Variable Structure Neural Networks for Real Time Approximation of Continuous Time Dynamical Systems Using Evolutionary Artificial Potential Fields

HASSEN MEKKI[1,2], MOHAMED CHTOUROU[1]

[1]Intelligent Control, design and Optimization of complex system (ICOS),
University of Sfax, Tunisia
[2]National School of Engineering of Sousse,
University of Sousse, Tunisia
hassen.mekki@eniso.rnu.tn; mohamed.chtourou@enis.rnu.tn

Abstract - A novel neural network architecture, is proposed and shown to be useful in approximating the unknown nonlinearities of dynamical systems. In the variable structure neural network, the number of basis functions can be either increased or decreased this is according to specified design strategies so that the network will not overfit or underfit the data set. Based on the Gaussian radial basis function (GRBF) variable neural network, an online identification of continuous-time dynamical systems is presented. The location of the centers of the GRBFs is analyzed using a new method inspired from evolutionary artificial potential fields method combined with a pruning algorithm. A minimal number of neuron is guaranteed by using this method. It is in noted, that both the recruitment and the pruning is made by a single neuron. By consequence, the recruitment phase does not perturb the network and the pruning does not provoke an oscillation of the output response. The weights of neural network are adapted so that the dynamics of the system checks the imposed performances, in particular the stability of the system.

Key-words- Variable structure neural network; Radial basis functions, Evolutionary artificial fields, Identification of dynamical systems.

1 Introduction:

Recently, neural network research has gained increasing attention. In fact, artificial neural networks are able of learning and reconstructing complex nonlinear mappings and they have been widely studied by control researchers in identification analysis and the design of control systems [6], [10], [14], [15], [16], [18], [19].
The network size, often measured by number of hidden units in a single hidden layer network, reflects the capacity of the neural network to approximate an arbitrary function. What is the required size of the neural network to solve a specific problem? the fundamental question in the design of neural networks. If the training starts with a small network, it is possible that the learning process cannot be achieved. On the other hand, if a large network is used, the learning process can be very slow and an over-fitting may occur.

The approaches, which assume a priori the number of RBFs, usually lead to the problem of poor generalization. In addition, these approaches usually work offline, so they are not suitable for practical real-time applications where the online learning is required for the neural-network-based controller design. To remedy the aforementioned shortcomings, several growing RBF networks have been proposed in [9], [11], [12], [20], [24], [27].
In using RBF networks, the basis function are placed on regular points of a square mesh, for example, covering a relevant region of space where the state is known to be contained [2], [3], [13]. This region therefore is the network approximation region, which is in general known for a given system. The distance between the points affects the number of basis functions required to cover the region and hence determines the size of the neural network.

It seems well that if the size of the neural network input vector increases, it will have an excessive increases of the neural network size which will provoke oscillation in the output responses.
To remedy theses problems, a novel neural network architecture is proposed, where the location of the centers of the GRBFs is analyzed using a new method inspired from evolutionary artificial potential field's method.
The weights of neural network are adapted so that the dynamics of the system checks the imposed performances, in particular the stability of the system.
This paper is organized as follows. Section 2 provides the design of the continuous-time dynamical system approximator. This is done by the use of multilayered neural networks for the identification of uncertain nonlinear functions. In section 3, we describe a novel self-organizing RBF network that can dynamically vary its structure in real time. The proposed self-organizing RBF network is capable of adding or removing RBFs to ensure the desired approximation accuracy and at the same time to keep the appropriate network complexity. The location of the centers of the GRBFs is analyzed using a new method inspired from evolutionary artificial potential fields method. To show the obtained performances of the proposed algorithm, section 4 presents two simulation examples. The first one deals with a based variable structure network on-line identification of a nonlinear function. The second example treats the on-line identification of nonlinear dynamical system.

2 Approximation network synthesis and analysis

The nonlinear continuous-time dynamical systems that we consider in this paper are modeled by the following vector differential equation:

$$\dot{x}(t) = f\big(x(t), u(t)\big), \qquad x(t_0) = x_0 \qquad (1)$$

Where $x(t) \in \mathbb{R}^n,\ u(t) \in \mathbb{R}^m,\ and\ f(.,.)$: $\Omega \to \mathbb{R}^n$ is a continuous vector-valued function of many variables defined on a compact set $\Omega \subset \mathbb{R}^{n+m}$ is the $[x\ u]$ space. We assume that the function f is unknown to us. Let $A \in \mathbb{R}^{n\times n}$ be a Hurwitz matrix as:

$$A = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 1 \\ -k_0 & -k_1 & -k_2 & \dots & -k_{n-1} \end{bmatrix}$$

The real numbers $k_0, k_1, k_2, \dots, k_{n-1}$ are chosen using the pole assignment technique.

