaching a sound source, avoiding obstacles and finding a recharge station.

Of particular interest to generate motor paths are artificial recurrent neural networks (RNN) that are strongly inspired by the structure of natural neural circuits, where the recursion of excitatory and inhibitory feedback connections is extensively present [8]. This bioinspired computation paradigm is very powerful and can be used for a large variety of complex tasks, among them e.g. the approximation of nonlinear dynamic systems, the recognition of temporally extended patterns [9], and the learning of precise timing sequences [10]. The main issue for RNNs dwells in the training phase, classical learning algorithms tend to be trapped in local minima, suffer of vanishing learning signal when the error is propagated "back in time " [11], and in general their dynamic behavior and stability is quite complex to study especially when the architecture comprises more than few neurons.

More recently a new class of RNNs has attracted particular attention in the scientific community. They are referred as *Liquid State Machine* or *Echo State Neural Networks* [12] and consist of a relatively big set of hidden neurons which synaptic connections are randomly initialized and kept constant during the entire learning phase. It is demonstrated that if the time constants of the different neural units are properly initialized and the number of neurons is sufficiently large ($50 - 1000\, units$) the circuit is endow of rich dynamic movements that can be harnessed to solve complex non linear dynamic problems. This is generally performed by readout units that apply simple linear regressions of pulls of neuronal outputs.

It has been shown that if the output of the readout units are feedback to other neurons of the neural circuit it is possible to learn complex periodic signals [13]. This feature can be e.g. useful to learn optimal motor paths for mobile robotic systems. Generally this requires that the target signal is available to the learning algorithm in order to compute the adaptation step. However, in [14] it is shown that this constrain can be removed with the introduction of a reward-modulated Hebbian learning rule that is based on a weak information about the performance of the neural network. In particular, only a binary on-off signal is used to modulate the synaptic plasticity that is based on the correlation between the readout neuron output and its inputs. This approach is of particular interest because resembles what normally happen in a dynamic biological environment where most of the times it is not clear which of the past motor actions performed by the agent (the biological organism) contributed to increase its actual reward. This is also refereed as temporal credit-assignment problem [15] and in order to be solved it is necessary that the agent keeps track not only of its internal state, but also of the environment state.

With this work we propose a joint trajectories generator based on a chaotic RNN intended to control the motion of a light-weigh quadruped robot. In order to allow the implementation of the control system on relatively small computational units (e.g. a micro-controller), we simplified

M. Folgheraiter is with the Robotics and Mechatronics Department, SST Nazarbayev University, 53 Kabanbay Batyr Ave, Astana, Kazakhstan (phone: +7 7172 709063; e-mail: michele.folgheraiter@nu.edu.kz).

G. Gini is with the DEIB Department, Politecnico di Milano, Piazza Leonardo da Vinci 32, Milano, Italy (e-mail: gini@elet.poli.it).

as much as possible the network architecture without losing generality and scalability. The main advantages of this approach is the possibility to approximate complex spatial and temporal motor paths, to allow real-time computation of the adaptation step, to avoid being trapped in local minima during the training phase, and to allow extending the control architecture dynamically during the operation of the robot without completely reconfiguring the network architecture.

The rest of this paper is organized as follow: next section introduces the network architecture and details the neural model. Section III presents the robot structure and the simulation environment, section IV reports the chosen parameters and brings preliminary results. Finally, section V draws the conclusions and suggests some possible future research directions.

II. THE NEURAL MODEL

A very important feature for a mobile robot is the capability to learn new behaviors by detecting and modeling new associations between the stimuli received by its sensory system and the actions performed by its actuation system. This task can be carry out by a neural network and a proper learning algorithm and generally it is quite straightforward when the action and the correlated stimuli are temporally near.

