FEEDFORWARD NEURAL NETWORKS COMBINED WITH A NUMERICAL INTEGRATOR STRUCTURE FOR DYNAMIC SYSTEMS MODELING

Atair Rios Neto
INPE Instituto Nacional de Pesquisas Espaciais
12227-010 São José dos Campos, SP, Brasil
e-mail: atairrn.@uol.com.br

Paulo Marcelo Tasinaffo
INPE Instituto Nacional de Pesquisas Espaciais
12227-010 São José dos Campos, SP, Brasil
e-mail: tasinaffo@dem.inpe.br

Abstract: *The use of neural networks in the control of dynamic systems always implies in the need of training the neural net in order to get an internal model of the system. One of the ways of representing the internal model of the dynamics of the system is to design the neural network to learn a system approximation in the form of a discrete model with delayed inputs in the form of a NARMA (Nonlinear Auto Regressive Moving Average) model. The neural net designed and trained in this way has the disadvantage of needing too many neurons in the input and hidden layers. In this work a new approach to represent the dynamics of the system using a feedforward neural network is preliminarily tested. In this approach, using the structure of an ordinary differential equation (ODE) numerical integrator it is possible to have the neural network designed to only learn the dynamic system derivative function. As a consequence, an unnecessary complexity in the design of the neural network is avoided and it has to only learn an algebraic static function. A simple but practical ODE dynamical model of an orbit transfer between Earth and Mars problem is considered for the tests. The results obtained reinforce the expectations that the new approach is advantageous in terms of simplicity and performance when compared to the approach where the neural network plays the role of a NARMA model.*

Keywords: *Dynamic Systems Neural Modeling, Neural Networks, Dynamic Systems Modeling.*

1. Introduction

The usual approach in neural control schemes employing feedforward neural networks has been that of using the neural network to model and play the role of a discrete nonlinear input-output NARMA (Nonlinear Auto Regressive Moving Average) type of model, in the approximation of the dynamic system (e.g.: Chen and Billings, 1992; Hunt et al, 1992; Liu et al, 1998). There are at least two difficulties with this approach. The first is the adjustment of the order of the input-output model, in terms of the number of delayed responses and of delayed control actions. The second is the situation of usually having too many inputs, and thus too many neural connections and related parameters, to deal with in the training of the neural network.

A possibility yet to be better explored is that offered by the methods of numerical integration of ordinary differential equations (ODE) (e.g., Stoer and Bulirsch, 1980). The computer propagation model provided by an ODE numerical integrator is a discrete forward model of the dynamic system, which by itself can be used as an internal working model in control schemes. These numerical integrators have characteristics which are quite relevant for a dynamical system model to be used in control, since they allow: (i) parallel processing, component by component of the dynamic system state; (ii) local accuracy to be adjusted by available methods of automatically varying the order or the step size of the integrator (e.g., Fehlberg, 1968; Prothero, 1980); and (iii) estimation of accumulated global prediction errors (e.g., Rios Neto and Kondapalli, 1990).

If in the structure of an ODE numerical integrator algorithm a feedforward neural network is used to approximate and to replace the derivative function of the ordinary differential equations dynamic system mathematical model, a discrete model is obtained which has quite advantageous characteristics (Rios Neto, 2001; Wang and Lin, 1998). With this approach, the difficulty with too many inputs in the training of the neural network is alleviated, since it is only necessary to learn an algebraic and static function, and the inputs are occurrences of the state and control variables in their envelope of variation.

