International Journal of Innovative Computing, Information and Control
Volume **15**, Number **1**, February **2019**

ICIC International ©2019 ISSN 1349-4198
pp. 383–406

AN INTRODUCTION TO UNIVERSAL NUMERICAL INTEGRATORS

Paulo Marcelo Tasinaffo, Gildárcio Sousa Gonçalves
Adilson Marques Da Cunha and Luiz Alberto Vieira Dias

Computer Science Division
Technological Institute of Aeronautics
Praça Marechal Eduardo Gomes, 50 Vila das Acácias, 12228-900 São José dos Campos/SP, Brazil
tasinaffo@ita.br; { gildarciosousa; cunha.adilsonmarques2 }@gmail.com; l_vdias@yahoo.com.br

Received May 2018; revised September 2018

Abstract. *In this article, the concept of a universal numerical integrator is mathematically defined. Based on this definition it is possible to derive three useful methodologies for representation and modeling of real-world dynamic systems. The three methodologies in question are the NARMAX (Nonlinear AutoRegressive Moving Average with eXogenous inputs), the instantaneous derivatives methodology and the mean derivatives methodology. All of these methodologies work to model real-world dynamic systems through supervised learning using input/output training patterns. The concept of universal numerical integrators combines the mathematical definitions of a classical numerical integrator (e.g., Runge-Kutta Integrators, Adams-Bashforth Integrators, and Predictor-Corrector Methods) and a universal approximator of functions (e.g., artificial neural networks or Mamdani-type fuzzy inference systems). The practical applications of these methodologies are vast and involve, for example, river basin outflow forecasting, sunspot prediction and control theory. In control theory these three methodologies can be used in adaptive control, predictive control or in IMC (Internal Model Control) structures.*

Keywords: Numerical integrators, Universal approximator of functions, NARMAX methodology, Instantaneous and mean derivatives methodologies, Euler neural networks, Runge-Kutta neural networks, Adams-Bashforth neural networks

1. Introduction.

The development of the mathematical theories used in the analytical approach of problems involving the solution of dynamic systems began with the development of differential and integral calculus by Newton and Leibniz that were elaborated during the last quarter of century XVII. In this context the first differential equations arise. Great advances have occurred since then, and as early as the eighteenth century the differential equations became a new branch of mathematics and became an independent discipline.

Great advances were made during the eighteenth century, such as the method of resolution by separable variables, linear equations, Bernoulli's equations, and Riccati's equations. At the end of the eighteenth century the study of the differential equations already elaborated by Euler, Lagrange and Laplace became so important that they already included problems involving variational calculus, celestial mechanics, fluid dynamics, complex variables, power series, Bessel functions, among others (see [1]). However, it can be said that it was only during the 19th century that the great movement of foundation and expansion took place in the studies of differential equations. At this time, the qualitative theory of the differential equations appeared and, later, the study of the stability and instability of dynamic systems without previously knowing the analytical solution of such systems (e.g., [1,6,7]). Introduced by Poincaré in 1881, qualitative theory (e.g.,

DOI: 10.24507/ijicic.15.01.383

[1,7,76,77]) is a fundamental landmark in the evolution of differential equations. The general stability problem was studied independently by Poincaré and Liapounov and are considered the founders of the qualitative theory of the differential equations. In 1908, mathematicians Runge and Kutta developed the high-order numerical integrators and the discrete solution for nonlinear and non-autonomous dynamic systems (see [3-5,8-11]), but still in the deterministic case. Although a general and continuous analytical solution for systems of ordinary differential equations for the nonlinear and non-autonomous case is not yet known, in the 20th and 21st centuries much theoretical grounding in the study of such equations was obtained with great success using the discrete approach (e.g., [12,14-17,19-23]). However, this whole theory is applicable only to theoretical models and does not work well when the application of interest in modeling of real world dynamic systems must be developed from experimental data or from laboratory experiments.

In this way, the most appropriate theory to deal with experimental data is the estimation or least squares theory. Its origin is attributed to two great mathematicians: Gauss in the year 1795 and Legendre in the year 1808. These two mathematicians developed this theory for the following purpose: to predict the orbits of comets and thus to determine the date of their returns to Earth through data obtained through astronomical observations. The estimation theory, at this time, was developed from the concepts of differential and integral calculus and left some theoretical aspects open. Fortunately, Kalman in [2] idealizes the estimation theory from the concept of random variables and, therefore, the stochastic treatment for formulation and resolution of dynamic systems, making it more complete from the mathematical point of view in relation to the Gaussian approach and Legendre.

The Russian mathematician A. N. Kolmogorov (1903-1987) solves in 1957 the problem proposed by Hilbert in 1900, that is, which is based on the demonstration of the possibility of representing any function of n-dimensional space through a linear combination of limited and one-dimensional functions. From this it becomes possible to elaborate the existence of a universal approximator of functions (see [24-30]). Hornik et al. [25] adapt Kolmogorov's work to the case of artificial neural networks, and from that point on it becomes known the mathematical proof of the existence of a universal approximator of functions for the mapping of an m-dimensional space for the n-dimensional space through neural networks with feedforward architecture, more precisely, MLP (Multilayer Perceptron) networks. In this way, artificial neural networks are regarded as universal estimators for experimental data. Because they are universal estimators, an attempt has been made to represent real world dynamical systems from it and not from the theory of differential equations, since the latter does not deal directly with experimental data. Thus, the representation and modeling of dynamic systems from MLP networks with the NARMAX methodology and later application in control theory were developed, among others, by Hunt and Sbarbaro in [72].

However, dealing with neural networks in practice is very costly because it requires many empirical adjustments and reasonable consumption of computational processing of hardware and software, making it in some cases even impracticable. In order to minimize the practical problems of using neural networks, in the modeling of dynamic systems, other techniques were still also proposed using neural networks. In this way, it seems to be convincing that it is happening, an interesting event, to adapt the estimation theory to the theory of the differential equations. This has the great purpose of making neural networks more user friendly from such techniques and at the same time continuing to use experimental data. However, this approach has the drawback of leaving the mathematical part of the development of neural networks a little more complex.

Wang and Lin in [13] propose the representation of dynamic systems from a Radial Basis Function (RBF) network or a Multilayer Perceptron network inserted into a high-order numerical integration structure, such as the use of single step integrators of the type Runge-Kutta 4-5. These authors introduce in the literature the term Runge-Kutta neural networks or instantaneous derivatives methodology. It can be said that this is a very relevant work, because it combines the theoretical development of the neural networks together with some theoretical aspects of the ordinary differential equations. Melo and Tasinaffo in [18] introduce the neural networks with Multilayer Perceptron architecture inserted into high-order numerical integrators with multiple steps named Adams-Bashforth.

Rios Neto et al. in [56-58,69,70] apply the higher-order neural integration structures of the Adams-Bashforth and Runge-Kutta types in dynamic system modeling (using the instantaneous derivatives methodology) and subsequent application in predictive control theory. They still compare this performance with the NARMAX methodology.