The implemented recurrent neural network (see Fig. 1) consists of N neurons modeled by a first order differential equation 1 where x_i represents the membrane potential, τ_i the time constant, C a parameter that modulates the dynamic regime of the entire ANN from ordered to chaotic, r_j and u_j, the output and input of the j^{th} neuron respectively, and z_j the signal generated by the readout unit. In particular a sigmoidal activation function (Eq. 3) is used as non linear element in the neuron's model that limits the output in the range $[-1, 1]$.

$$\tau_i \dot{x}_i(t) + x_i(t) = C \sum_{j=1}^{N} w_{ij}^{c} r_j(t) + \sum_{j=1}^{L} w_{ij}^{fb} z_j(t) + w_i^{in} u_i(t) \tag{1}$$

Gaussian noise with zero average and variance $var = 0.05$ is added to the neurons output and the readout units (Eq. 2 and Eq. 4 respectively) as a mechanism of probabilistic inference [16].

$$r_j(t) = \Phi(x_j(t)) + GNoise_j(t) \tag{2}$$

$$\Phi(x) = \frac{1 - e^{-kx}}{1 + e^{-kx}} \tag{3}$$

The connections between the N neurons were represented by a N–by–N sparse matrix of real numbers. The probability of connection between two neurons is $P = 0.1$ and the weights were randomly initialized in the range $[-1, 1]$ with 20% of inhibitory synapses and 80% of excitatory one. The neural network has two kind of inputs: the external input $\mathbf{u}(t)$ that influences the neuron potentials according to the weights vector $\mathbf{W^{in}} \in \mathbb{R}^N$ and the signal feedback from the readout neurons that is scaled by a L–by–N matrix $\mathbf{W^{fb}}$. The readout units $\mathbf{z}(t)$ compute a linear combination (Eq. 4) of the neurons output $\mathbf{r}(t)$ by using a N–by–L matrix $\mathbf{W^{Ad}}$. This is the only parameter of the RNN that is adapted during learning.

$$\mathbf{z}(t) = W^{Ad}\mathbf{r}(t) + GNoise(t) \tag{4}$$

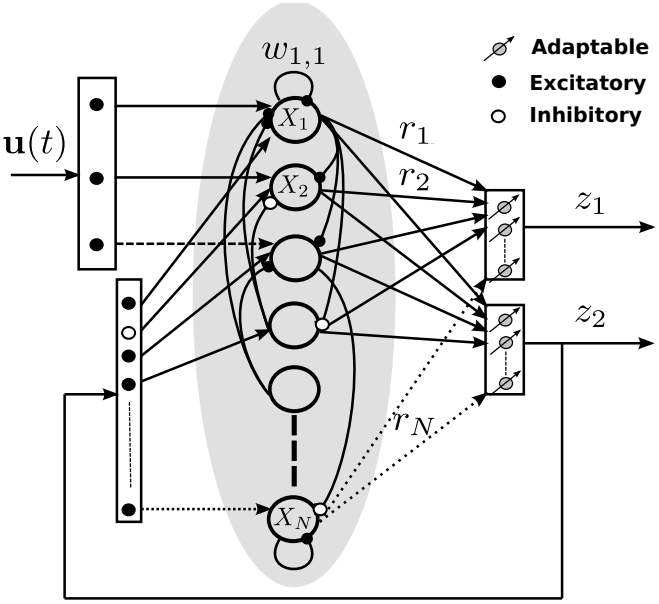

Fig. 1. The architecture of the chaotic recurrent neural network.

Learning is implemented with a simple rule that is based on the error calculated by the difference between the filtered version $\mathbf{z}(t)$ of the readout unit vector and the target vector $\bar{\mathbf{z}}(t)$ at the instant t as in equation 5.

$$\mathbf{Err(t)} = F(\mathbf{z}(t)) - \bar{\mathbf{z}}(t) \tag{5}$$

Where the filter $F(s)$ is defined as the transfer function in Eq. 6, that has unit gain and where a allows to set a proper cutting frequency.