We express (1), similarly as in [8], in the following form:

$$\dot{x}(t) = Ax(t) + g\big(x(t), u(t)\big), \quad x(t_0) = x_0 \qquad (2)$$

Where

$$g\big(x(t), u(t)\big) = -Ax(t) + f\big(x(t), u(t)\big) \qquad (3)$$

We then analyze an RBF network with $n+m$ inputs and n outputs that we will use to approximate the unknown function $g(x,u)$ in real-time, equivalently, the unknown dynamical system (1), as given in the fig. 1.

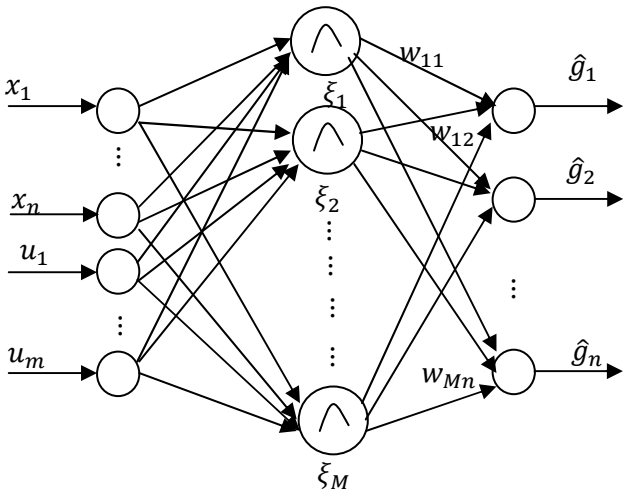


Fig. 1: RBF neural network architecture

Let:

$$\hat{g}\big(x(t), u(t)\big) = W\xi\big(x(t), u(t)\big) \qquad (4)$$

where $W \in \mathbb{R}^{n\times M}$ is a weight matrix and $\xi\big(x(t), u(t)\big) \in \mathbb{R}^M$. The RBF network approximator has the following form:

$$\dot{\hat{x}}(t) = A\hat{x}(t) + \hat{g}\big(x(t), u(t)\big), \quad \hat{x}(t_0) = \hat{x}_0 \qquad (5)$$

Note that the previous approximator has access to the system true state. We define the approximation error as:

$$e(t) = x(t) - \hat{x}(t), \quad (6)$$

Combining (5) and (6), we obtain the approximation error dynamics

$$\dot{e}(t) = Ae(t) + g\big(x(t), u(t)\big) - \hat{g}\big(x(t), u(t)\big) \qquad e(t_0) = e_0 \quad (7)$$

We assume that:

$$\left\| g\big(x(t), u(t)\big) - \hat{g}\big(x(t), u(t)\big) \right\| \leq d_g \quad (8)$$

Where $d_g > 0$ and $\|.\|$ is the Euclidean norm.

2.1 Approximation method:

The online identification method is introduced by fig. 2:

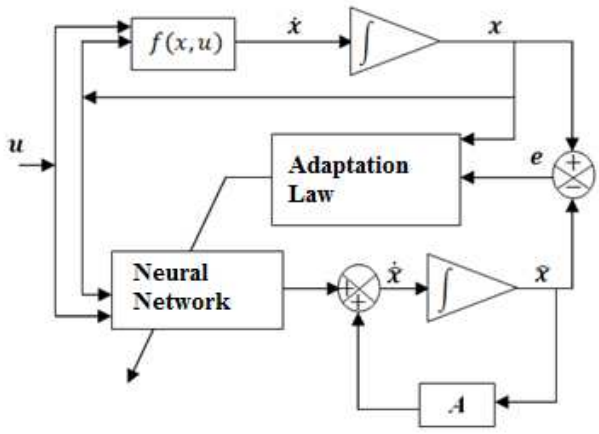


Fig. 2. Diagram of the real-time RBF network approximator.

The weights of neural network are adapted so that the dynamics of the system checks the imposed performances, in particular the stability of the system. In fact, by construction, the matrix A is Hurwitz. Then, for any given real symmetric positive-definite matrix Q, the solution P to the continuous Lyapunov matrix equation:

$$A^T P + PA = -2Q \quad (9)$$

is symmetric positive definite. Consider then the Lyapunov function candidate $V(e) = (1/2)e^T Pe$.

It's easily to demonstrate that the time derivative of V is negative definite [8].

As it is shown in [8], the adaptation law is given by:

$$\dot{W} = Proj_W(\alpha Pe\xi^T(x,u)) \quad (10)$$

It can be written in the following form:

$$\dot{w}_{kj} = \begin{cases} 0 \;\; if \;\; w_{kj} = -\omega \; et \; (Pe\xi^T(x,u))_{kj} < 0 \\ 0 \;\; if \;\; w_{kj} = \omega \; et \; (Pe\xi^T(x,u))_{kj} > 0 \\ \alpha(Pe\xi^T(x,u))_{kj} \;\; else \end{cases} \quad (11)$$

where $k = 1, \ldots, p$ and $j = 1, .., M$, w_{kj} is the weight from the j^{th} RBF to the k^{th} output neuron. $(Pe\xi^T(x,u))_{kj}$ is the element in the k^{th} row and the j^{th} column of matrix $Pe\xi^T(x,u)$. ω and α are design parameter.