Tasinaffo et al. in [64,71] published two articles on the topic of neural networks and low-order numerical integration structures in the representation of non-linear dynamic systems and subsequent application in control. Among the methodologies proposed by them is the mean derivatives methodology. It is worth mentioning that the mathematical consistency of this methodology is evaluated through some previously existing theorems in the qualitative theory of the differential equations. In [61] the mean derivatives methodology with artificial neural networks with Multilayer Perceptron architecture is used. A detailed mathematical formalism of mean derivatives methodology can be found in [64-66]. In [65] the mean derivatives methodology is used with Mamdani-type fuzzy inference systems. In Tasinaffo and Rios Neto [71] is developed a very brief theoretical and practical application of the mean derivatives methodology in predictive control structures.

The mathematical concept of universal numerical integrators encompasses, in a general way, the three methodologies for modeling nonlinear dynamical systems in the real world. The three methodologies in question are the NARMAX methodology, the instantaneous derivatives methodology (which involves high order numerical integrators) and the mean derivatives methodology (which involves low order integrators). Universal numerical integrators can also be perfectly used in control theory.

The content of this article is divided into 7 sections. Section 2 defines and classifies the main types of classical numerical integrators. Section 3 mathematically defines a universal approximator of functions. Section 4 mathematically defines a universal numerical integrator. This definition combines the concepts of classical numerical integrators with universal approximator of functions. Section 5 describes in detail the three methodologies for the representation of nonlinear dynamic systems, from the concept of universal numerical integrators. Section 6 briefly describes the use of universal numerical integrators in control theory. Section 7 briefly describes the main conclusions of this article.

2. **Brief Description about the Key Types of Numerical Integrators.**

In this section we suggest some computational algorithms used to numerically solve ordinary differential equations or system of differential equations. Let $f = f(y, t)$ be a real function of two real variables defined for $a \leq t \leq b$ where a and b are finite, and for every real value of y. The equation

$$\dot{y} = f(y, t) \tag{1}$$

is called an ordinary differential equation of the first order; it symbolizes the following problem: find a function $y = y(t)$, continuous and differentiable for $t \in [a, b]$, such that

$$\dot{y}(t) = f(y(t), t) \tag{2}$$

for all $t \in [a,b]$. The function $y = y(t)$ with this property is called the solution of the differential equation (1). In general, a differential equation can have many solutions. To avoid this type of problem, you must specify the value of the function $y(t)$ at a given point, for example, $y(t) = \eta$ in $t = a$. In many application problems it may be found not with a simple differential equation, but with a system of m simultaneous first order equations with dependent variables ${}^1y$, ${}^2y$, ..., ${}^my$. If each of these variables satisfies a given initial condition for the same value a of t, then we have an initial value problem for a first-order system, which can be written:

$$\begin{aligned} {}^1\dot{y} &= {}^1f\left(t, {}^1y, {}^2y, \ldots, {}^my\right), \quad {}^1y(a) = {}^1\eta \\ {}^2\dot{y} &= {}^2f\left(t, {}^1y, {}^2y, \ldots, {}^my\right), \quad {}^2y(a) = {}^2\eta \\ {}^m\dot{y} &= {}^mf\left(t, {}^1y, {}^2y, \ldots, {}^my\right), \quad {}^my(a) = {}^m\eta \end{aligned} \tag{3}$$

Introducing the vector notation:

$$\vec{y} = \left[{}^1y, {}^2y, \ldots, {}^my\right]^T \tag{4a}$$

$$\vec{f} = \left[{}^1f, {}^2f, \ldots, {}^mf\right]^T = \vec{f}(t, \vec{y}) \tag{4b}$$

$$\vec{\eta} = \left[{}^1n, {}^2n, \ldots, {}^mn\right]^T \tag{4c}$$

then, we can write the initial value problem (3) in the form:

$$\dot{\vec{y}} = \vec{f}(t, \vec{y}), \quad \vec{y}(a) = \vec{\eta} \tag{5}$$

It can be shown that a second order ordinary differential equation can be transformed into a system of two first order differential equations, a third order equation in a system of two second order differential equations, and so on. In this way, any high order system can be transformed into a first order system and fall into the situation given by Equations (3) or (5). Given the existence of differential equations (3) for systems, then it can be said that they will not always have an analytical solution, especially if they are non-linear. In these cases only numerical solutions can be obtained. Thus, there are basically two classes of numerical methods for solving a system of first order differential equations (e.g., [3-8,11]):

a) *simple step method*, which computes the value of y in only one step $\Delta t = h$ in the independent variable t by using only the information coming from the functions made between t and $t + \Delta t$ (local point of view);

b) *multi-step method*, which computes a step by using the information over several preceding steps (global view).

The general approximation for a numerical solution of a differential equation involves a sequence of discrete points, equally spaced or not, in the form t_0, t_1, t_2, ... At each point t_n, the solution $y(t_n)$ is approximated by a number y_n which is computed from earlier values. That is, at each step, an approximation is made using the values of the previous approximations. In other words, an error at a given point is propagated with integration progress. Thus, $h_n = t_{n+1} - t_n$ must be adjusted and the error in $y(t_n)$ must be estimated.

Simple Step Methods. The simple step method is an algorithm which describes a numerical technique for calculating the approximation for solution in y_{n+1}, in terms of the value in an earlier step (y_n). The simple step methods are mainly divided into two categories: Taylor series methods and Runge-Kutta methods.

The Runge-Kutta Methods. It is much easier to have a polynomial that agrees with y and $\dot{y}$ on t_n and with $\dot{y}$ (or f) on several other arguments close to t_n than a polynomial that agrees with t_n and its k derivatives in t_n (Taylor series). The Runge-Kutta methods are based on this principle. The second-order equations of the Runge-Kutta method are given below, but without demonstration:

$$k_1 = h \cdot f(y_n, t_n) \tag{6a}$$

$$k_2 = h \cdot f(t_n + h/2, y_n + k_1/2) \tag{6b}$$

$$y_{n+1} \cong y_n + k_2 \tag{6c}$$

Thus, Equations (6a), (6b) and (6c) form a scheme to obtain y_{n+1}, where the values of t_n and y_n are known, for an accuracy $O(h^2)$. This solution also involves knowing $f(y,t)$ in two distinct points given by (y_n, t_n) and $(t_n + h/2, y_n + k_1/2)$. The equation described is the second-order Runge-Kutta process. The general Runge-Kutta method is defined as (see [8]):

$$y_{n+1} = y_n + h \cdot \Phi(t_n, y_n, h) \tag{7a}$$

where,

$$\Phi(t_n, y_n, h) = \sum_{r=1}^{R} C_r \cdot k_r \tag{7b}$$

$$k_1 = f(y_n, t_n) \tag{7c}$$

$$k_r = f\left(t_n + h \cdot a_r, y_n + h \cdot \sum_{s=1}^{r-1} b_{rs} \cdot k_s\right) \tag{7d}$$

$$a_r = \sum_{s=1}^{r-1} b_{rs} \quad r = 2, 3, \ldots, R \tag{7e}$$

The parameters C_r, a_r and b_{rs} are determined by comparing the terms of the successive powers of h between (7a) and the Taylor algorithm. With this, there are many different sets of Runge-Kutta equations for the same order proposed by several authors. For example, there are at least two distinct third-order formulas for Runge-Kutta integrators and they are Heun formula (e.g., [5]) and Kutta formula (e.g., [8]). Finally, the four-order Runge-Kutta equations (e.g., [3,5]) are listed below, which is one of the most used numerical algorithms for solving systems of ordinary differential equations due to their high precision and elegant simplicity:

$$y_{n+1} = y_n + \frac{h}{6} \cdot (k_1 + 2 \cdot k_2 + 2 \cdot k_3 + k_4) \tag{8a}$$

$$k_1 = f(y_n, t_n) \tag{8b}$$

$$k_2 = f\left(t_n + \frac{1}{2} \cdot h, y_n + \frac{1}{2} \cdot h \cdot k_1\right) \tag{8c}$$

$$k_3 = f\left(t_n + \frac{1}{2} \cdot h, y_n + \frac{1}{2} \cdot h \cdot k_2\right) \tag{8d}$$

$$k_4 = f\left(t_n + h, y_n + h \cdot k_3\right) \tag{8e}$$

There are many integration schemes that rely on values at more than one prior point to estimate the next point on the solution curve. Among these methods are those of Adams-Bashforth, Adams-Moulton, Simpson rule and Predictor-Corrector Method.