$$F(s) = \frac{a}{s + a} \tag{6}$$

The weight matrix $\mathbf{W^{Ad}}$ is adapted according to rule represented in equation 7

$$\mathbf{W^{Ad}}(t+1) = \mathbf{W^{Ad}}(t) + \eta F(\mathbf{r}(t))\mathbf{Err(t)} \tag{7}$$

,where η defines the speed of the learning process that can be kept constant during the training phase or can decay over time. Particular attention should be paid to initialize this parameter, too large values will make the readout units

to follow rapidly the target, but without allowing the RNN to learn from the data. On the contrary too small values of η will make the learning process very slow avoiding the converge to the target signal in a reasonable amount of time.

III. THE TARGET ROBOT AND THE SIMULATION ENVIRONMENT

The robot we developed consists of four legs with three degrees of freedom each (Fig. 2). The mechanical parts were made of polylactide (PLA) and they were developed using a 3D printing process. Each of the proximal joints are equipped with bearings in order to reduce the friction and the load at the servomotor output shaft. Distal joints (knees) are directly connected to very light digital servomotors (weighting 3 grams each). The motors are controlled with a $32-bit$ micro-controller belonging to the family $STM32F4xxx$ produced by the STMicroelectronics company and running at $180MHz$ with a computational power of $225DMIPS$.

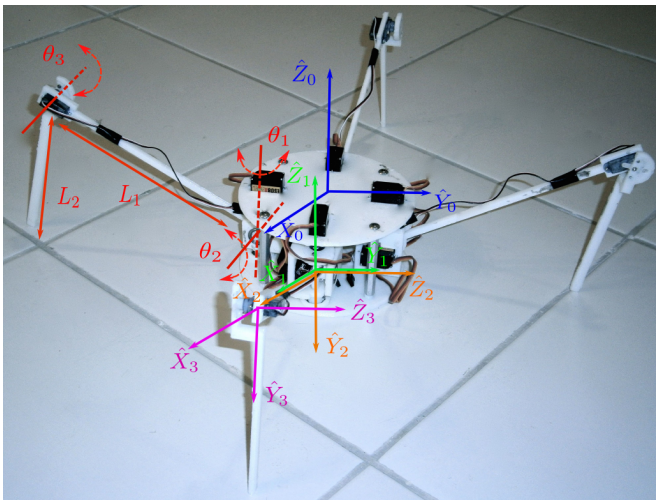

Fig. 2. The robot has a total of 12DOFs, three rotational joints for each leg which angular positions are defined by θ_1, θ_2, θ_3. The servomotors are located in the base and in the knee of each leg. Thanks to the usage of very light materials and motors the weight of the entire system, including the electronics, is 0.4Kg. The legs have an extension of $0.4m$ allowing the robot to overcome obstacles up to $0.1m$ in height.

The direct kinematic model of a single leg can be easily obtained by applying the Denavit-Hartenberg (D-H) convention. In particular, to affix the frames to the robot links and to obtain the D-H parameters we used the method described in [17]. Fig. 2 includes four reference systems: frame $\{0\}$ attached to the center of the robot's body is used as main reference system, the frame $\{1\}$ is attached to the Link-1 of the leg, and the frames $\{2\}$ and $\{3\}$ are attached to the Link-2 and Link-3 respectively. Finally, the foot's tip of the robot can be described as a point vector relative to frame $\{3\}$.

The D-H parameters relative to a single leg are reported in the table I.