In what follows, in Section 2, for the benefit of those not familiar with neural networks, the fundaments about feedforward multiplayer perceptron networks and the property they have of learning nonlinear mappings are introduced. In Section 3, the possibility of using an ODE integrator as a discrete forward model of the dynamic system is considered, together with the basic idea of taking a feedforward neural network to learn the derivative function of a dynamic system, and then use it in the structure of an ODE numerical integrator to get a discrete forward internal model. In Section 4, conditions and results of the preliminary tests are presented. In Section 5, conclusions are drawn

2. Fundaments: Feedforward Multilayer Perceptrons

Among the types of neural artificial networks used for modeling and identification of systems (Chen and Billings,1992) the most basic and frequently used is the Multilayer Perceptron made up of layers of basic artificial neurons connected forward, as illustrated in Fig.1, for the ith neuron of a kth hidden layer with n_k neurons:

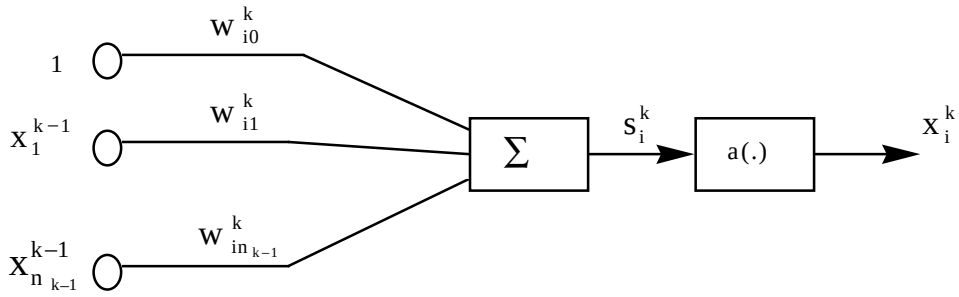

Figure 1: Perceptron Artificial Neuron

$$s_i^k = \sum_{j=1}^{n_{k-1}} w_{ij}^k x_j^{k-1} + w_{i0}^k \tag{1}$$

$$x_i^k = a(s_i^k), \quad k = 1,2,...,l-1 \tag{2}$$

with the activation function a(s) being typically taken as :

$$a(s) = 1/(1+\exp(-s)) \text{ or } a(s) = \tanh(s) \tag{3}$$

The inputs to the first hidden layer are $x_i^0 = x_i, i = 1,2,...,n$, the network input vector. For the neurons of the output layer $(k = l)$ it is sufficient to have and are frequently used zero threshold weights (w_{i0}^l) and identity activation functions:

$$\hat{y}_i \equiv x_i^l = \sum_{j=1}^{n_{l-1}} w_{ij}^l x_j^{l-1}, i = 1,2,...,m \tag{4}$$

Feedforward artificial neural networks can be trained to uniformly and with the desired accuracy represent a nonlinear and continuous mapping (see, e.g., Zurada, 1992):

$$f \in C : x \in D \subset R^n \rightarrow y \in R^m \tag{5}$$

The existing theory (Cybenko, 1988; Hornik et al, 1989) guarantees that for the case of the Mutilayer Perceptron it is enough to have a neural network built with just one hidden layer, as illustrated in Fig. 2.

A feedforward network training is usually done by supervised learning from mapping data sets:

$$\{(x(t), y(t)) : y(t) = f(x(t)), t = 1,2,...,L\} \tag{6}$$

adjusting (estimating) the weight parameters to approximately fit the artificial neural net correspondent computational model to this data of input-output patterns.

The processing by the trained artificial neural net of the input data $x(t)$, to produce outputs $\hat{y}(t)$, can be viewed and treated as a parameterized mapping:

$$\hat{y}(t) = \hat{f}(x(t), w) \tag{7}$$

where w is the vector of weight parameters. In the case of the perceptron neural net with one hidden layer and hyperbolic tangent activation function (Fig.2), Eq. (7) is expressed as:

$$\hat{y}_i(t) = \sum_{j=1}^{n_1} w_{ij}^2 (\tanh[\sum_{k=1}^{n} w_{jk}^1 x_k(t) + w_{j0}^1]) \quad (8)$$