Multiple-Step Method. There are basically three ways of deriving these numerical expressions (e.g., [5]): through a Taylor series expansion, by numerical integration and through interpolations.

In this section will be considered only the integrators obtained by numerical integration, because this is the most used methodology. Suppose that for the equally spaced values $t_1, t_2, \ldots, t_n$ the numerical solutions $y_1, y_2, \ldots, y_n$ were obtained by some simple step method and now it is desired to obtain the solution in y_{n+1} using one or more previous values of $y(t)$. The way to solve this problem through the numerical integration is to represent Equation (1) by:

$$y(t_{n+p}) = y(t_n) + \int_{t_n}^{t_{n+p}} f(y(t), t) \cdot dt \tag{9}$$

where $f(y(t), t)$ is the derivative equation of the expression (1).

Next, without entering into the merits of the demonstrations, some very interesting expressions are stated in relation to numerical methods of multiple steps to solve first order dynamic systems. The main discrete expressions, ranging the order from one to four, corresponding to the Adams-Moulton algorithms (see [6]) that determine the discrete values in $y(t_{n+1})$ are, respectively, given by:

$$y(t_{n+1}) = y(t_n) + h \cdot f_n \tag{10a}$$

$$y(t_{n+1}) = y(t_n) + \frac{h}{2} \cdot [f_{n+1} + f_n] \tag{10b}$$

$$y(t_{n+1}) = y(t_n) + \frac{h}{12} \cdot [5 \cdot f_{n+1} + 8 \cdot f_n - f_{n-1}] \tag{10c}$$

$$y(t_{n+1}) = y(t_n) + \frac{h}{24} \cdot [9 \cdot f_{n+1} + 19 \cdot f_n - 5 \cdot f_{n-1} + f_{n-2}] \tag{10d}$$

Also in [6] the formulas of the Adams-Bashforth algorithms, also ranging the orders from one to four, can be found as listed as follows:

$$y(t_{n+1}) = y(t_n) + h \cdot f_n \tag{11a}$$

$$y(t_{n+1}) = y(t_n) + \frac{h}{2} \cdot [3 \cdot f_n - f_{n-1}] \tag{11b}$$

$$y(t_{n+1}) = y(t_n) + \frac{h}{12} \cdot [21 \cdot f_n - 16 \cdot f_{n-1} + 5 \cdot f_{n-2}] \tag{11c}$$

$$y(t_{n+1}) = y(t_n) + \frac{h}{24} \cdot [55 \cdot f_n - 59 \cdot f_{n-1} + 37 \cdot f_{n-2} - 9 \cdot f_{n-3}] \tag{11d}$$

Although equation $\dot{y} = f(y)$ does not explicitly contain the inputs of control u, as an independent variable, this equation can still be directly applied to feedback control systems. Specifically, if the dynamic plant is given by,

$$\dot{y} = f(y, u, t) \tag{12}$$

It is possible to select the control law as,

$$u = g(y, t) \tag{13}$$

Thus, the final dynamic system will be given by:

$$\dot{y} = f[y, g(y), t] \tag{14}$$

Equation (14) can of course be written in form $\dot{y} = h(y, t)$ falling back into the original situation given by Equation (1).

3. **Universal Approximator of Functions.**

For a proper mathematical study of the function approximation theory some definitions are a priori needed (see [30]).

Definition 3.1. *A function $f : D \to R^n$ is uniformly continuous on $D \subseteq R^n$ point if for each $\varepsilon > 0$ there is a $\delta(\varepsilon)$ so that for each $x, y \in D$ satisfying $|x - y| < \delta(\varepsilon)$, so $|f(x) - f(y)| < \varepsilon$. The function $f : D \to R$ is continuous on D if it is continuous for each point in D.*

Definition 3.2. *A real number b is a superior quota of a set $X \neq \phi$ of real numbers if and only if $x \leq b$ for every $x \in X$. Besides that, if any number smaller than b is superior quota of X, so one may say that b is a supreme of $(sup)X$.*

One can write the functions approximator as $F(x, w)$, where $w \in R^p$ is the parameters vector used to define the mapping of the approximator. Suppose that $\Omega^p \subset R^p$ denotes the subset of all values that the approximator parameters can obtain. Thus, it is assumed that

$$G = \{F(x, w) : w \in \Omega^p, p \geq 0\} \tag{15}$$

is the class of functions, in the form of $F(x, w)$, $w \subset \Omega^p$, for any $p \geq 0$. In this case, when one refers to the functions of G class, one omits how great p is. That said, a *uniform approximator* is defined as follows.

Definition 3.3. *A function $f : D \to R$ may be uniformly approximated on $D \subseteq R^n$ by functions of class G if for each $\varepsilon > 0$, there exists some $F \in G$ such that* $\sup_{x \in D} |F(x) - f(x)| < \varepsilon$.

Regarding Definition 3.3, note that the supreme operation on the function $F(x) - f(x)$ establishes that even the maximum difference in the greatness of these two functions, which inevitably occurs to some x, should still be less than ε. Thus, it gives now the definition of a universal approximator.

Definition 3.4. *A mathematical structure defining a class of functions G_1 is considered a universal approximator for the class functions G_2, if each $f \in G_2$ can be uniformly approximated by G_1.*

Thus, according to Jang et al. [28], the establishment of a broad class of universal approximators is mere existence theorems and not constructive methods. The starting theorem to establish the existence of universal functions approximators is the Stone-Weierstrass theorem. This theorem provides a useful way to determine whether certain approximators are really universal for a class of continuous functions and defined on a compact set. The Stone-Weierstrass theorem is stated below (see [26,28,30]).

Theorem 3.1. (T1) *(Stone-Weierstrass). A continuous function $f : D \to R$ can be uniformly approximated on $D \subseteq R^n$ by the class functions G if,*

(1) the constant function $g(x) = 1$, $x \in D$ belongs to G,

(2) if g_1, g_2 belong to G, so $a \cdot g_1 + b \cdot g_2$ will belong to G for every $a, b \in R$,

(3) if g_1, g_2 belongs to G, so $g_1 \cdot g_2$ will belong to G, and

(4) if $x_1 \neq x_2$ are two distinct points in D, then there will be a function in $g \in G$ so that $g(x_1) \neq g(x_2)$.