The forward kinematic can be obtained substituting the D-H parameters in the homogeneous transform T_{i+1}^{i} that

Joint-i	α_{i-1}	$\mathbf{a_{i-1}}$	$\mathbf{d_i}$	θ_i
1	0^0	a_i	d_1	θ_1
2	-90^0	0	0	θ_2
3	0^0	L_1	0	θ_3

TABLE I
D-H PARAMETERS OF THE KINEMATIC MODEL OF A ROBOT'S LEG.

describes frame $\{i+1\}$ with respect to frame $\{i\}$. Applying the pre-multiplication rule and from simple algebraic passages it is possible to obtain the matrix T_3^0 as in Eq. 8. This represents the orientation and the position of frame $\{3\}$ relative to frame $\{0\}$. Where L_1 and L_2 are the length of the two leg's segments (see Fig. 2) and a_0 and d_1 the parameters that define the relative position of the frame $\{1\}$ origin with respect to frame $\{0\}$.

$$T_3^0 = \begin{bmatrix} c_1c_2c_3 - c_1s_2s_3 & -c_1c_2s_3 - c_1s_2c_3 & -s_1 & L_1c_1c_2 + a_0 \\ c_3s_1c_2 - s_1s_2s_3 & -s_3c_2s_3 - s_1s_2c_3 & c_1 & L_1s_1c_2 \\ -s_2c_3 - c_2s_3 & +s_2s_3 - c_2c_3 & 0 & L_1s_2 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

In order to test the neural architecture before the final implementation a simulation environment was developed in Matlab/Simulink (Fig. 3). The kinematic and dynamic properties of the robot's legs are represented with the toolbox SimMechanics. The joints are controlled in position using a set of PID controllers that receive as input the position error and generate as output the reference torque for the actuator. Virtual sensors are integrated to allow monitoring different kinematic quantities like the position, the velocity, and the acceleration of each joint. Furthermore, additional sensors are added to detect the Cartesian position of different points of the kinematic structure. The possibility to measure the torques and the velocities at each joint allow us to calculate the power consumed to produce specific trajectories. This is important if the adaptation mechanism has as target the minimization of the energy consumption and therefore the improvement of the walking efficiency.

The SimMechanics toolbox does not allow to detect and to simulate the contact between objects; however, this feature is particularly important to study different robot gaits and the adaptation behaviors. In order to overcome this limitation we implemented the contact model using the possibility to apply external forces and torques to different points of the kinematic structure. In particular the floor and the obstacles were modeled with equation 9 where P_z^{foot}, $\bar{P}_z^{foot}$, and V_z^{foot} represent the z coordinate of the actual and reference position and the actual velocity of the foot respectively. The constant K_P and K_V set the elastic and damping behavior of the contact respectively. By choosing proper combinations of these parameters it is possible to simulate different kind of materials, i.e. from soft to very rigid one.

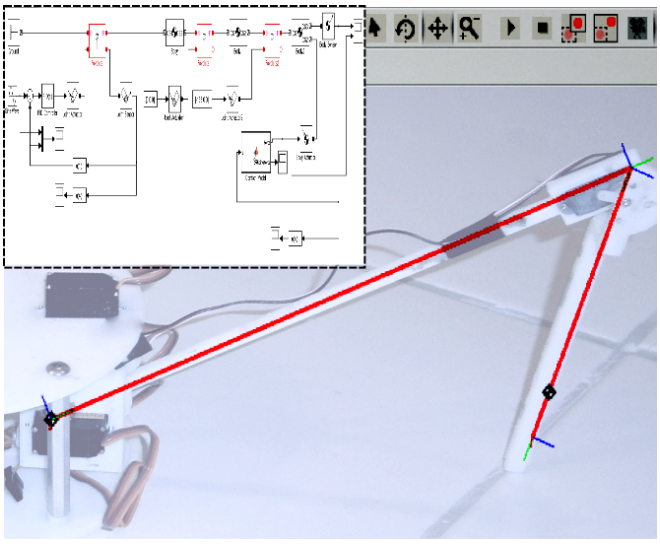

Fig. 3. Dynamic model developed with the toolbox SimMechanics.

$$F_z = K_P(P_z^{foot} - \bar{P}_z^{foot}) - K_V(V_z^{foot}) \quad (9)$$

The implementation of the software for the robot prototype will be based on the Simulink toolboxes "RapidSTM32" and "Real Time Workshop". By using these tools it will be possible to convert the symbolic code developed in Simulink directly into binary code suitable to be flashed in the micro-controller. This is very important in order to bootstrap the developing phase and to allow an easy tuning of the control system.