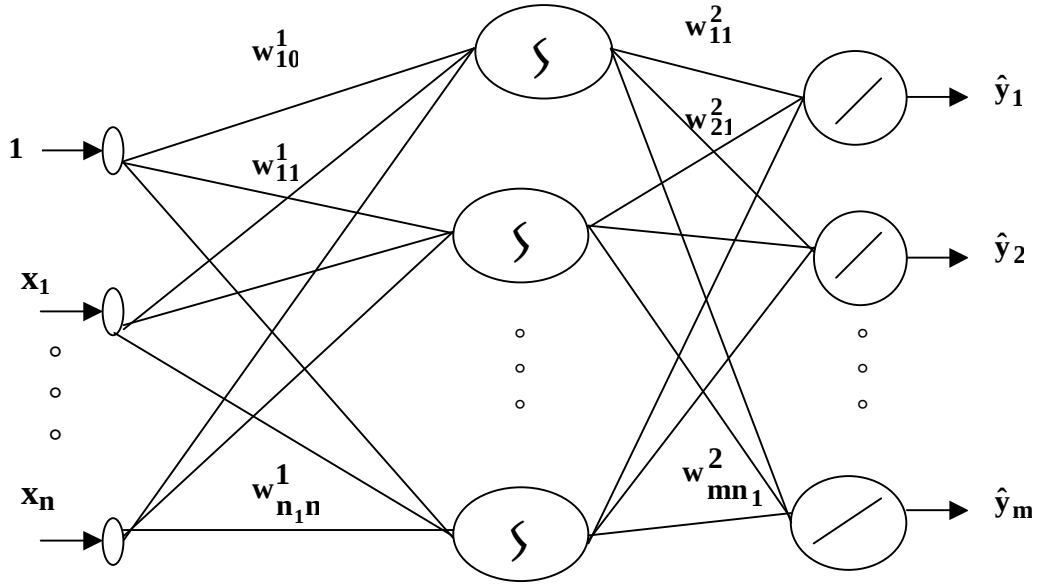

Figure 2: Multilayer Perceptron: One Hidden Layer

This capacity of feedforward artificial neural nets of representing nonlinear mappings can be used to approximately model dynamic systems as long as they are invariant in time (see, e.g., Chen and Billings, 1992). To do so, it is usually assumed the possibility of approximating the dynamic system by a NARMA type of discrete model, like:

$$x(t+\Delta t) = f(x(t), x(t-\Delta t), ..., x(t-n_x \Delta t); u(t), u(t-\Delta t), ..., u(t-n_u \Delta t)) \quad (9)$$

where $n_x, n_u, \Delta t$ are to be adjusted, depending upon the problem treated and desired accuracy. This possibility is considered to use the feedforward multilayer perceptron to play the role of a discrete model as in Eq. (9), adjusting the size of the neural network (number of layers and number of neurons per layer) to attain the desired accuracy, for a given choice of $n_x, n_u, \Delta t$.

At this point it is opportune to remember the similar situation that occurs when ODE numerical integrators are used and dynamic systems are treated by discrete approximations, like in Eq.(9).

3. Numerical Integrator Discrete Forward Model and Neural Numerical Integrator

Consider a dynamic system, with a mathematical model given by a set of ordinary differential equations:

$$\dot{x} = f(x,u) \quad (10)$$

where $x \in R^n$ is the state vector; $u \in R^m$ is the control vector; $f(x,u)$ is the derivative function, coming from physical laws governing the dynamic system.

Consider now an ordinary differential equation (ODE) numerical integrator (e.g., Stoer and Bulirsch, 1980) to get a discrete approximation of the system of Eq.(10):

$$x(t+\Delta t) \cong f_n(x(t), x(t-\Delta t), ..., x(t-n_o \Delta t); u(t), ..., u(t-n_o \Delta t); \Delta t) \quad (11)$$

where $f_n(x(t), x(t-\Delta t), ..., x(t-n_o \Delta t); u(t), ..., u(t-n_o \Delta t); \Delta t)$ is the function that results from using evaluations of the derivative function of Eq. (10) in the numerical integrator algorithm; n_o is related to the order of the approximation; if its value is greater than zero, one has the situation where a finite difference type of integrator is used (for example, an Adams-Bashforth method); if it is zero, a single step type of integrator is used (for example, a Runge-Kutta method); and Δt , the step size, is assumed sufficiently small to assure *u(t)* constant along the discretization interval.