3 Variable structure neural network

There are five parameters characterizing the RBF network approximation to be determined:

- The number of RBFs N
- The type of the RBF $\xi_j(x,u)$;
- The location of the center $C_{(j)}$;
- The radius in each coordinate $\sigma_{i(j)}$;
- The weight vector for each output neuron ω_k;

In this section, we present a novel RBF network structure that is capable of determining the number of RBFs N, the location of the center $C_{(j)}$ and the weight vector for each output neuron ω_k by itself. The determination of ω_k is already seen in the section 2. We first show how to determine the location of the center $C_{(j)}$. The strategy of determination of the number of RBFs needed in the proposed online identification problem will be discussed in section 3. The proposed self-organizing RBF network is capable of adding or removing RBFs to ensure the desired approximation accuracy and at the same time to keep the appropriate network complexity.

3.1 Determination of the RBFs location center

In using RBF networks, the basis function are placed on regular points of a square mesh, for example, covering a relevant region of space where the state is known to be contained [2], [3]. This region therefore is the network approximation region, which is in general known for a given system. The distance between the points affects the number of basis functions required to cover the region and hence determines the size of the neural network.
It seems well that if the size of the neural network input vector increases, it will have an excessive increases of the neural network size which will provokes oscillation in output responses.

To remedy theses problems, a novel neural network architecture, is proposed, where the location of the centers of the GRBFs, is analyzed using a new method inspired from evolutionary artificial potential field's method.

3.2 The Artificial Potential Field

This method is especially used in the real-time robot path planning. In the artificial potential field methods, a robot is considered as a particle under the influence of an artificial potential field U whose local variations reflect, for instance, the positions of obstacles and of the goal that the robot is supposed to reach [17], [21], [23], [26]. The potential field function is defined as the sum of an attraction field that pulls the robot towards the goal and a repulsive field that repels it forms the obstacles. The movement is executed in an iterative way, in which an artificial force is induced by:

$$\vec{F}(q) = -\vec{\nabla}U(q) \tag{12}$$

which forces the robot to move to the direction that the potential field decrees, where $\vec{\nabla}$ the gradient with respect to q and $q = (x, y)$ which represents the coordinates of the robot position. The complete potential field is a superposition of contributions from obstacles, waypoint (if applicable), and the goal:

$$U(q) = \sum_{j=1}^{n_0} U_j^0(q) + \sum_{j=1}^{n_w} U_j^w(q) + U^g(q) \tag{13}$$

where n_0 and n_w denote the number of obstacles and waypoints, respectively, and U_j^0 and U_j^w are their potential. U^g is the potential generated by the goal (navigation target).

In our approach, a neuron plays the role of the robot, the other neurons play the roles of the obstacles and the current true state of the system, is the goal point

Different potential functions have been proposed in literature. The most commonly used attractive potential take the form [22], [25]:

$$U_{att}(q) = \frac{1}{2}\beta\rho^m(q, q_{goal}) \tag{14}$$

where β is a positive scaling factor, $\rho(q, q_{goal}) = \|q_{goal} - q\|$ is the distance between the neuron q and the goal q_{goal}, and $m = 1$ or 2. For $m = 1$, the attractive potential is conic in shape and the resulting attractive force has constant amplitude except at the goal, where U_{att} is singular. For $m = 2$, the attractive potential is parabolic in shape. The corresponding attractive force is then given by the negative gradient of the attractive potential:

$$F_{att}(q) = -\nabla U_{att}(q) = \beta(q_{goal} - q)/\|q_{goal} - q\| \tag{15}$$

which is a constant force on the space: it does not tend to infinity with increasing distance from q_{goal}. However, it is not zero at q_{goal}.

One commonly used repulsive potential function takes the following form [1]:

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho(q,q_{neu})} - \frac{1}{\rho_0}\right)^2 & if\ \rho(q, q_{neu}) \le \rho_0 \\ 0 & if\ \rho(q, q_{neu}) > \rho_0 \end{cases} \tag{16}$$

Where η is a positive scaling factor, $\rho(q, q_{neu})$ denotes the minimal distance from the center of neuron q and the center of other neuron, q_{neu} denotes the center of the nearest neuron, and ρ_0 is a positive constant denoting the distance of influence of the neuron. The corresponding repulsive force is given by:

$$F_{rep}(q) = -\nabla U_{rep}(q)$$

$$= \begin{cases} \eta\left(\frac{1}{\rho(q,q_{neu})} - \frac{1}{\rho_0}\right)\frac{1}{\rho(q,q_{neu})^2}\nabla\rho(q,q_{neu}) \; if \; \rho(q,q_{neu}) \le \rho_0 \\ 0 \qquad\qquad\qquad\qquad if \; \rho(q,q_{neu}) > \rho_0 \end{cases} \quad (17)$$

The total force applied to the neuron is the sum of the attractive force and the sum of the repulsive force :

$$F_{total} = F_{att} + \sum F_{rep} \quad (18)$$

This determines the motion of the neuron.

