INPE-11303-PRE/6740

MODELING WITH ODE NEURAL NUMERICAL INTEGRATORS APPLIED TO THE DYNAMICS OF AN ORBIT TRANSFER PROBLEM

Atair Rios Neto
Paulo Marcelo Tasinaffo

ADVANCES IN SPACE DYNAMICS 4: CELESTIAL MECHANICS AND ASTRONAUTICS,
H. K. Kuga, Editor, 243-252 (2004).
Instituto Nacional de Pesquisas Espaciais – INPE, São José dos Campos, SP, Brazil.
ISBN 85-17-00012-9

INPE
São José dos Campos
2004

ADVANCES IN SPACE DYNAMICS 4: CELESTIAL MECHANICS AND ASTRONAUTICS,
H. K. Kuga, Editor, 243-252 (2004).
Instituto Nacional de Pesquisas Espaciais – INPE, São José dos Campos, SP, Brazil.
ISBN 85-17-00012-9

MODELING WITH ODE NEURAL NUMERICAL INTEGRATORS APPLIED TO THE DYNAMICS OF AN ORBIT TRANSFER PROBLEM

Atair Rios Neto
e-mail: atairrn@uol.com.br
Paulo Marcelo Tasinaffo
e-mail: tasinaffo@dem.inpe.nr
INPE – Instituto Nacional de Pesquisas Espaciais

ABSTRACT

The usual approach to nonlinear dynamic systems neural modeling has been that of training a feedforward neural network to represent a discrete non-linear input-output NARMAX type of model. In this paper, the alternative approach of combining feedforward neural networks with the structure of ordinary differential equations (ODE) numerical integrator algorithms is considered. The neural network is used to approximate the algebraic derivative function in the dynamic system ODE model. A Kalman filtering training algorithm with parallel processing is employed in the supervised estimation of neural network weight parameters. The main objective is to have an approach where dynamic systems neural modeling is a simpler task since the feedforward neural network has to only learn the algebraic and static function of the dynamic system ODE derivatives. To illustrate the effectiveness of the adopted approach, results of test in a simple but practical dynamic system, representative of the dynamics of a problem of orbit transfer between Earth and Mars, are presented.

Keywords: Dynamic Systems Neural Models, Numerical Integrators, Feedforward Neural Nets, Kalman Filter Supervised Training, Orbit Transfer.

I INTRODUCTION

Neural networks have been used widely for identification of dynamical systems (e.g.: Carrara 1997; Silva, 2001; Hunt et al, 1992; Chen and Billings, 1992). It is possible to train a feedforward neural network to represent a discrete nonlinear input-output NARMAX type of discrete model of a dynamic system, reducing the neural modeling problem to an equivalent one of learning how to represent a non linear function (Narendra and Parthasarathy, 1990). Neural networks of the multiplayer perceptron type, with just a single internal hidden layer, are enough for this task of representing nearly any function encountered in applications (Cybenko, 1988; Hornik et al, 1989).

In recent works another approach was adopted, using ordinary differential equations (ODE) numerical integrators combined with neural networks to model dynamic systems (Rios Neto, 2001; Wang and Lin, 1998). It was shown the possibility of getting a discrete forward model of the dynamic system using the structure of numerical integrators where the feedforward neural network approximates the derivative function in the ordinary differential equations mathematical model of the dynamic system. The approach is intended to simplify and to provide more flexibility to the problem of dynamic systems neural modeling, since the neural network has only to learn an algebraic function with the dynamic behavior being implied by the numerical integrator structure and since it is now possible to try to control accuracy by changing the discretization step size and numerical integrator order (e.g., Rios Neto and Rama Rao, 1990).

With the purpose of evaluating and selecting solutions to the problem of getting internal models for use in dynamic systems control schemes, specially in the case of transfer orbit control, the latter approach is applied to the practical and representative case of modeling the nonlinear dynamics of a spacecraft in an orbit transfer between Earth and Mars.

In what follows, in section II, for the benefit of those who are not familiar with neural computing, the fundaments about feedforward neural networks and about dynamic systems neural modeling with NARMAX type of discrete models are given. To complete the information about the numerical tools used in the tested approach, basic information about the Kalman filter supervised training method used in the training of the neural network is also given In section III, the use of feedforward nets to model the derivative function of a dynamic system in the structure of a numerical integrator algorithm is explained. In section IV, the simulations and results of the application to the problem of modeling the dynamics of an orbit transfer between Earth and Mars are presented. In the last section some conclusions about the new approach are drawn.