The numerical integrator in Eq.(11) can be used recursively as an approximate discrete predictive model of the dynamic system of Eq.(10) in internal model control schemes. The error in each step can be controlled by varying step size and or the order of numerical integrator; and the resulting numerical algorithm can be processed in parallel for each component of the state of the dynamic system.

In the dynamic system of Eq. (10), the derivative function in the ODE mathematical model is an algebraic function that can be approximated by a feedforward neural network:

$$\dot{x} = f(x,u) \cong \hat{f}(x,u,\hat{w}) \tag{12}$$

where $\hat{f}(x,u,\hat{w})$ is to represent the neural network trained; and $\hat{w}$ the neural network learned weights.

Consider now this neural approximation of the derivative function in the structure of an ordinary differential equation (ODE) numerical integrator (e.g., Stoer and Bulirsch, 1980) to get a discrete approximation of the system of Eq.(10):

$$x(t+\Delta t) \cong \hat{f}_n(x(t),x(t-\Delta t),...,x(t-n_o\Delta t);u(t),...,u(t-n_o\Delta t);\Delta t;\hat{w}) \tag{13}$$

The neural numerical integrator in Eq.(13) is an approximation of the ODE numerical integrator of Eq. (11) and thus an approximate discrete predictive model of the dynamic system of Eq.(10) which can be used as an internal model in control schemes.

4. Preliminary Tests Results

The considered approach was preliminarily tested in the problem of modeling the dynamics involved in a practical problem of orbit transfer between Earth and Mars.

This is a problem where the state variables are the rocket mass *m*, the orbit radius *r*, the radial speed *w* and the transversal speed *v*, and where the control variable is the thrust steering angle θ, measured from local horizontal. The ODE (e.g., Sage, 1968) of this dynamic system are:

$$\dot{m} = -0.0749 \tag{14.a}$$

$$\dot{r} = w \tag{14.b}$$

$$\dot{w} = \frac{v^2}{r} - \frac{\mu}{r^2} + \frac{T \cdot sin\theta}{m} \tag{14.c}$$

$$\dot{v} = \frac{-w \cdot v}{r} + \frac{T \cdot cos\theta}{m} \tag{14.d}$$

where the variables have been normalized with: $\mu = 1.0$, the gravitational constant; $T=0.1405$, the thrust; with $t_o=0$ and $t_f=5$, initial and final times, where each unit of time is equal to *58.2* days. The numerically simulated solutions used a random control law, where in each discrete interval the control θ could be changed between $-\pi$ and $+\pi$.