IV. PARAMETERS TUNING AND PRELIMINARY RESULTS

In order to test the RNN and to fine-tuning the main parameters of the neural circuit we defined a set of periodic trajectories that resemble a possible motor path for the joints of the robot. In particular, as a target function a possibility is to choose a non linear combination of k sinusoidal signals having different frequencies w_j and phases ϕ_j (Eq. 10).

$$z_i(t) = f(\sum_{j=1}^{K} m_j(sin(w_j t + \phi_j))) \quad (10)$$

The main network's parameters are represented in table II. As integration method for the neuron's potential we chose $Dormand - Prince$ (available in the Simulink library) with a variable integration step. However, it is worth to mention here that for the final real-time implementation it will be necessary to choose a fixed integration step that will be comparable with the chosen computational time. The computational time is a very important parameter for the learning algorithm. This represents the frequency with which the adaptable synapses are updated and strongly depends on the rapidity of the signals we want to reproduce (in our case represented by the periodic signals we want to learn).

TABLE II

THE NEURAL NETWORK PARAMETERS.

Quantity	Value
Number of Neurons N	10-50
Number of External Inputs	2
Number of ReadOut Units L	1
Computational Time	variable
Neuron Time Constants τ	0.01-2.5
Learning Constant η	0.00001-0.0001
Filters Constant a	1
Chaos-Modulation Constant C	0.5-2

The first test we conducted on the implemented RNN was to verify the capability to modulate the level of the chaotic behavior. For this particular experiment we changed the constant C in a range of values between 0.5 (completely ordered) to 2 (chaotic). According to our experiments the more the network is chaotic the richer is the dynamics it can exhibit. However, for very small networks, like the one we simulated, it will result more difficult to learn periodic signals due to the fact that the state of the network is always changing.

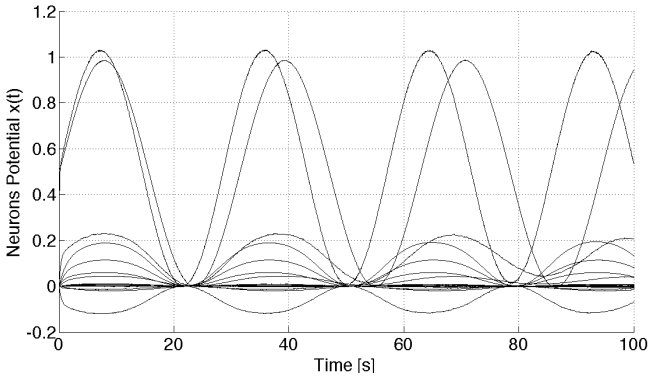

Fig. 4. Neurons Potential with chaos-modulation constant C=0.5

In figure 4 we can see the potential of the neurons. In this example a network consisting of $20-neurons$ was simulated where the chaos-modulation constant was settled at a low value $C = 0.5$ and only a sinusoidal input with frequency $f = 0.01rad/s$ was used to stimulate the dynamic system. How it is possible to notice the neurons' potential oscillate with almost constant amplitudes.

Figure 5 depicts instead the state of the network in the case the value of the chaos-modulation constant is increased to $C = 2$. Due to the fact that now the recurrent feedbacks of the neurons is amplified the RNN assumes a more chaotic behavior, it is possible to notice how this time the neurons potential oscillate with more irregularity. Notice also that this time the range of the potential amplitudes is larger in comparison with the previous case. This does not represent a problem due to the fact that thanks to the activation function the neurons output is limited in the range $[-1, 1]$. Figure 6 reports the outputs of the 20 neurons overlapped with the Gaussian noise.