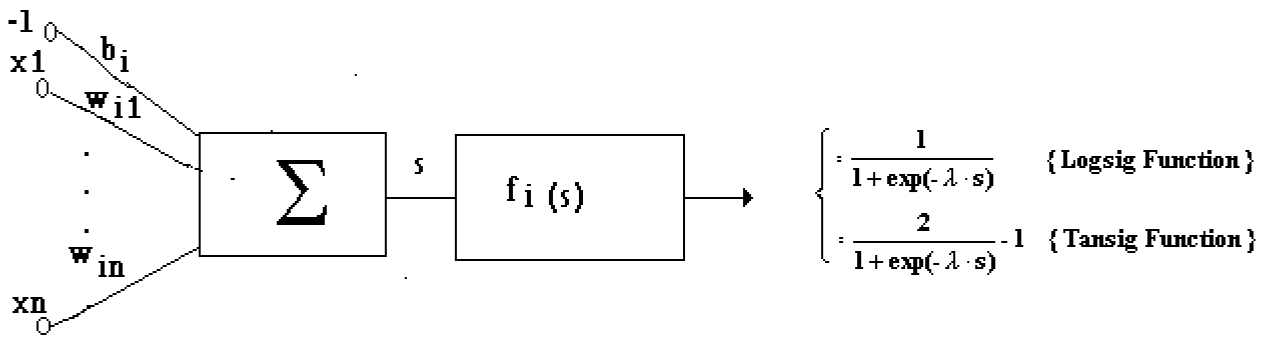
II FUNDAMENTALS: FEEDFORWARD NEURAL NETWORKS, SUPERVISED TRAINING, AND DYNAMIC SYSTEMS MODELLING

Feedforward Neural Networks:

It is not easy to find out a unique definition about what is an artificial neural network (Zurada (1992)) In very general terms, artificial neural networks are computation models, made up of artificial computationally modeled neurons, that result from exploring analogies with biological neural networks of life organisms.
The feedforward multiplayer perceptron is a neural network where Perceptron type of artificial neurons (e.g.,Braga et al, 2000) (see Figure 1 for a generic ith neuron) are organized in layers connected in a feedforward architecture. The Perceptron neuron is a classifier, where inputs (x_j's) are initially classified with respect to a hyperplane characterized by the weights (w_{ij}'s) and a bias (b_i), in series with a second non linear classification done by an activation function ($f_i(.)$), which limits the classification to be inside asymptotic normalized limits of (0,1), in the case of the Logsig activation or (-1,+1), in the case of the Tansig activation.

The *Feedforward* neural network (Figure 2) is made up of columns of Perceptrons (f's) having as inputs the inputs to the neural network, in the case of the first layer, or the outputs of previous layer, in the case of a generic hidden layer, and as outputs the input to next layer, in the case of an hidden layer, or the output of the network, in the case of the output layer. The output vector $\mathbf{x}^k$ of a layer **k** has thus as its **ith** component:

$$x_i^k = f^k\left(\sum_{j=1}^{n_{k-1}} w_{ij}^k \cdot x_j^{k-1} - b_i^k\right) \quad \text{p/} \quad i = 1, 2, ..., n_k \tag{1}$$



-1
x1
xn
b_i
w_{i1}
w_{in}
$\Sum$
s
$f_i(s)$
$= \frac{1}{1+\exp(-\lambda \cdot s)}$
{Logsig Function}
$= \frac{2}{1+\exp(-\lambda \cdot s)} - 1$
{Tansig Function}

Figure 1 – The Perceptron Neuron.