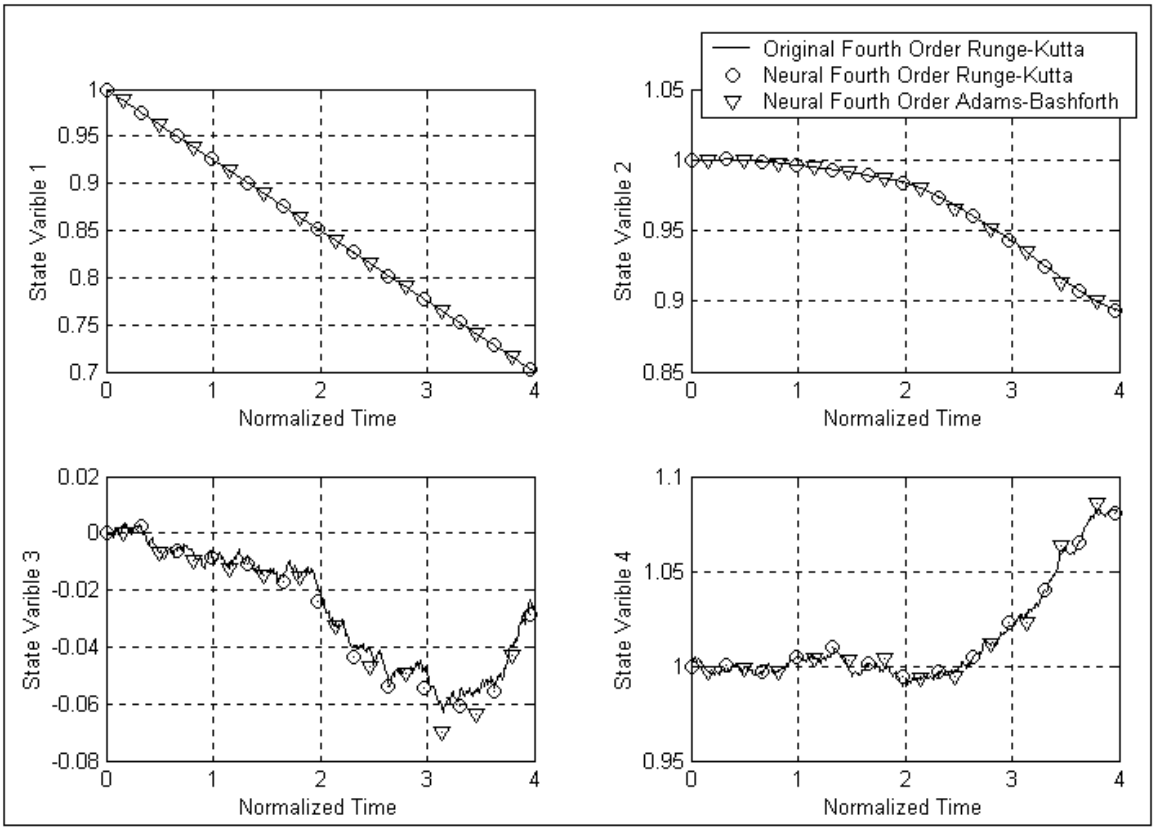

Figure 4 – Neural Integrator Modeling of Dynamics of Orbit Transfer Between Earth and Mars.

Figure 4 shows numerically simulated trajectories of the dynamics of the Earth Mars orbit transfer problem. The continuous curve represents the numerical solution with the 4th Runge-Kutta integrator using the original derivative function as given by Eqs (14.a) to (14.d); this is considered to be the validation solution, used to evaluate the neural modeled solutions. The circles represent the solution obtained with the neural 4th Runge-Kutta and the triangles the solution with the neural 4th Adams-Bashforth. In all cases a step size of 0.01 units of normalized time was used, and at each 50 propagation steps in time the trajectories were reinitialized, with values of state variables assumed not contaminated by errors. This is a reasonable assumption since in practice the state of a system to be controlled is frequently corrected in a feedback loop, by processing navigation measured observations with a state estimator providing state estimation errors that can be considered negligible for the purpose of the control involved.

To approximate the vector derivative function, a multiplayer perceptron feedforward neural network with the hyperbolic tangent as activation functions, in the hidden layer, and neurons with identity activation, in the output layer, was used. The empirical adjustment of the number of neurons in the hidden layer, using a parallel processing Kalman filtering algorithm in the training, led to the adoption of a solution with 41 neurons in this layer, for which a mean square error (MSE) of 3.3565e-05 in the test data set was reached.

In order to get an indication of existence of advantageous characteristics, tests were also done with the correspondent NARMA model. To guarantee conditions for comparison the same Kalman training algorithm and the same feedforward multilayer perceptron, but trained as a NARMA model to directly model the dynamics of the same problem, were used. Just one delay time step was used for state and control in the NARMA neural model, with the same delay time of 0.01 units of normalized time, and with the same sequence of controls used in the test with the neural integrator; and having as well the trajectories reinitialized at each 50 propagation steps in time, with values of state variables which were assumed not contaminated by errors.