The attractive and the repulsive phenomena are given in fig. 3.

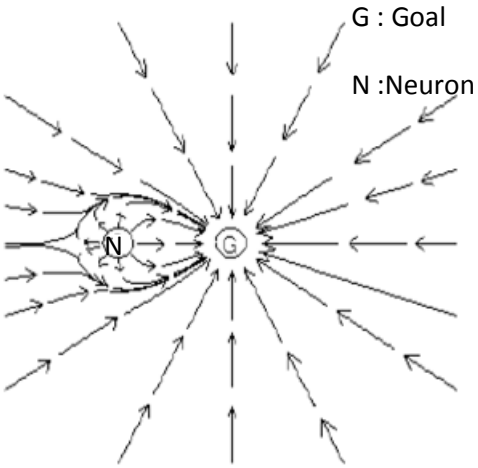


Fig. 3: The Artificial potential field

The parameters $\beta, \eta \; and \; \rho_0$ are chosen so that we obtain a scenario similar to fig. 4. We obtain concentrations of neurons in a ball of dimension $(n+m)$ centered in the desired point. By construction we obtain several layers (or orbit) of radius $r_i \approx i.\sigma$ were i the rank of the orbit and σ is the width of the GRBF.

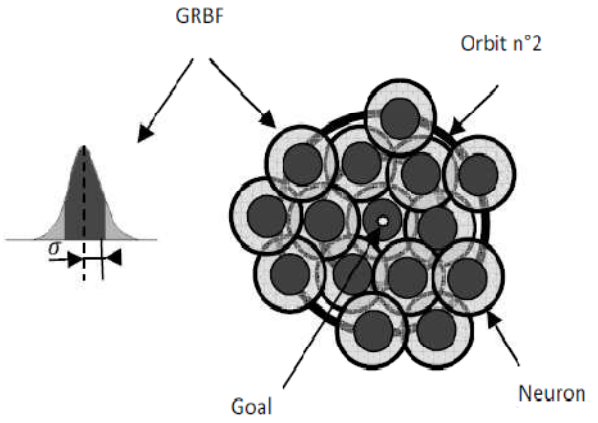


Fig. 4: Location of the GRBF in the space $(dim = 2)$

3.3 Determination of the number N

The proposed self-organizing RBF network is capable of adding or removing RBFs to ensure the desired approximation accuracy and at the same time to keep the appropriate network complexity.

3.3.1 Adding RBFs

As the system trajectory evolves in time, the approximation error e is measured. We first check if the Euclidean norm of the approximation error $\|e\|$ exceeds a predetermined threshold e_{max}, and the period between the two adding operations is greater than the minimum response time T_r. e_{max} and T_r are design parameters. If these two conditions are satisfied, we recruit a new neuron which will be placed on a neighborhood of the last orbit. It is noted, that the recruitment is made by a single neuron. Consequently, the recruitment phase does not perturb the network.

3.3.2 Removing RBFs,

The RBF removing operation is also implemented sequentially for all N coordinates. We first measure the approximation error e. If the Euclidean norm of the approximation error $\|e\|$ is smaller than τe_{max}, where $\tau \in]0,1[$ is a design parameter, we remove a neuron from the last orbit. It is in noted, that the pruning is also made by a single neuron, which aims at not provoking an oscillation of the output response.

3.4 Self-Organizing RBF Network Algorithm

Choose the design parameters $A, Q, d_{threshold}$,

$e_{max}, \tau, T_r, T_d, \alpha, and\ \omega$. Initialize some GRBFs in a neighborhood of the initial condition and the weight matrix W of the initial RBF network. In each sampling period, repeat the following steps.

1) Compare the current true state x of (2) and the current approximated state $\hat{x}$ of (5) to obtain the approximation error $e = x - \hat{x}$.
2) If $\|e\| > e_{max}$ and the period between two adding operation is greater than T_a, go to 3); otherwise, go to 4).
3) Add a new neuron which will be placed on a neighborhood of the last orbit. The radius of this orbit is given by: $r_{i_{last}} = (i_{last} + 1).\sigma$; where i_{last} is the rank of the last layer.
4) If $\|e\| \leq \tau.e_{max}$, go to 5); otherwise go to 6).
5) Remove a neuron from the last orbit
6) Update the weight matrix W using (11).
7) Compute $\hat{g}$ using (4) in the uniform grid, and then evaluate the approximated state $\hat{x}$ using (5).