The Stone-Weierstrass theorem, which is derived from the real classical analysis, can be used to demonstrate that some neural network architectures have a universal approximation capacity (see [26]). The Stone-Weierstrass theorem states conditions warranting that neural networks with MLP (Multilayer Perceptron) architecture, or RBF (Radial-Basis Functions), or Mamdani-type Fuzzy Inference System (e.g., [28,30]) can approach any continuous functions in the sense given by Definitions 3.3 and 3.4. So before listing some important theorems taken from [30] concerning the theme of universal approximators, some important functions used in neural networks theory are set up.

A sigmoid function is every *continuous* function in the form of "s". An example of such a function can be the hyperbolic tangent function (or tansig function) with horizontal asymptotes in -1 and $+1$ given by the equation $\varphi(net) = \frac{2}{1+\exp(-\lambda \cdot net)} - 1$. Another example of sigmoid function can be the logistic function (or logsig function) with horizontal asymptotes in 0 and -1 and given by the equation $\varphi(net) = \frac{1}{1+\exp(-\lambda \cdot net)}$. Besides these, there are other functions with "s" shape, but they are discontinuous. Two of them are defined below.

Signal Function (Signum Function): a discontinuous function given by the equation $\varphi(net) = sgn(net) = \begin{cases} 1 & \text{if } net > 0 \\ -1 & \text{if } net < 0 \end{cases}$ or $\varphi(net) = \begin{cases} 1 & \text{if } net > 0 \\ 0 & \text{if } net = 0 \\ -1 & \text{if } net < 0 \end{cases}$. Note that this equation is the limit of $\lambda \to \infty$ in the hyperbolic tangent function.

Heaviside Function or Threshold Function: a discontinuous function given by equation $\varphi(net) = \begin{cases} 1 & \text{if } net \geq 0 \\ 0 & \text{if } net < 0 \end{cases}$. This equation was originally used in McCulloch and Pitts neuron in 1943 (see [77]). Note that this equation is the limit of $\lambda \to \infty$ in the Logistic function.

Note that the signal function and threshold are not sigmoid functions, since they are not continuous. Given the definition of these functions, one may list the following universal functions approximators (see [30]) that can be demonstrated from Theorem 3.1.

Theorem 3.2. (T2). *MLP networks of two layers, an inner layer of neurons defined by the sigmoid functions or threshold function and a linear output are universal approximators for $f \in G_{cb}(n, D)$, $D = [a, b]$.*

Theorem 3.3. (T3). *RBF networks defined by radial basis class functions $G_{fbr} = \left\{ g(x) = \sum_{i=1}^{p} a_i \exp(-\gamma_i |x - c_i|^2) \right\}$ with $a_i, \gamma_i \in R$ and $c_i \in R^n$ for $i = 1, 2, \ldots, p$ are universal approximators for $f \in G_{cb}(n, D)$.*

Theorem 3.4. (T4) *Fuzzy systems with triangular or Gaussian pertinence functions in the input and a defuzzification that uses the average of the centers are universal approximators for $f \in G_{cb}(n, D)$, $D = [a, b]$.*

Of course, there are other universal approximators of functions besides these, such as the Support Vector Machine (SVM) neural networks (see [41-46]) and neural networks with Deep Learning architecture (e.g., [47-54]). In the model presented in this section it is clear that any problem that can be solved with a neural network with MLP architecture could also be solved with an RBF network or a Mamdani-type fuzzy inference system or even a paraconsistent inference system.

4. Mathematical Definition of a Universal Numerical Integrator.

The mathematical definition for universal numerical integrators is a very general definition and intrinsically encompasses the three main empirical modeling methodologies of real-world dynamical systems, namely: 1) the NARMAX methodology (Nonlinear AutoRegressive Moving Average with eXogenous inputs), 2) the mean derivatives methodology and 3) the instantaneous derivatives methodology. It is important to point out that the methodology of the instantaneous derivatives is also sub-divided into two classes: the single-step and the multiple-step instantaneous derivatives methodology. Here, real-world dynamic modeling means supervised learning systems that learn the dynamic behavior of the world's physical phenomena from a set of in-and-out data training patterns of the plant in question. In this way, a universal numerical integrator empirically learns the dynamics of systems governed, in general, by nonlinear and autonomous ordinary differential equations. As learning is empirical, it is not necessary to know the theoretical mathematical model of the real world plant considered. In addition, a universal numerical integrator is defined from the combination of the concepts of a conventional numerical integrator coupled to any type of universal approximator of functions.

In the NARMAX methodology the universal approximator of functions needs to learn the static part as also the dynamic part of the plant. In the mean derivatives methodology the universal approximator of functions must learn only the static function (of course the mean derivatives) of the plant. The dynamic part of the plant in the mean derivatives methodology is in charge only and exclusively of Euler integrator. In the instantaneous derivatives methodology the universal approximation of functions must learn only the static function (of course the instantaneous derivatives) of the plant. The dynamic part is in charge of the highest order integrators. In this way, a universal numerical integrator can be defined mathematically in three categories: universal numerical integrators of the NARMAX type, universal numerical integrators with mean derivatives and universal numerical integrators with instantaneous derivatives. Next, each of them is mathematically defined separately.

(i) Universal numerical integrators of the NARMAX type. They are defined as follows:

$$\begin{aligned} y(t+\Delta t) = \hat{f}_{NARMAX}\big(y(t), y(t-\Delta t), \ldots, y(t-n_x\Delta t);\\ u(t), u(t-\Delta t), \ldots, u(t-n_u\Delta t), \hat{w}\big) \end{aligned} \tag{16}$$

Here the function $\hat{f}_{NARMAX}(\ldots)$ can be any universal approximator of functions. The symbol $\hat{}$ means that the function $\hat{f}_{NARMAX}(\ldots)$ is an estimate of the nonlinear dynamics of the system considered, and, not, an exact value. Since the function $\hat{f}_{NARMAX}(\ldots)$ is a universal approximator of functions, the estimate obtained can always be smaller than a desired error (see [25]). The vector $\hat{w}$ are the parameters to be estimated through supervised training.

(ii) Universal numerical integrator with mean derivatives. They are defined as follows:

$$y(t+\Delta t) = \hat{f}_{MD}(y(t), u(t), \hat{w}).\Delta t + y(t) \tag{17}$$

Here, the function $\hat{f}_{MD}(y(t), u(t), \hat{w}) = \tan_{\Delta t}{}^k\alpha^i = \frac{y^i(t+\Delta t)-y^i(t)}{\Delta t}$ is the mean derivatives function and can be represented using any universal approximator of functions. Both the mean derivatives functions and the instantaneous derivatives functions are static functions. It is interesting to observe that a static function, in computational terms, is much less expensive to be learned, through a nonlinear estimation problem, than a dynamic function (see [13]). It can also be mathematically proved that Equation (17) is discrete and exact general solution for autonomous nonlinear ordinary differential equations (see [64-66]).