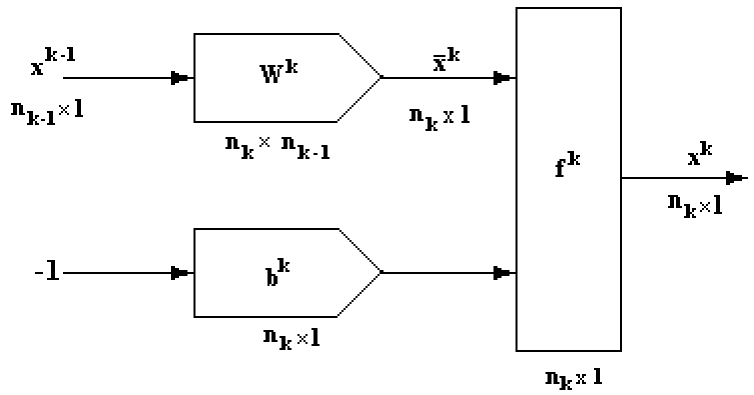


Figure 2 – Input/Output of a General Hidden Layer k.

Supervised Training:

The training of a feedforward network to approximate a given function f(x) is done by supervised learning from data sets of input output patterns:

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

The neural network is seen as a parameterized mapping where the parameters or weights are adjusted to fit the input-output patterns:

$$\hat{y}(t) = \hat{f}(x(t), w) \tag{3}$$

where the **w** is the weight vector of parameters to be adjusted. The usual approach of supervised training is to minimize, with respect to the vector of weights **w**, the functional:

$$\mathbf{J} = \frac{1}{2} \cdot [\, (\mathbf{w} - \overline{\mathbf{w}})^T \overline{\mathbf{P}}^{-1} (\mathbf{w} - \overline{\mathbf{w}}) + \sum_{t=1}^{L} \left(\mathbf{y(t)} - \hat{\mathbf{f}}(\mathbf{x(t)},\mathbf{w})\right)^T \mathbf{R}^{-1}(\mathbf{t}) \left(\mathbf{y(t)} - \hat{\mathbf{f}}(\mathbf{x(t)},\mathbf{w})\right)] \tag{4}$$

An iterative algorithm for minimizing the functional of Equation (4) can be obtained if a stochastic meaning is given to the weighting matrices, a linear approximation is taken, and the following stochastic linear estimation problem is considered in a ith iteration (Rios Neto, 1997):

$$\overline{\mathbf{w}} = \mathbf{w(i)} + \overline{\mathbf{e}} \tag{5.a}$$
$$\mathbf{z(t,i)} = \mathbf{H(t,i)} \cdot \mathbf{w(i)} + \mathbf{v(t)} \tag{5.b}$$
$$\mathbf{E}[\overline{\mathbf{e}}] = \mathbf{0}, \qquad \mathbf{E}[\mathbf{ee}^{\mathbf{T}}] = \overline{\mathbf{P}} \tag{5.c}$$
$$\mathbf{E[v(t)]} = \mathbf{0}, \quad \mathbf{E}[\mathbf{v(t)}\,\mathbf{v}^{\mathbf{T}}\mathbf{(t)}] = \mathbf{R(t)} \tag{5.d}$$
$$\mathbf{z(t,i)} \equiv \alpha\mathbf{(i)} \cdot [\mathbf{y(t)} - \overline{\mathbf{y}}\mathbf{(t,i)}] + \hat{\mathbf{f}}_{\mathbf{w}}(\mathbf{x(t)},\overline{\mathbf{w}}\mathbf{(i)}) \cdot \overline{\mathbf{w}}\mathbf{(i)} \tag{5.e}$$
$$\mathbf{H(t,i)} \equiv \hat{\mathbf{f}}_{\mathbf{w}}(\mathbf{x(t)},\overline{\mathbf{w}}\mathbf{(i)}) \tag{5.f}$$

Where,
i = 1, 2, ..., I; t= 1,2,…, L;
$\overline{\mathbf{w}}\mathbf{(i)}$ is the a priori estimate of **w** coming from the previous iteration, starting with $\overline{\mathbf{w}}\mathbf{(1)} = \overline{\mathbf{w}}$;
$\overline{\mathbf{y}}\mathbf{(t,i)} = \hat{\mathbf{f}}(\mathbf{x(t)},\overline{\mathbf{w}}\mathbf{(i)})$ is the output of the feedforward network*;*
$\hat{\mathbf{f}}_{\mathbf{w}}(\mathbf{x(t)},\overline{\mathbf{w}}\mathbf{(i)})$ is the matrix of first partial derivatives with respect to **w**;
$\mathbf{0} < \alpha\mathbf{(i)} < \mathbf{1}$ is a parameter to be adjusted in order to guarantee the hypothesis of linear perturbation.