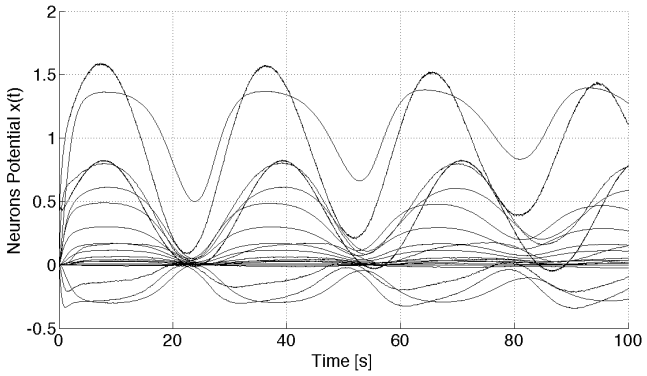

Fig. 5. Neurons Potential with chaos-modulation constant C=2

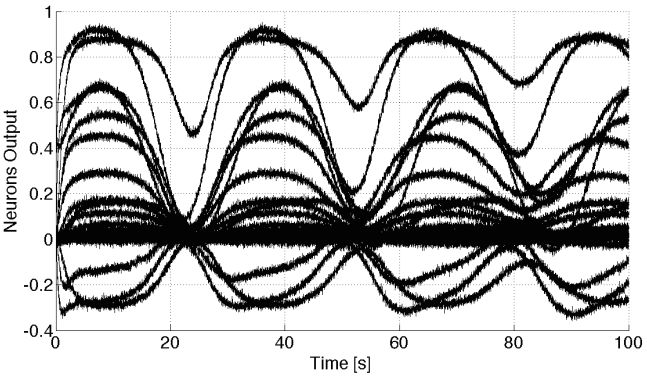

Fig. 6. Neurons Output with chaos-modulation C=2

To test the capability of the RNN to learn and reproduce periodic signals we started using a simple target periodic function as in equation 11.

$$\bar{z}(t) = 0.5sin(0.2t) + 0.5sin(0.22t) \tag{11}$$

The dimension of the network this time was increased to $N = 50$, the training period was fixed to $600s$ and the learning constant was chosen as $\eta = 0.0001$. The only readout unit collects and adapts the last 45 neural outputs. After the training the RNN was tested to verify the ability to reproduce autonomously the periodic signal. How it is possible to notice in figure 7 the synapses of the readout unit are converging toward a stable value. Small oscillations are present during the learning process, and generally this should be avoided reducing the learning constant. However, for this first set of simulations we were more concerned in assessing the model functionality rather than optimizing its parameters.

From the simulation it is also clear that not all the synapses are in this case needed to approximate the target function, this is due to the fact that the complexity of the periodic signal we want to reproduce is relatively low. From the graph it appears that when the adaptation was stopped the synapses did not reach their optimal values, indeed a residual error

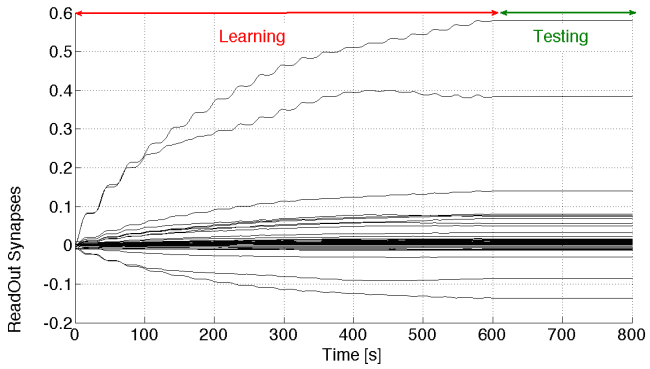

Fig. 7. Synapses adaptation.

still exist from the comparison with the target signal. Finally figure 8 depicts the output of the readout unit, its filtered version (red line in Fig. 8) and the target signals. How it is possible to notice after the learning phase terminates the neural network is still able to reproduce the target signal.