8) Determination of the motion of the GRBF in the space using (18).

4 Simulation results

In this section, we test our proposed real-time self-organizing RBF network approximator on two examples: the first one deal with the on-line identification of a nonlinear function using the gradient method. The second example treats the real-time approximation of nonlinear dynamical system using the adaptation law given in equation (11).

4.1 On-line identification of a nonlinear function

For this problem, the considered nonlinear function is described by the following equation:

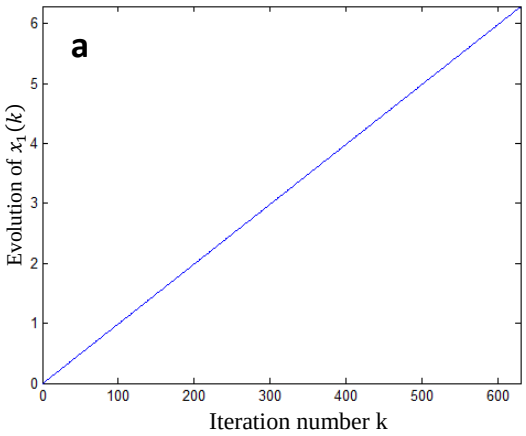
$$y(x_1, x_2) = x_2 \cos(x_1) + 2sin(x_1) \qquad (17)$$

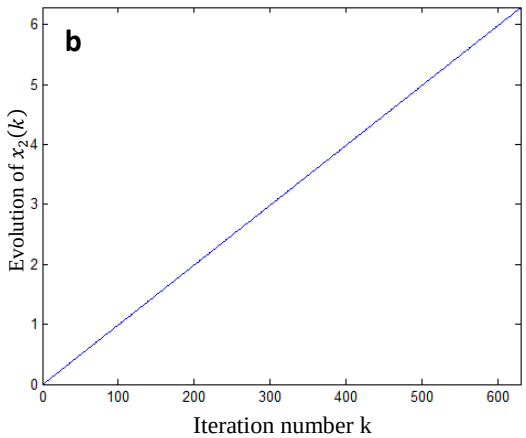
The aim is to seek an optimal number of RBF using the proposed algorithm, to approximate adaptively the function with small error.
Based on simulation studies, the parameters of the proposed algorithm are chosen as follows:

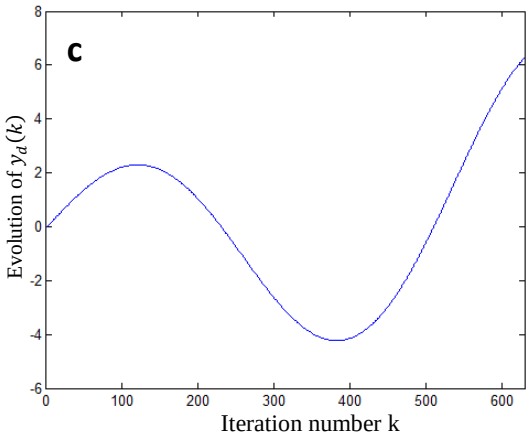
$$e_{max} = 0.1,\ \ \tau = 0.5,\ \ \sigma = 0.5,$$
$$\rho_0 = 2.\sigma,\ \ \eta = 1.5$$

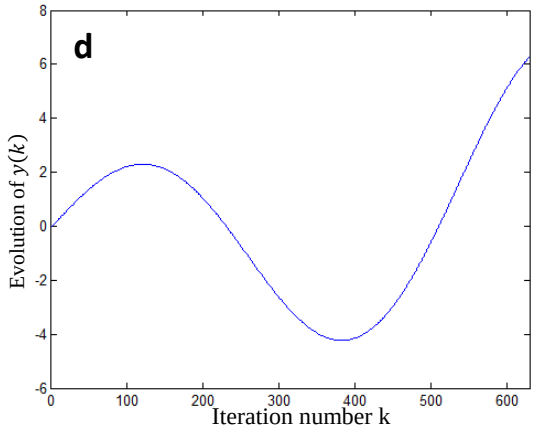
The initial number of the hidden units is chosen as $N = 4$.

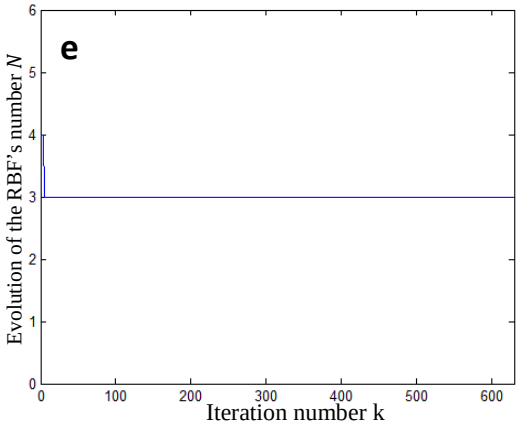
Fig. 5 shows the simulation results of the process using the proposed algorithm. Fig. 5.(a) and fig. 5.(b) show respectively, the evolution of $x_1(t)$ and $x_2(t)$. The evolutions of the desired function and its estimate are given in fig. 5.(c) and fig. 5.(d) respectively. It is clear that the approximation error is so acceptable. Fig. 5.(e) represents the evolution of the hidden units number. It's noted that we used a minimal number of neuron on this simulation. Fig. 5(f) represents the localization of the centers of the used RBFs. It's obvious that the centers are joined together all around the state (x_1, x_2) to be estimated.