A Kalman filtering, batch solution can then be obtained, in a ith iteration (see, e.g., Jazwinski, 1970):

$$\hat{\mathbf{w}}\mathbf{(i)} = \overline{\mathbf{w}} + \mathbf{k(i)}[\mathbf{z(i)} - \mathbf{H(i)} \cdot \overline{\mathbf{w}}] \tag{6.a}$$
$$\mathbf{k(i)} = \overline{\mathbf{P}} \cdot \mathbf{H}^{\mathbf{T}}\mathbf{(i)}[\mathbf{H(i)} \cdot \overline{\mathbf{P}} \cdot \mathbf{H}^{\mathbf{T}}\mathbf{(i)} + \mathbf{R}]^{-1} \tag{6.b}$$
$$\overline{\mathbf{w}}\mathbf{(i+1)} = \hat{\mathbf{w}}\mathbf{(i)}, \qquad \alpha\,\mathbf{(i)} \leftarrow \alpha\,\mathbf{(i+1)} \tag{6.c}$$

The *off line* solution after **i=1, 2, ..., I** iterations is given by:

$$\hat{\mathbf{w}} = \hat{\mathbf{w}}\mathbf{(I)}, \;\; \mathbf{P(I)} = [\mathbf{I} - \mathbf{K(I)} \cdot \mathbf{H(I)}]\overline{\mathbf{P}} \tag{7}$$

It can be shown that the Kalman filtering turns out to be a Newton Modified type of method, which is more effective and efficient than the usual gradient type Backpropation method. In this paper a simplified version of this algorithm with a parallel processing structure was used (Rios Neto, 1997).

Dynamic System Neural Modeling:

In a control problem it is important to identify a discrete model of the physical system, to be used as an internal model, in the control structure (e.g., Norgaard et al, 2000). The neural network can approximate this model if it is put in parallel with the real system and the error between this system and the network outputs is used as a performance to train the net. Thus, if it is assumed that the system can be modeled and governed by the following non linear difference equation NARMAX type of model (e.g., Norgaard et al, 2000) :

$$\mathbf{y^p(t+1) = f[y^p(t), ..., y^p(t-n+1); u(t), ..., u(t-m+1)]} \tag{8}$$

where the output of system $\mathbf{y^p}$ in time **t+1** depends of **n** past output values and of **m** past controls values, the self-evident approach to do the neural modeling of the system is to choose the input/output feedforward neural network structure as the same as that of the discrete model, as in Equation (8). Denoting the output of the network as $\mathbf{y^m}$ it then results:

$$\mathbf{y^m(t+1) = \hat{f}[y^p(t), ..., y^p(t-n+1); u(t), ..., u(t-m+1), w]} \tag{9}$$

where the function $\mathbf{\hat{f}}$ represents the non-linear mapping network, parameterized with respect to its connection weights **w**. The network has during training, as inputs, the past values of real system outputs, as well as past values of inputs to the real system. After training, the network gives a good representation of the real plant (that is, $\mathbf{y^m \cong y^p}$) and can be used as a discrete model of the plant.

III DYNAMIC SYSTEMS AND NEURAL NUMERICAL INTEGRATORS

Consider a dynamic system and suppose that its ODE mathematical model is known and that an artificial neural network was trained to represent the derivative function in this model, resulting that:

$$\mathbf{\dot{x} = f(x,u) \cong \hat{f}(x,u,\hat{w})} \tag{10}$$

Where,

$\mathbf{x \in R^n}$ is the vector of state variables;

$\mathbf{u \in R^n}$ is the vector control variables;

$\mathbf{\hat{w}}$ is the weight vector of the trained neural network;

$\mathbf{f(x,u)}$ is the derivative function;

$\mathbf{\hat{f}(x,u,\hat{w})}$ represents the trained neural network approximating the derivative function.

Consider now an ODE numerical integrator to get a discrete approximation of dynamic system of Equations (10):

$$\mathbf{x(t+\Delta t) \cong f_n(x(t), x(t-\Delta t), ..., x(t-n_o\Delta t); u(t), ..., u(t-n_o\Delta t); \Delta t)} \tag{11}$$

Where, (see e.g., Rao, 1984) if $\mathbf{n_0}$ is equal to zero one has a simple step integrator, for example, the Euler and Runge-Kutta integrators; if $\mathbf{n_0}$ is greater than zero one has a multi step, for instance, a finite difference type of integrator, as the Adams-Bashforth.