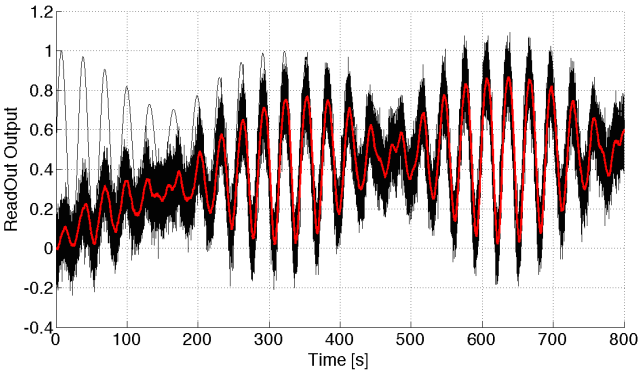

Fig. 8. Readout output (red line), target function (black line).

Another experiment we tried was to modify the learning mechanism (see Eq. 14). This time, instead of using an adaptation rule based only on the error (calculated as in Eq. 7) we introduced an additional modulating signal M which can assume a binary value that is calculated as in equation 13 and according to the performance criterion evaluated as in 12. Notice that the performance criterion is calculated on the base of the unfiltered readout output, this has the effect to introduce a stochastic behavior also in the synapses adaptation.

$$Perf = -(\mathbf{z}(t) - \bar{\mathbf{z}}(t))^2 \tag{12}$$

The network dimension this time was reduced to 10 neural units, the learning time interval extended to $1400s$, the learning constant reduced to $\eta = 0.00001$, and the chaos-modulation constant set to $C = 1.5$. How it is shown in Fig. 9 the synapses are now converging more smoothly to the final value. This reflects also in a better approximation of the target signal as it is shown in Fig. 10.

$$\begin{aligned}&\text{if } (Perf > F(Perf))\\&\quad M = 1\\&\text{else}\\&\quad M = 0\end{aligned} \tag{13}$$

$$\mathbf{W^{Ad}}(t+1) = \mathbf{W^{Ad}}(t) + \eta \mathbf{r}(t)\mathbf{Err}(\mathbf{t})M \tag{14}$$

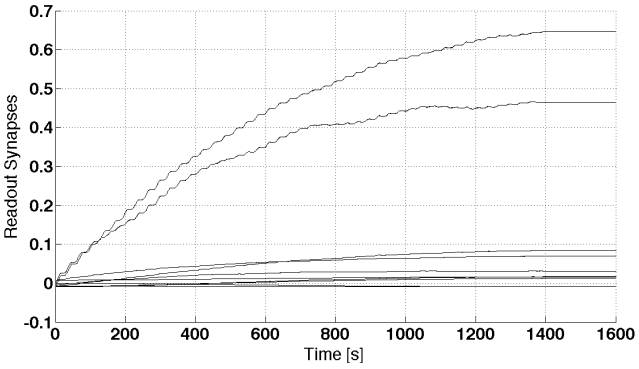

Fig. 9. Synapses adaptation with the second learning mechanism.

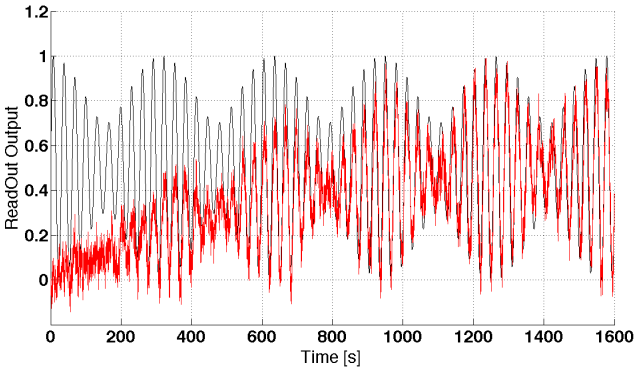

Fig. 10. Readout output (red line), target function (black line).