(iii) Universal numerical integrators with instantaneous derivatives. The general simple-step Runge-Kutta method is defined as (see [8]):

$$y(t+\Delta t) = y(t) + \Delta t \cdot \Phi\left(t, y(t), \Delta t, \hat{w}\right) \tag{18a}$$

where,

$$\Phi(t, y(t), \Delta t, \hat{w}) = \sum_{r=1}^{R} C_r \cdot k_r \tag{18b}$$

$$k_1 = \hat{f}_{ID}(y(t), t, \hat{w}) \tag{18c}$$

$$k_r = \hat{f}_{ID}\left(t + \Delta t \cdot a_r, y(t) + \Delta t \cdot \sum_{s=1}^{r-1} b_{rs} \cdot k_s, \hat{w}\right) \tag{18d}$$

$$a_r = \sum_{s=1}^{r-1} b_{rs} \quad r = 2, 3, \ldots, R \tag{18e}$$

The function $\hat{f}_{ID}[\ldots]$ is defined as the function of instantaneous derivatives. In the pioneering work of Wang and Lin in [13] a neural network with RBF (Radial Basis Function) architecture is coupled to a fourth order Runge-Kutta type integration structure. These authors introduced in the literature the terminology Runge-Kutta neural network.

The equations from (18a) to (18e) are the general expressions of a Runge-Kutta numerical integrator of any order. To obtain the coefficients C_r, K_r, a_r, and b_{rs}, it is necessary to expand the ordinary differential equation of the dynamic system $\dot{y} = f(y, t)$ into Taylor series. In this expansion will appear high-order partial derivatives. Thus, the coefficients of Runge-Kutta method will appear when the high-order partial derivatives are eliminated through one non-trivial algebraic manipulation. Thus, a Runge-Kutta integrator 4-5 means a Taylor series expansion up to the fourth order, with a fifth-order truncation error, and with the high-order partial derivatives being eliminated. For a complete mathematical demonstration of the Runge-Kutta integrator 3-4 see [8].

Complementarily, the general multiple-step Adams-Bashforth method is defined as:

$$\hat{y}(t + \Delta t) = y(t) + \frac{h}{\alpha}.\sum_{i=1-o}^{0} \beta_i \cdot \hat{f}_{ID}[y(t + i \cdot \Delta t), \hat{w}] \tag{18f}$$

It is interesting to note that if both coefficients k_r's of Equations (18c) and (18d), and coefficients β_i's of Equation (18f), are different from those appearing in the integrator structures (see Section 2 of this article), the universal numeric integrator probably will converge to the desired solution. However, in these cases, there will be no theoretical assurances that the functions $\hat{f}_{ID}[\ldots]$ will converge to the functions of instantaneous derivatives, that is, $\hat{f}_{ID}[\ldots]$ will learn another thing.

The mathematical treatment of multi-step universal numerical integrators is much simpler than single step integrators. For example, if the backpropagation algorithm is adapted to multi-step expressions, simply apply the rule that the derivative of a sum is the sum of the derivatives. However, if the backpropagation algorithm is adapted to the single-step expressions, it is necessary to apply the chain rule of the differential calculus, which is much more complex. The remainder of this article is devoted to a theoretical and practical discussion of the implications of the mathematical definitions presented in this section.

5. Classification of Universal Numerical Integrators.

Non-linear function mapping properties are the central point for the use of universal numeric integrators in control. Models of dynamic systems and their inverse ones have immediate utility in control. It is important to identify the direct model of the plant of the physical system to be studied. The identification of this model is the starting point, in order to use any of the possible non-linear control structures found in the literature.

The procedure of training a universal numerical integrator to represent the dynamics of a system will be referred to here as direct modeling. For example, a structure with a Multilayer Perceptrons neural network to achieve this is shown in Figure 1. As can be seen, the neural network model is placed in parallel with the system and the error between the system and the network outputs is used as the network training signal. This learning structure is a classic supervised learning problem where the need for a professor (training standards) is imperative.

An important question in the control context is the dynamic nature of the systems being studied. It is assumed that the system is governed (e.g., [72]) by the following non-linear equation of discrete time differences.

$$y^p(t+1) = f\left[y^p(t), \ldots, y^p(t-n+1); u(t), \ldots, u(t-m+1)\right] \tag{19}$$

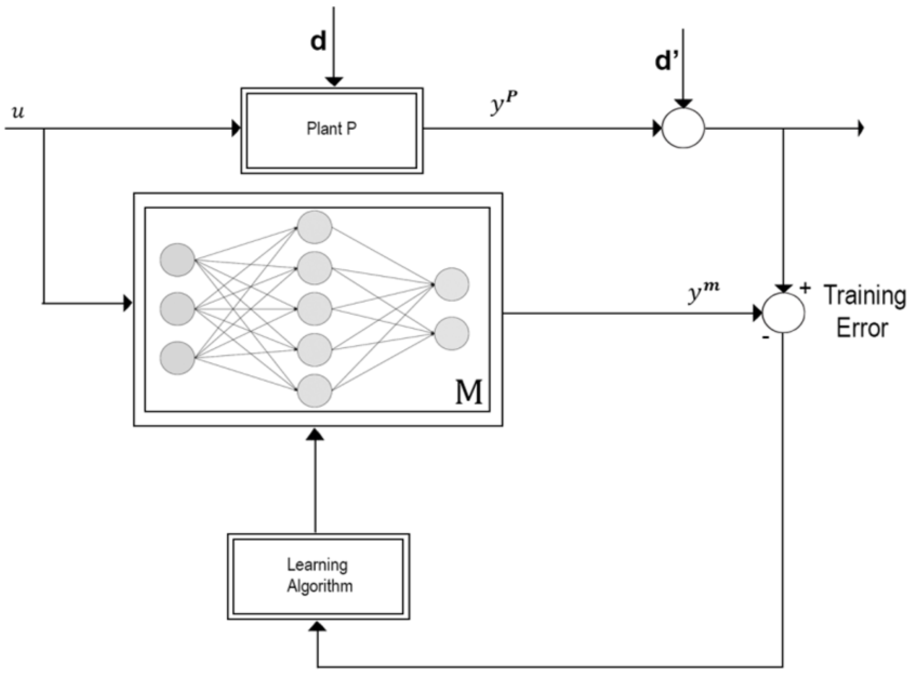


FIGURE 1. Identification of dynamic systems through neural networks (Source: see [55])

In this way, the output of the system y^p in time $t+1$ depends on the n delayed values of y^p and the m delayed inputs of the control variable u. An obvious approach to system modeling is to choose the input/output structures of the neural network as being the same as that of the system. Denoting the output of the network as (3) one then has,

$$y^m(t+1) = \hat{f}\left[y^p(t), \ldots, y^p(t-n+1); u(t), \ldots, u(t-m+1), w\right] \tag{20}$$

Here, $\hat{f}$ represents the non-linear input/output mapping of the network. Note that the entry for the network includes the past values of the system. If it is assumed that after a training period the network provides a good representation of the plant (i.e., $y^m \cong y^p$) then the network can be used independently of the plant. Such a neural model can be described by:

$$y^m(t+1) = \hat{f}\left[y^m(t), \ldots, y^m(t-n+1); u(t), \ldots, u(t-m+1), w\right] \tag{21}$$

As mentioned in Section 4, an artificial neural network can be considered as a particular case of a universal numerical integrator. In this way, in Figure 1 where artificial neural networks are seen, a universal numerical integrator can also be associated with this figure. In the original where Figure 1 was extracted (see [73]), the dynamic system modeling methodology employed was NARMAX.