The numerical integrator of Equation (11) can be used as the discrete predictive model of a dynamic system, which is given by an ODE, Equation (10). If the neural network which represents the ODE derivative function is substituted for the derivative function in the numerical integrator structure, a neural numerical integrator results, which can be used as a dynamic system internal model in control schemes (Rios Neto, 2001). As compared to a NARMAX type of neural network modeling, the neural numerical integrator modeling, besides simplifying the neural network training, allows approximation error control in any step, through the possibility of changing the step size or the order of the numerical integrator. If efficient training algorithms are used, there is also the possibility for on line updating of the ODE neural integrator model to get adaptive control solutions, as illustrated in Figure 3.

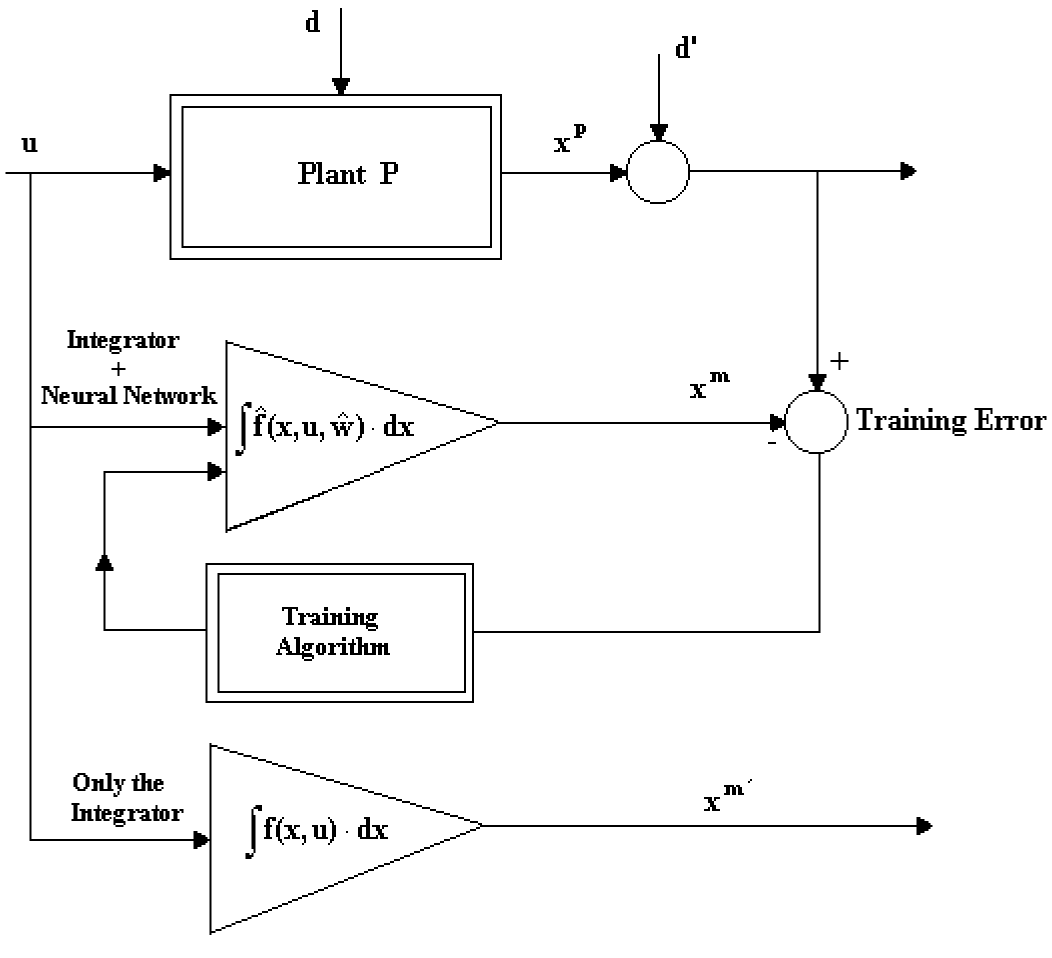


Figure 3 -Adaptive Scheme: On Line Training of Neural ODE Integrator

IV SIMULATIONS

To illustrate the effectiveness of the considered approach it was applied to the modeling of 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:

$$\dot{\mathbf{m}} = \mathbf{-0.0749} \tag{12.a}$$

$$\dot{\mathbf{r}} = \mathbf{w} \tag{12.b}$$

$$\dot{\mathbf{w}} = \frac{\mathbf{v}^2}{\mathbf{r}} - \frac{\mu}{\mathbf{r}^2} + \frac{\mathbf{T} \cdot \sin\theta}{\mathbf{m}} \tag{12.c}$$

$$\dot{\mathbf{v}} = \frac{\mathbf{- w \cdot v}}{\mathbf{r}} + \frac{\mathbf{T} \cdot \cos\theta}{\mathbf{m}} \tag{12.d}$$

Where the variables have been normalized with: μ **=1.0**, the gravitational constant; **T=0.1405**, the thrust; with $\mathbf{t_o=0}$ and $\mathbf{t_f=5}$, 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 $+\pi$.

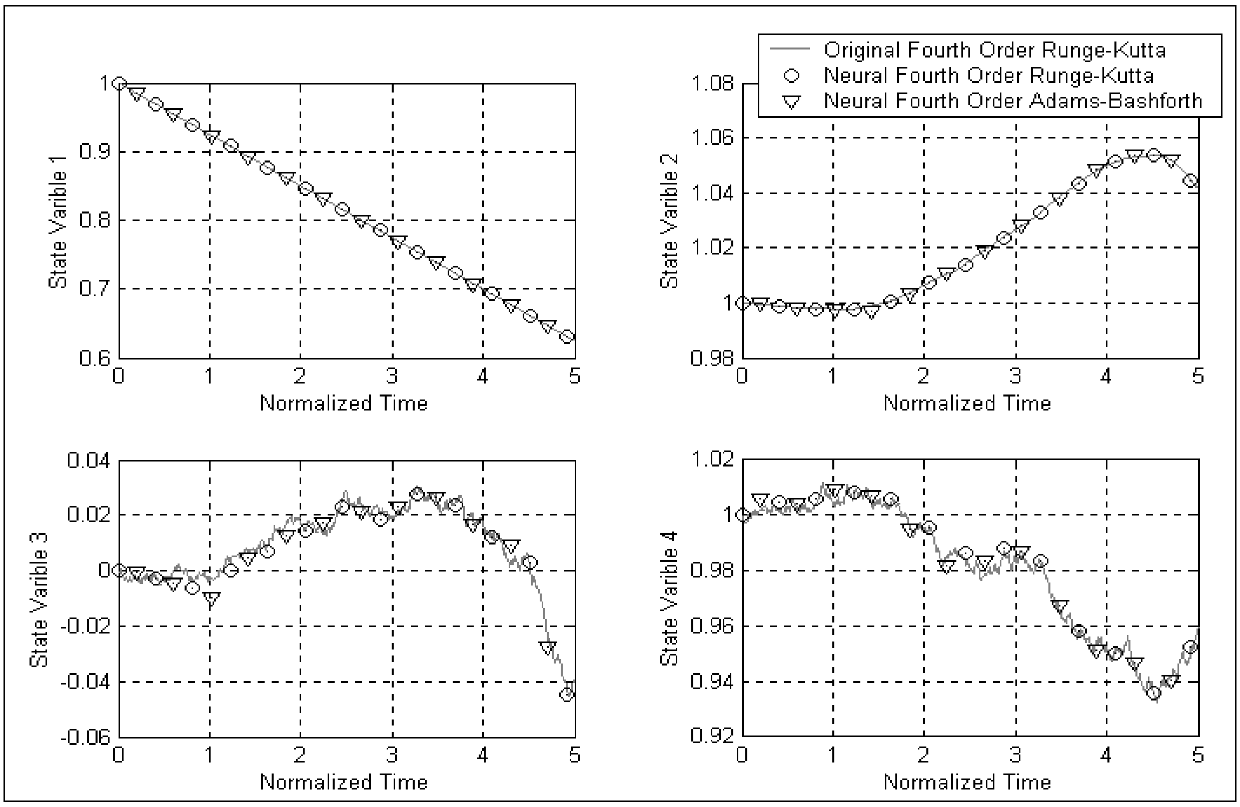


Figure 4 – Neural Integrator Modeling of Dynamics of Orbit Transfer Between Earth and Mars.