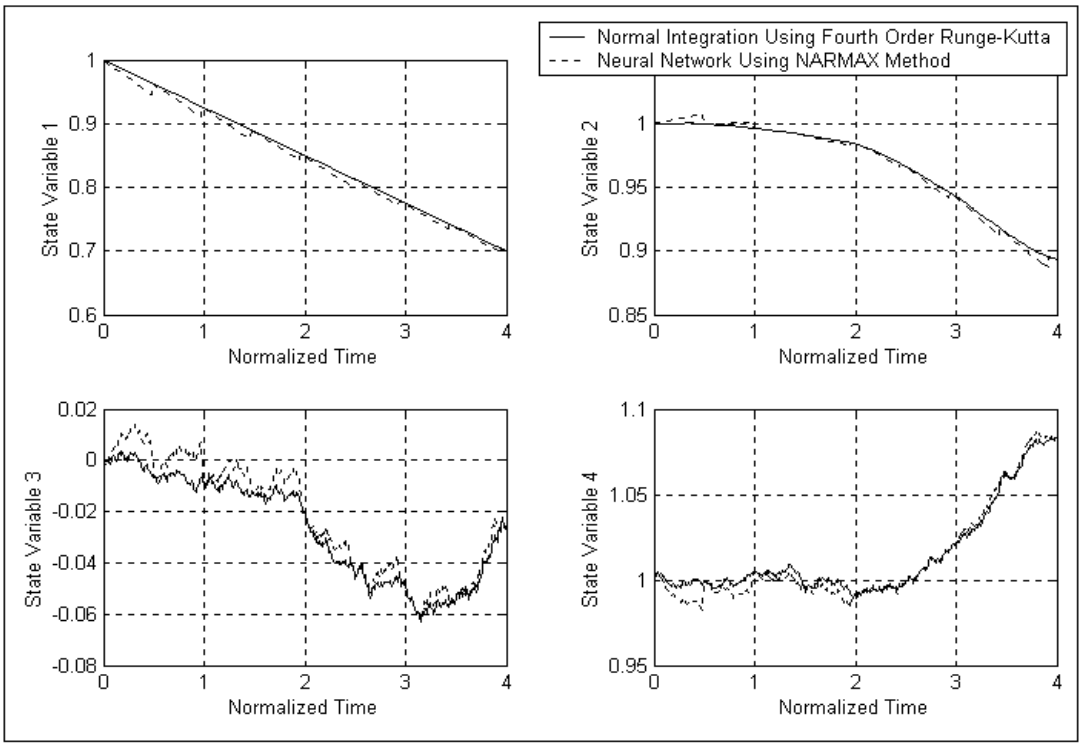

Figure 5 –NARMA Neural Modeling of Dynamics of Orbit Transfer Between Earth and Mars.

In this testing of the NARMA model, with the results depicted in Fig. 5, a MSE error of 2.1983e-07 in the test data set was reached. Though this error was of two orders of magnitude better than that with the neural integrator alternative, the results obtained with the NARMA neural model were worse than those obtained with the ODE neural integrators, as can be seen by visual inspection and comparison with results in Fig. 4. In the numerical simulations, the worst global state vector deviation errors with respect to the validation state vector trajectory, observed at the end of propagation arcs of 50 steps, were of magnitudes (absolute values) 7.7871e-03 and 2.8704e-02, for the neural integrator and NARMA neural model, respectively. These results indicate that for the NARMA neural modeling to be competitive with the ODE neural integrator, in terms of accuracy, it would need either to be of higher order, with a greater number of delays in its inputs, if a neural net with one hidden layer with the same number of neurons is used, or to use a neural net with more neurons, possibly distributed in more than one layer. Thus, the results indication is that, as expected, the combination with the ODE integrator allows the use of a simpler neural network, in terms of number neural connections.

5. Conclusions

A new approach to get dynamic systems discrete forward models, to be used in control schemes where an internal model is needed, was preliminarily tested.

In this approach, the structure of ODE numerical integrators are exploited to get neural discrete forward models where the neural network has only to learn and approximate the algebraic and static derivative function in the dynamic system ODE.