In this section a general approach is taken with the purpose of explicitly showing how to use universal numerical integrators to represent systems of ordinary differential equations in control schemes, where a discrete internal forward model is required (e.g., [13,18,56]). The internal structure of the numerical integration algorithms is used to obtain a model of a universal approximator of functions that only needs to learn the instantaneous derivatives functions or the mean derivatives functions of the original model, represented by systems of ordinary differential equations. In this methodology, the numerical integrators and the universal approximator will work together to represent the discretized dynamics of the original system.

In control schemes employing universal approximators of functions, the usual approach is to use the numerical integrator structures to generate the training standards of the network and only then to train the neural network through supervised learning using the NARMAX methodology. There are at least two difficulties with this approach: the main one is having to deal with many inputs in the universal approximation and, therefore, many parameters to be adjusted, making difficult the approximation training; and the second is the impossibility of varying the integration step.

In the new approach that will be adopted here, the number of network entries can be reduced and the integration step constantly varied. In the usual approach numerical integrators are only used to generate the training standards and in the approach described here it can work in conjunction with the universal approximator of functions throughout the training and simulation process.

When a structure of any numerical integrator model is coupled to a universal approximator of function, it is possible to generate a universal numerical integrator to approximate only the instantaneous derivatives or the mean derivatives functions of a mathematical or experimental model governed by non-linear ordinary differential equations. In this type of approach, the difficulty in dealing with many inputs in the training of the universal approximator of functions is alleviated, since it is only necessary to learn an algebraic and static function where the network inputs are only the occurrences of the state and control variables in their respective fields of activity. Recall that in the conventional approach sometimes it is necessary to deal with many delayed entries of the state and control variables, but in this new methodology this will no longer be necessary.

Thus, there are two forms of supervised training that the methodology of the universal numerical integrators allows to realize when it is known or when it is not known a priori the instantaneous derivatives functions of the theoretical model. These two methodologies are the direct methodology and the empirical or indirect methodology (see Figure 2).

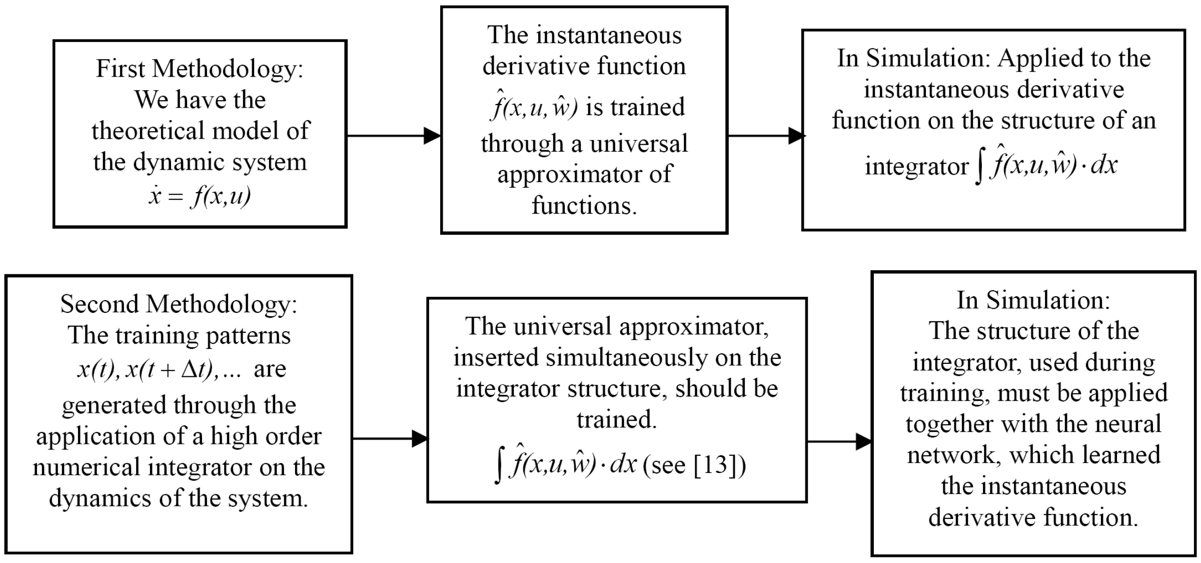

First Methodology: We have the theoretical model of the dynamic system $\dot{x} = f(x,u)$
The instantaneous derivative function $\hat{f}(x,u,\hat{w})$ is trained through a universal approximator of functions.
In Simulation: Applied to the instantaneous derivative function on the structure of an integrator $\int \hat{f}(x,u,\hat{w}) \cdot dx$
Second Methodology: The training patterns $x(t), x(t+\Delta t), \ldots$ are generated through the application of a high order numerical integrator on the dynamics of the system.
The universal approximator, inserted simultaneously on the integrator structure, should be trained. $\int \hat{f}(x,u,\hat{w}) \cdot dx$ (see [13])
In Simulation: The structure of the integrator, used during training, must be applied together with the neural network, which learned the instantaneous derivative function.

FIGURE 2. Two distinct ways of training a universal approximator in the structure of a numerical integrator

Direct Methodology. The first methodology is to simply randomly generate sufficient values of the instantaneous derivatives function within a domain of interest and directly train the universal approximator of functions to learn the instantaneous derivatives function of the theoretical model (see [56]). After the training, the simulation of the results should be done by applying the instantaneous derivative function trained, by the universal approximator of functions, in place of the theoretical on the structure of the integrator. For the analysis of the compatibility of the errors obtained by the network and the integrator, the numerical solution of the integrator must be obtained by applying the two derivative functions, the trained and the original, so that the results of the solution propagation are compared. In this methodology, the numerical integrator will only be used in the simulation of the results. In addition, this methodology can only be applied if the mathematical model of the plant is known.

Empirical or Indirect Methodology. In the second methodology the temporal solutions of the dynamic system $x(t)$, $x(t+\Delta t), \ldots, x(t+n\cdot\Delta t)$ are generated for a sufficient number of initial conditions and control values through the structure of an integrator, preferably of high order to guarantee the high precision of the training standards. The learning of the instantaneous derivatives function by the universal approximator of functions is only possible if the universal approximator is trained tied to the structure of the numerical integrator (see [13]). The analysis of the error obtained by the integrator set and neural network is also performed by comparing the numerical solutions propagated by the original and trained derivative functions. In this methodology, the numerical integrator is used in the generation of training standards and is required during training and network simulation. This methodology can be used both for theoretical models and also for real world plants where the theoretical model is not known.

Given this brief introduction to universal numerical integrators, in the next part, each of the three methodologies derived from this concept is explained in more detail. The three methodologies in question are NARMAX methodology, instantaneous derivatives methodology and mean derivatives methodology.