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 equations (12.a) to (12.b); this is considered to be the validation solution, used to evaluate the neural modeled solutions. The circles represents the solution obtained with the neural 4th Runge-Kutta and the triangle 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 41 neurons with tansig activations, in the hidden layer, and neurons with identity activation, in the output layer, was used. This feedforward neural net was trained with a parallel processing Kalman filtering algorithm, reaching a mean square error (MSE) of 3.3565e-05.

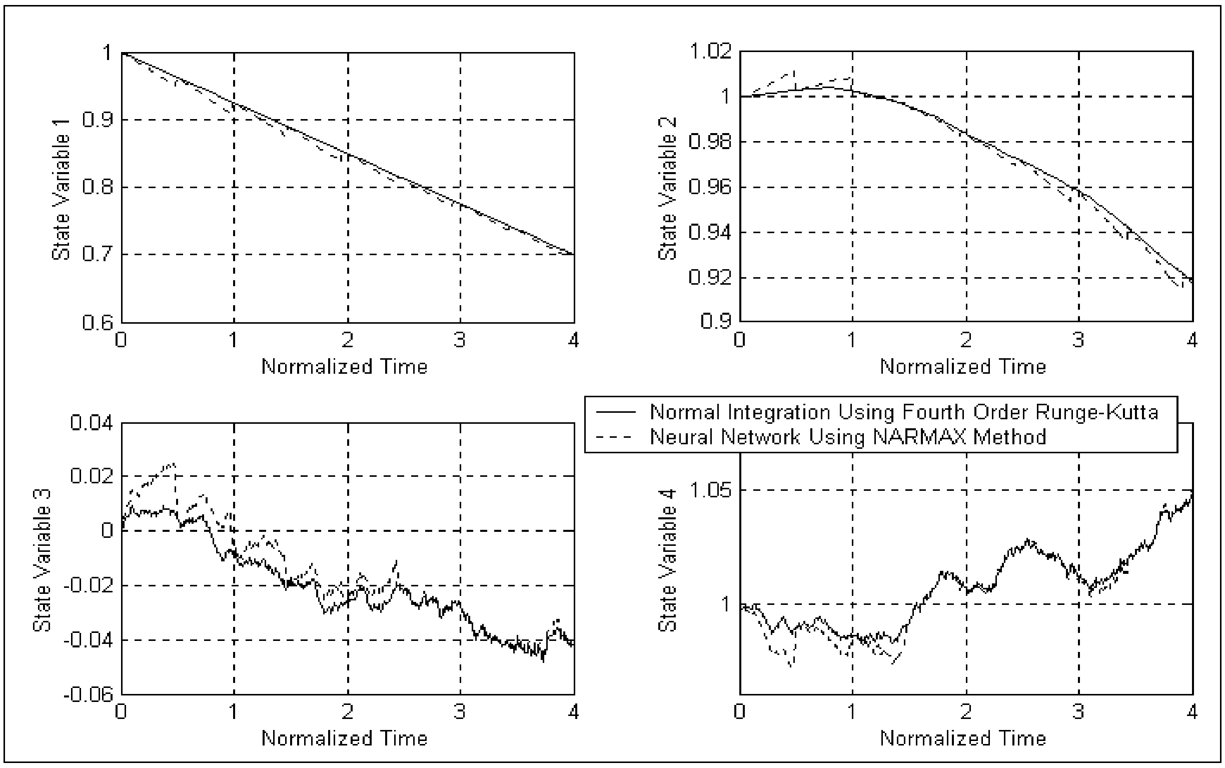


Figure 5 –NARMAX Neural Modeling of Dynamics of Orbit Transfer Between Earth and Mars.

In Figure 5, the results obtained, using the same Kalman training algorithm and the same feedforward multilayer perceptron, but trained as a NARMAX model to directly model the same problem, are presented. In order to guarantee conditions for comparison, just one delay time step was used for state and control in the NARMAX neural model, with the same delay time of 0.01 units of normalized time, 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. In this alternative, a MSE training error of 2.11e-07 was reached. Though this error was of two orders of magnitude better than that with the neural integrator alternative, the results obtained with the NARMAX neural model were worse than those obtained with the ODE neural integrators, as depicted in Figures 4 and 5. 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) 6.357e-03 and 2.963e-02, for the neural integrator and NARMAX neural model, respectively. These results indicate that the NARMAX neural modeling would need to be of higher order, with a greater number of delays in its inputs, in order to be competitive with the ODE neural integrator. The results indicate, as expected, that the combination with the ODE integrator allows the use of a simpler neural network, in terms of number neural connections.