The preliminary numerical tests indications reinforce the following expected characteristics:

(i) it is a simpler task to train a feedforward neural network to learn the algebraic , static function of the dynamic system ODE derivatives (where the inputs are samples of state and control variables), than to train it to learn a NARMA type of discrete model (where the inputs are samples of delayed responses and controls);

(ii) for the same level of accuracy, the neural network in the neural ODE integrator usually results to be simpler, in terms of the necessary number of layers and number of neurons, since it does not have to learn the dynamic law , but only the derivative function;

(iii) the existing knowledge about step size and order adjustment in numerical integration can be used to control expected prediction accuracy;

(iv) even in the situation where an ODE mathematical model is not available, as long as dynamic system input output pairs are available to be used as training information, the structure of the numerical integrator with a feedforward network in place of the derivative function can be trained to get a discrete internal model in control schemes;

(v) finally, it is important to consider that the use of a neural network in the dynamic system discrete model will naturally allow the implementation of adaptive control schemes, due to the learning capacity of the neural network.

6. Acknowledgment.

This work has been possible due to the support given by Instituto Nacional de Pesquisas Espaciais-INPE, to the authors; and the doctoral scholarship given by CNPq, to the second author.

7. References

Chen, S., Billings, S. A., 1992, "Neural Networks for Nonlinear Dynamic System Modeling and Identification", Int. J. Control, Vol. 56, No. 2, pp. 319-346.

Cybenko, G., 1988, "Continuous Valued Networks With Two Hidden Layers Are Sufficient", Technical Report, Department of Computer Science, Tufts University.

Felhlberg, E., 1968, "Classical Fifth, Sixth, Seventh, and Height-Order Runge Kutta Formulas With Step Size Control", Technical Report R-287, NASA, Washington, DC.

Hornik, K., Stinchcombe, M., White, H., 1989, "Multilayer Feedforward Networks are Universal Approximators", Neural Networks, Vol. 2, pp. 359-366.

Hunt, K. J., Sbarbaro, D., Zbikowski, R., Gawthrop, P. J., 1992, "Neural Networks for Control Systems – A Survey". Automatica, Vol. 28, $N_{o.}$ 6, pp. 1083-1112.

Liu, G. P., Kadirkamanathan, V. and Billings, S. A., 1998, "Predictive Control for Non-linear Systems Using Neural Networks", Int. Journal of Control, vol. 71, no. 6, pp. 1119-1132.

Narendra, K. S., Parthasarathy, K., 1990 "Identification and Control of Dynamical Systems Using Neural Networks", IEEE Transactions on Neural Networks, Vol. 1, pp. 4-27.

Prothero, A., 1980, "Estimating the Accuracy of Numerical Solution to Ordinary Differential Equations", in: Gladwell, I. And Sayers, D.K., Eds., Computational Techniques for Ordinary Differential Equations, Academic Press, London.

Rios Neto, A., Rama Rao, K., 1990, "A Stochastic Approach to Global Error Estimation in ODE Multistep Numerical Integration", Journal of Computational and Applied Mathematics, Vol. 30, pp. 257-281.

Rios Neto, A, 1997, "Stochastic Optimal Linear Parameter Estimation and Neural Nets Training in Systems Modeling", RBCM – J. of the Braz. Soc. Mechanical Sciences Vol. XIX, $N_{o.}$ 2, pp. 138-146.

Rios Neto, A.,, 2001, **"**Dynamic Systems Numerical Integrators in Neural Control Schemes", Proceedings of V Brazilian Congress of Neural Networks, Rio de Janeiro, RJ, Brasil.

Sage A. P., 1968, "Optimum Systems Control", Prentice-Hall, Inc., Englewood Cliffs, N. J.

Stoer, J. and Bulirsch, R., 1980, "Introduction to Numerical Analisys", Springer Verlag, N.Y.

Wang, Y.J., Lin, C.T., 1998, "Runge-Kutta Neural Network for Identification of Dynamical Systems in High Accuracy", IEEE Transactions On Neural Networks, Vol. 9, No. 2, pp. 294-307.

Zurada, J. M., 1992, "Introduction to Artificial Neural Systems", West Pub. Co.