5.1. A brief summary of NARMAX methodology.

A universal approximator of functions can be used to represent a non-linear dynamic system of the type given by Equation (22) from a discrete set of input/output training patterns,

$$\dot{y} = f(y, u) \tag{22}$$

In this equation, the function $f(.)$ is invariant in time (see [73]). In order to do this, it is usually assumed the possibility of approximation of the dynamic system given by Equation (22) by a discrete model of NARMAX type, given by:

$$y(t+\Delta t) = f(y(t), y(t-\Delta t), \ldots, y(t-n_x\Delta t); u(t), u(t-\Delta t), \ldots, u(t-n_u\Delta t), \hat{w}) \tag{23}$$

where n_x, n_u, Δt are constants that must be adjusted depending on the problem being addressed and the desired accuracy. This possibility is considered for the use of any universal approximator of functions to act as a discrete model as given by Equation (23). For example, for the particular case of an artificial neural network, the size of the neural network (number of layers and number of neurons per layer) can be adjusted to reach the desired accuracy for a given choice of n_x, n_u, Δt.

At this point it is convenient to remember a similar situation that occurs when numerical integrators and dynamical systems are used as in Equation (22), they are treated by discrete approximations as in Equation (23). Neural networks of multi-layer perceptron-type with one inner layer are sufficient to accomplish this representation task for virtually any function found in many engineering applications (see [25]). It is important to observe that the great difference between NARMAX methodology and the methodology that uses an Euler type integrator with mean derivatives is that in the first one, the universal approximator outputs are the future values $y(t+\Delta t)$, while in the second, the outputs are the mean derivatives functions at the instant t.

5.2. **Instantaneous derivatives methodology.** In the instantaneous derivatives methodology (e.g., [13,18,56-59]), the universal approximator of functions will only play the role of a static function given by $\vec{\hat{f}}(\vec{y},\vec{u}) \cong f(\vec{y},\vec{u})$ where the final dynamics, governed by the differential equations $\vec{\dot{y}} \cong \vec{\hat{f}}(\vec{y},\vec{u})$, will have its answer obtained by means of some structure of numerical integration of high order, being this one of simple steps or multiple steps.

Wang and Lin in [13] originally developed this methodology using a Runge-Kutta 4-5 single-step integrator coupled to a neural network with Radial Basis Functions (RBF) architecture. In this methodology, it is important to note that the training and simulation phases will basically depend on two approximation errors: one, due to the mean quadratic error of the supervised training that will be subject to the universal approximator; and the other, due to the order and integration step of the numerical integrator used. It is known that, in general, the greater the order of the integrator used and the smaller the integration step used, the greater the precision obtained in the simulation. Figure 3 schematically illustrates the instantaneous derivatives methodology using a single or multiple step integrator. In addition, in [18] the universal numerical integrator called the Adams-Bashforth neural networks can be found. In [18] a neural network with Multilayer Perceptrons architecture is coupled to a fourth-order Adams-Bashforth integrator.

As can be seen in Figure 3, the great advantage of coupling a universal approximator of functions in a high-order numerical integrator structure is that the instantaneous derivatives function, which governs the dynamics of the system considered, can be adapted to real-world problem, where only the input and output data of the plant are known. Only

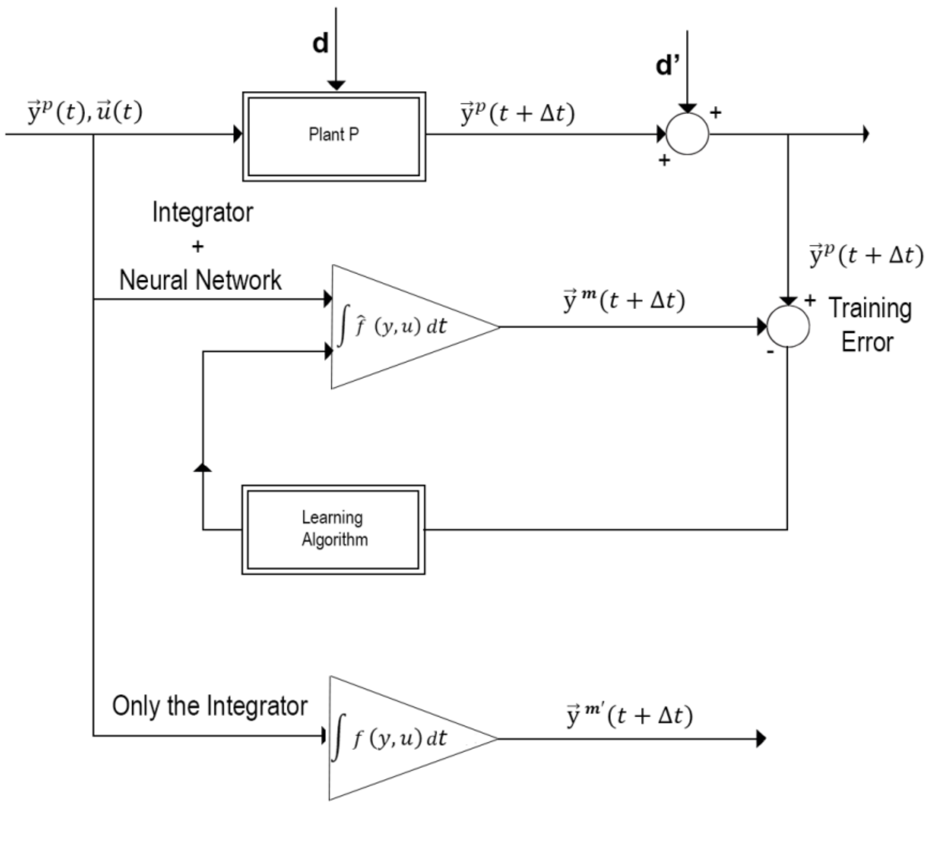


FIGURE 3. Illustrative scheme of the use of a universal approximator coupled to a high-order numerical integrator structure (Source: see [67])

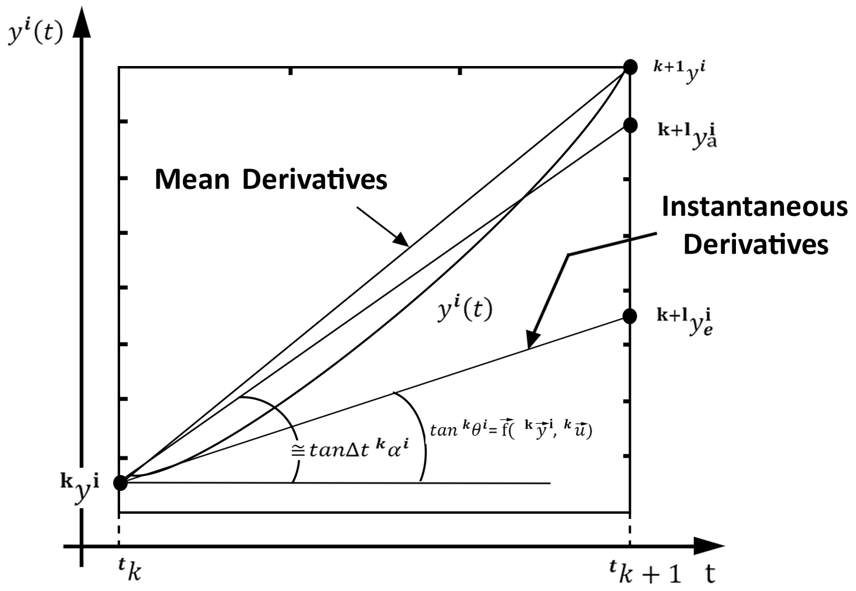
with the use of a high-order numerical integrator, without the use of a universal approximator, it is impossible to adapt the dynamics of the system in question to real-world data. This impossibility can be visualized in Figure 3 through variable $\vec{y}^{m'}(t+\Delta t)$.