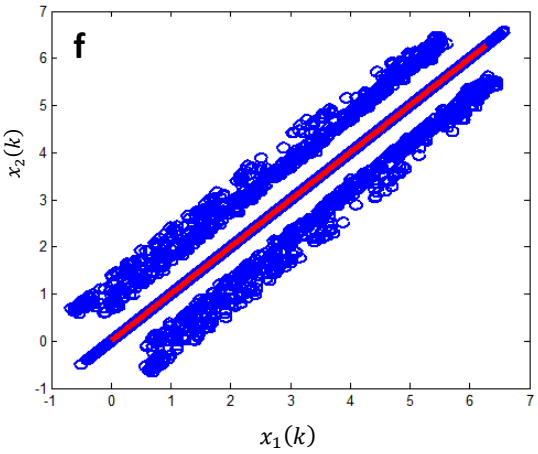


Fig. 5. (a) and (b) Evolution of x_1 and x_2 respectively. (c) and (d) Evolution of the desired function and its approximation respectively. (e) Evolution of the RBF's number. (f) Location of RBF center (o) and state (-)

4.2 Online identification of nonlinear dynamical system

This example illustrates a one-link rigid robotic manipulator. The dynamic equation of the one-link rigid robotic manipulator is given by [5]:

$$ml^2\ddot{q} + d\dot{q} + mlgcos(q) = u \qquad (18)$$

where the link is of length l and masse m, and q is the angular position with initial value $q(0) = 0.1$ and $\dot{q}(0) = 0$.
The parameters $m = 1, l = 1,\ d = 1,$ and $g = 1$.
The above dynamical equation can be written as the following state equation:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_2 - cos(x_1) + u \end{cases} \qquad (19)$$

Based on simulation studies, the parameters of the proposed algorithm are chosen as follows:

$$A_c = \begin{pmatrix} 0 & 1 \\ -1 & -2 \end{pmatrix}, Q = \begin{pmatrix} 1.9 & 0 \\ 0 & -1.9 \end{pmatrix}, \alpha = 2,$$
$$\omega = 0.1, e_{max} = 0.1, \tau = 0.5, \sigma = 0.5,\ \eta = 1.5$$

The initial number of the hidden units is chosen as $N = 9$.

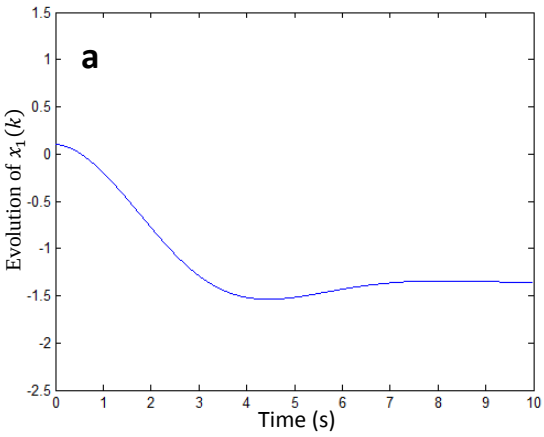
Fig. 6 shows the simulation results of the process using the proposed algorithm. Fig. 5.(a) and fig.

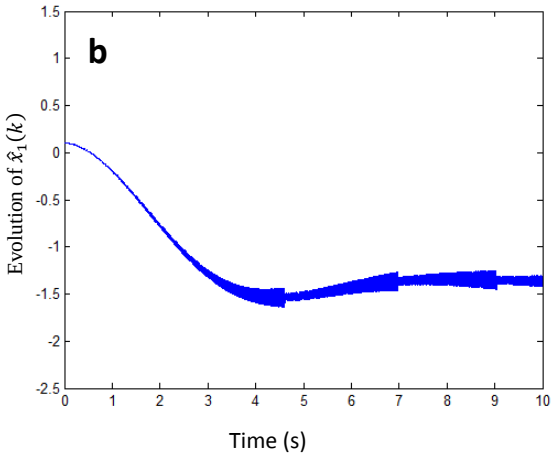
5.(b) show the evolution of the state $x_1(t)$ and its approximation respectively. The evolutions of the real state $x_2(t)$ and its estimate are given in fig. 5.(c) and fig. 5.(d) respectively. It is clear that the approximation error is acceptable. The chattering presented in the curve of the evolution of $x_1(t)$ and $x_2(t)$ is due to the use of the given identification algorithm. Fig. 5.(e) represents the evolution of the hidden units number. It's clear that we used a minimal number of neuron on this simulation.