V. CONCLUSIONS AND FUTURE WORK

In this work we applied a chaotic RNN to develop a control system for a quadruped robot. This new computational paradigm reproduces the structure of neural microcircuits present in the vertebrates cortex. The main advantages in comparison with more classical RNN architectures is represented by the fact that the ANN is easy to train and less prone to get trapped in local minima of the error function. The control system of the robot can be adapted and optimized in real-time without requiring a off-line training phase. Furthermore, thanks to a very simple learning mechanism new readout units can be added dynamically during the robot operation and without requiring to modify the ANN main architecture. Preliminary results demonstrated that even in the case of small networks it is possible to reproduce period signals that can be used as reference joint trajectories for walking. Future work will be dedicated to more complex control tasks that may require as additional inputs the information coming from sensors capable to detect the contact of the leg with the floor and/or eventual obstacles.

REFERENCES

[1] F. Delcomyn, "Neural basis of rhythmic behavior in animals." *Science*, no. 210, pp. 492–498, 1980.

[2] R. D. Beer, H. J. Chiel, R. D. Quinn, K. S. Espenschied, and P. Larsson, "A distributed neural network architecture for hexapod robot locomotion," *Neural Computation*, vol. 4, no. 3, pp. 356–365, May 1992. [Online]. Available: http://dx.doi.org/10.1162/neco.1992.4.3.356

[3] D. Spenneberg and F. Kirchner, "Scorpion: A biomimetic walking robot," *VDI BERICHTE*, vol. 1679, pp. 677–682, 2002.

[4] H. Cruse and C. Bartling, "Movement of joint angles in the legs of a walking insect," *Journal of Insect Physiology*, vol. 41, no. 9, pp. 761–771, 1995.

[5] B. Webb and T. Consilvio, *Biorobotics*. Mit Press, 2001.

[6] A. J. Ijspeert, "A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander," *Biological Cybernetics*, vol. 84, no. 5, pp. 331–348, 2001. [Online]. Available: http://dx.doi.org/10.1007/s004220000211

[7] M. Folgheraiter, G. Gini, A. Nava, and N. Mottola, "A bioinspired neural controller for a mobile robot," in *Proceedings of IEEE Robio06*, Kunming (China), December 17-20 2006.

[8] J. J. Hopfield, "Learning algorithms and probability distributions in feed-forward and feed-back networks," *Proceedings of the National Academy of Sciences*, vol. 84, no. 23, pp. 8429–8433, 1987.

[9] S. H. J. Schmidhuber, "Lstm can solve hard long time lag problems," in *Advances in Neural Information Processing Systems 9: Proceedings of the 1996 Conference*, vol. 9. MIT Press, 1997, p. 473.

[10] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with lstm recurrent networks," *The Journal of Machine Learning Research*, vol. 3, pp. 115–143, 2003.

[11] P. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct 1990.

[12] H. H. Jaeger, H., "Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication," *Science*, 2004.

[13] W. Maass, P. Joshi, and E. D. Sontag, "Computational aspects of feedback in neural circuits," *PLoS Comput Biol*, vol. 3, no. 1, p. e165, 01 2007. [Online]. Available: http://dx.plos.org/10.1371%2Fjournal.pcbi.0020165

[14] G. M. Hoerzer, R. Legenstein, and W. Maass, "Emergence of complex computational structures from chaotic neural networks through reward-modulated hebbian learning," *Cerebral Cortex*, vol. 24, no. 3, pp. 677–690, 2012.

[15] R. Sutton, U. of Massachusetts at Amherst. Department of Computer, and I. Science, *Temporal Credit Assignment in Reinforcement Learning*, ser. COINS technical report. UMI, 1984. [Online]. Available: http://books.google.it/books?id=wE2lSgAACAAJ

[16] W. Maass, "Noise as a resource for computation and learning in networks of spiking neurons," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 860–880, May 2014.

[17] J. J. Craig, *Introduction to Robotics: Mechanics and Control*. Pearson, 2014.