V CONCLUSIONS

The numerical experiment done with the orbit transfer problem reinforces the advantageous characteristics expected with the use of ODE neural integrator for modeling dynamic systems. It was observed that:

a) It is an easier task to train a neural network to learn an algebraic function or a derivative function, than train it to learn a discrete model of the full dynamic system;

b) The neural network results to be simpler in what concerns the number of neuron connections;

c) The flexibility to change the step size and the order of the integrator can be used to control the accuracy of the discrete model in approximating the dynamic system.

The quality of the results obtained indicates that the modeling approach can be considered for on board implementations of orbit control schemes where an internal working model of controlled system is necessary. It is worth noticing that the parallel processing characteristic is completely preserved if finite difference type of ODE integrators are used. The use of efficient parallel processing training algorithms, as the Kalman filtering, opens the possibility for on line updating of the ODE neural integrator model to get adaptive control solutions.

VI REFERENCES

[01] – Braga, A. P., Ludermir, T. B., Carvalho, A. C. P. L. F. *Redes Neurais Artificiais Teoria e Aplicações*. LTC, 2000.

[02] – Carrara, V. *Redes Neurais Aplicadas ao Controle de Atitude de Satélites com Geometria Variável*. (Doctoral Thesis in Space Sciences) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, INPE 1997. INPE-6384-TDI/603.

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

[04]- Cybenko, G. *Continuous Valued Networks With Two Hidden Layers Are Sufficient*. Technical Report, Department of Computer Science, Tufts University.

[05] – Hornik, K., Stinchcombe, M., White, H. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, Vol. 2, pp. 359-366, 1989.

[06] – Hunt, K. J., Sbarbaro, D., Zbikowski, R., Gawthrop, P. J. Neural Networks for Control Systems – A Survey. *Automatica*, Vol. 28, $N_{o.}$ 6, pp. 1083-1112, 1992.
[07] – Jazwinski, A. H. *Stochastic Processes and Filtering Theory*. Academic Press, New York and London, 1970.
[08] – Narendra, K. S., Parthasarathy, K. Identification and Control of Dynamical Systems Using Neural Networks. *IEEE Transactions on Neural Networks*, Vol. 1, march 1990, pp. 4-27.
[09] – Norgaard, M., Ravn, O., Poulsen, N. K., Hansen, L. K. *Neural Networks for Modeling and Control of Dynamic Systems*. Springer, London, 2000.
[10] – Rao, K. R. *A Review on Numerical Methods for Initial Value Problems*. INPE-3011-RPI/088, Feb., 1984.
[11] – Rios Neto, A., Rama Rao, K. A Stochastic Approach to Global Error Estimation in ODE Multistep Numerical Integration. *Journal of Computational and Applied Mathematics*, Vol. 30, pp. 257-281, 1990.
[12] – Rios Neto, A. 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, 1997.
[13] – Rios Neto, A. Dynamic Systems Numerical Integrators in Neural Control Schemes. *V Congresso Brasileiro de Redes Neurais*, 2 – 5 abril, 2001, Rio de Janeiro, RJ, Brasil.
[14] – Sage A. P. *Optimum Systems Control*. Prentice-Hall, Inc., Englewood Cliffs, N. J., 1968.
[15] – Silva, J. A. *Controle Preditivo Utilizando Redes Neurais Artificiais Aplicado a Veículos Aeroespaciais*. (Doctoral Thesis in Space Science) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, INPE 2001. INPE-8480-TDI/778.
[16] – Wang, Y.-J., Lin, C.-T. Runge-Kutta Neural Network for Identification of Dynamical Systems in High Accuracy. *IEEE Transactions On Neural Networks*, Vol. 9, No. 2, pp. 294-307, March 1998.
[17] – Zurada, J. M. *Introduction to Artificial Neural System*. West Pub. Co., 1992.