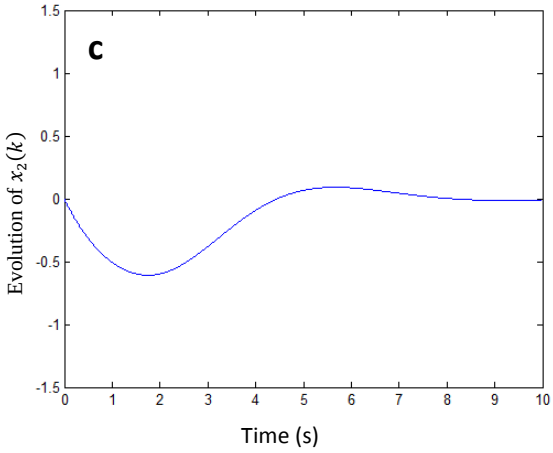
Comparing with other works [3], [5], [7], [8], it is so clear that we used a minimal number of neuron while respecting the imposed performances. We can notice, too, that the recruitment is made by a single neuron. As a consequence, the recruitment phase does not perturb the network.

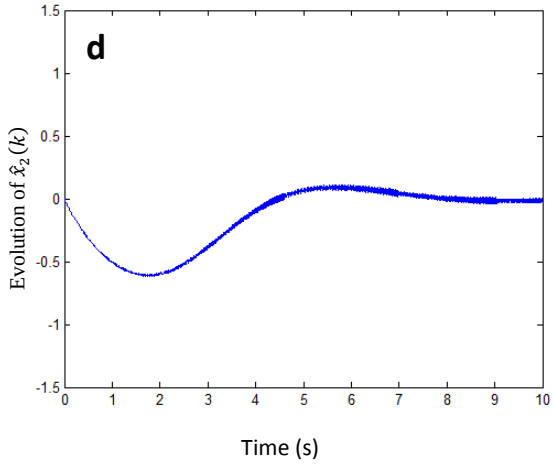
5 Conclusion

An online identification of continuous-time dynamical systems is presented. A novel neural network architecture, is proposed and shown to be useful in approximating the unknown nonlinearities of dynamical systems. In the variable structure neural network, the number of basis functions can be either increased or decreased with time according specified design strategies so that the network will not overfit or underfit the data set. Based on the Gaussian radial basis function of variable neural network, the location of the centers of the GRBFs is analyzed using a new method inspired from evolutionary artificial potential fields method combined with a pruning algorithm.









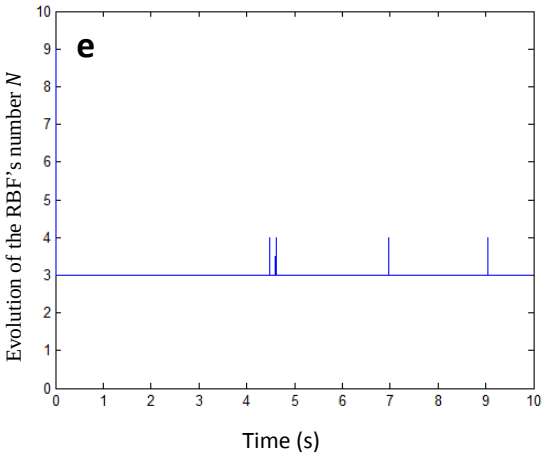


Fig. 6. (a) and (b) Evolution of x_1 and its approximation respectively. (c) and (d) Evolution of x_2 and its approximation respectively. (e) Evolution of the RBF's number.

References:

[1] C. Qixin, H. Yanwen, Z. Jingliang, An evolutionary artificial potential field algorithm for dynamic path planning of mobile robot,IEEE/ RSJ Int. conf. on Intelligent Robots and Systems, pp. 3331-3336, Beijing, oct. 2006

[2] D. Shi, D. S. Yeung, J. Gao, Sensitivity analysis applied to the construction of radial basis function networks, Neural networks vol. 18 pp. 951-957, 2005

[3] G. P. Liu, V. Kadirkamanathan and S. A. Billings, Variable neural networks for adaptive control of nonlinear systems, IEEE Trans. Trans. Syst. Man Cybern. B. Cybern., vol. 398, no. 1, pp. 34-43, Feb. 1999

[4] H. Al-Duwaish, S. Z. Rizi, Design of neuro-controller for multivariable nonlinear time-varying systems, WSEAS TRANSACTION on SYSTEMS and CONTROL, Vol. 5, pp. 711-720, Sep 2010

[5] H. Mekki, M. Chtourou, N. Derbel, "Variable structure neural networks for adaptive control of nonlinear systems using the stochastic approximation ", Simulation Modeling Practice and Theory 14 (2006) 1000-1009.

[6] J. Balicki, An adaptive quantum-basqed evolutionary algorithm for multiobjective optimization, WSEAS TRANSACTION on SYSTEMS and CONTROL, Vol. 4, pp.603-612, Dec. 2009

[7] J. Lian, Y. Lee, S. D. Sudhoff and S. H. Zak, "Variable Structure Neural Network Based Direct Adaptive Robust Control of Uncertain Systems", American Control Conference, Seattle, Washington, USA, June 2008.

[8] J. Lian, Y. Lee, S. D. Sudhoff, S. H. Zak, "Self-Organizing Radial Basis Function Netwoek for Real-Time Approximation of Continuous-Time Dynamical Systems, IEEE Transactions on neural network, Vol. 19, No. 3, March 2008.

[9] J. R. B. Jr and M. do C. Nicoletti, A multiclass version of a constructive neural network algorithm based on linear separability and convex hull, ICANN,; Part II, LNCS, pp. 723-733, 2008.

[10] K. Salahshoor, M. R. Jafari, "On-line identification of non-linear systems using adaptive RBF-based neural networks", International Journal of Information Science and Technology, Vol. 5, No. 2, 99-121, July/December 2007.

[11] L. Augusto D. C. Meleiro, F. J. V. Zuben, R. M. Filho, "Constructive learning neural network applied to identification and control of fuel-ethanol fermentation process", Engineering Applications of Artificial Intelligence 22 (2009) 201-215.

[12] L. Ma, K. Khorasani, New training strategies for constructive neural networks with application to regression problems, Neural networks Vol.17 pp. 589-609, 2004.

[13] M. Belkheiri, F. Boudjema, Function approximation based augmented backstepping control for an introduction machine, WSEAS TRANSACTION on SYSTEMS and CONTROL, Vol. 2, Sep 2007.

[14] M. D. C. Nicoletti and J. R. Bertini, an empirical evaluation of constructive neural network alghorithms n classification tasks, Int. J. Innovative Computing and Application, Vol. 1, No. 1, pp. 1-13, 2007

[15] M. d. C. Nicoletti, and al., constructive neural network algorithms for feedforward architectures suitable for classification tasks, Springer-Verlag Berlin Heidelberg, SCI 258, pp.1-23, 2009.

[16] M. Grochowski and W. Duch, Constructive neural network algorithms tat solve highly non-separable problems, Springer-Verlag Berlin Heidelberg, SCI 258, pp. 49-70, 2009.

[17] M. G. Park, J. H. Jeon and M. C. Lee, obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing, IEEE international symposium on industrial electronics, vol. 3, pp. 1530-1535, 2001.

[18] M. Kubalcik, V. Bobal, Self-tuning of continuous-time systems, WSEAS TRANSACTION on SYSTEMS and CONTROL, Vol. 5, pp. 802-813, Oct. 2010.

[19] M. R. Jafari, al., On-line identification of non-linear systems using an adaptive RBF-based neural network, Proceeding of WCECS, 58-3, October 2007, San Francisco, USA.

[20] N. B. Karayiannis and G. W. Mi, Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques, IEEE Trans. Neural Netw., vol. 8, no. 6, pp- 1492-1506, Nov, 1997.

[21] Neri. F. (2012), Learning and predicting financial series by combining evolutionary computation and agent simulation, Transactions on computational collective intelligence, Vol. 6, Springer, Heidelberg, Vol. 7, pp. 202-221

[22] P. Y. Ahang, T. S. Lü and L. B. Song, Soccer robot path planning based on the artificial potential field approach with simulated annealing, Robotica, vol. 22, pp. 563-566, 2004.

[23] P. Vadakkepat, T. H. Lee, L. Xin, Application of evolutionary artificial potential field in robot soccer system, IFSA World Congress and 20th NAFIPS int. Conf. vol. 5, pp. 2781-2785, Vancouver, BC, Canada juil. 2001

[24] S. Lee and R. M. Kil, A gaussian potential function network with hierachiclally self-organizing learning, Neural Netw., vol. 4, no. 2, pp. 207-224, 1996.

[25]S. S. Ge and Y. J. Cui, New potential functions for mobile robot path planning, IEEE Transactions on Robotics and automation, vol. 16, No. 5, pp. 615-620, October 2000.

[26] S. S. Ge and Y. J. Cui, New potential functions for mobile robot path planning, IEEE Trans. On Robot. And Auto., vo. 16, No. 5, Octob. 2000.

[27] S. S. Baboo, P. Subashini, M. Krishnaveni, Combining self-organizing maps and radial basis function networks for tamil handwritten character reconition, ICGST International Journal on Graphics, vision and image processing, Vol. 9, pp. 1-7, 2009.