The instantaneous derivatives methodology has at least two advantages over the NARMAX methodology: the fact that neural training is static and not dynamic (it is only necessary to learn the static function $\vec{f}(\vec{y},\vec{u})$) and allow the variation of the integration step in the simulation phase. The NARMAX methodology requires a new training, if the integration step Δt is to be varied.

5.3. **The mean derivatives methodology.** In the mean derivatives methodology, as its name says, it is avoided to work with the instantaneous derivatives, replacing them with the mean derivatives. Details of this methodology and its detailed mathematical development can be found in [60-66]. Thus, given the discrete numerical solution for a dynamic system of the type $\dot{\vec{y}} = \vec{f}(\vec{y},\vec{u})$, expressed by the vector sequence ${}^{0}\vec{y}^{i}$, ${}^{1}\vec{y}^{i}$, ${}^{2}\vec{y}^{i}$, …, ${}^{k}\vec{y}^{i}$, ${}^{k+1}\vec{y}^{i}$, …, ${}^{n}\vec{y}^{i}$ for ${}^{k}\vec{y}^{i} = \vec{y}^{i}[t+k\cdot\Delta t]$ and ${}^{k+1}\vec{y}^{i} = \vec{y}^{i}[t+(k+1)\cdot\Delta t]$; presently, by definition, the mean derivatives of the system is defined by (see Figure 4):

$$\tan_{\Delta t}{}^{k}\alpha_{j}^{i} = \frac{{}^{k+1}y_{j}^{i} - {}^{k}y_{j}^{i}}{\Delta t}, \text{ for } j = 1,2,\ldots,m \tag{24}$$

where m is the total number of state variables of the system considered. The vector notation for the mean derivatives can be expressed by $\tan{}_{\Delta t}{}^{k}\alpha^{i} = \left[\tan_{\Delta t}{}^{k}\alpha_{1}^{i} \quad \tan_{\Delta t}{}^{k}\alpha_{2}^{i} \ \ldots \ \tan_{\Delta t}{}^{k}\alpha_{m}^{i}\right]^{T}$. Figure 4 illustrates the basic differences between mean derivatives and instantaneous derivatives. Notice that $\lim\limits_{\Delta t\to 0}\tan_{\Delta t}{}^{k}y^{i} = \vec{f}\left[{}^{k}\vec{y}^{i},{}^{k}\vec{u}\right]$ and that the mean derivatives methodology is dependent on the integration step Δt. Besides this, the instantaneous derivatives methodology is independent of it. Thus, when it is desired to change the integration step in the mean derivatives methodology, it will be necessary to carry out a new supervised training. In [64] it is demonstrated the following computational algorithm for neural solution of the dynamic system $\dot{\vec{y}} = \vec{f}[\vec{y},\vec{u}]$, through the



$y^{i}(t)$
${}^{k+1}y^{i}$
${}^{k+1}y_{a}^{i}$
Mean Derivatives
Instantaneous Derivatives
$y^{i}(t)$
${}^{k+1}y_{e}^{i}$
$\tan{}^{k}\theta^{i} = \vec{f}({}^{k}\vec{y}^{i}, {}^{k}\vec{u})$
$\cong \tan\Delta t\,{}^{k}\alpha^{i}$
${}^{k}y^{i}$
t_{k}
$t_{k}+1$
t

FIGURE 4. Graphical representation of the mean derivatives $\tan_{\Delta t}{}^{k}\alpha^{i}$ and the instantaneous derivatives function $\tan{}^{k}\theta^{i} = \vec{f}\left({}^{k}\vec{y}^{i},{}^{k}\vec{u}\right)$ for the dynamic system $\dot{\vec{y}}^{i} = \vec{f}\left[{}^{k}\vec{y}^{i},{}^{k}\vec{u}\right]$ (Source: see [66,67])

mean derivatives methodology (easily expandable and applicable to any type of universal approximator of functions).

1) Given the finite domains of interest $\left[y_j^{\min}(t_0), y_j^{\max}(t_0)\right]^{n_1}$ in t_0 for $j = 1, 2, \ldots, n_1$ of the state variables and also the finite domains of interest $\left[u_j^{\min}(t_0), u_j^{\max}(t_0)\right]^{n_2}$ for $j = 1, 2, \ldots, n_2$ of control variables, we generate q random vectors, according to a uniform distribution within these intervals in the form:

$$p_i = \left[y_1^i(t_0) \;\; y_2^i(t_0) \;\; \ldots \;\; y_{n_1}^i(t_0); u_1^i(t_0) \;\; u_2^i(t_0) \;\; \ldots \;\; u_{n_2}^i(t_0)\right]^T \tag{25a}$$

and

$$P = [p_1 : p_2 : \ldots : p_q]_{(n_1+n_2)xq} \tag{25b}$$

which will be the input training patterns of the universal approximator of functions at time t_0.

2) By using a high-order integrator, all initial conditions p_i are propagated for $i = 1, 2, \ldots, q$. Thus, all states are generated in the instant $t_0 + \Delta t$, that is,

$$p_i^{\Delta t} = \left[y_1^i(t_0 + \Delta t) \quad y_2^i(t_0 + \Delta t) \;\; \ldots \;\; y_{n_1}^i(t_0 + \Delta t)\right]^T \tag{26a}$$

and

$$P^{\Delta t} = \left[p_1^{\Delta t} : p_2^{\Delta t} : \ldots : p_q^{\Delta t}\right]_{n_1xq} \tag{26b}$$

3) The output vectors T_i are determined (being this denominated direct methodology of the mean derivatives) in the form:

$$\begin{aligned} T_i &= \frac{1}{\Delta t}\left[y_1^i(t_0 + \Delta t) - y_1^i(t_0) \quad y_2^i(t_0 + \Delta t) - y_2^i(t_0) \;\; \ldots \;\; y_{n_1}^i(t_0 + \Delta t) - y_{n_1}^i(t_0)\right]^T \\ &= \left[\tan_{\Delta t} {}^k\alpha_1^i \quad \tan_{\Delta t} {}^k\alpha_2^i \;\; \ldots \;\; \tan_{\Delta t} {}^k\alpha_{n_1}^i\right]^T = \left(\tan_{\Delta t} {}^k\alpha^i\right) \end{aligned} \tag{27a}$$

and

$$T = [T_1 : T_2 : \ldots : T_q]_{n_1xq} \tag{27b}$$

Since the function $\tan_{\Delta t} {}^k\alpha^i$ is also autonomous (e.g., [64-66]), then only a Δt propagation on all the initial conditions of the input vectors p_i is sufficient, for $i = 1, 2, \ldots, p$, in order to set up the training patterns P and T required by the universal approximator of functions.

4) Given the input vectors P and the output vectors T required by the universal approximator, then this universal approximator can undergo a supervised training process to learn the mean derivatives functions.

5) After consolidating the supervised training of the universal approximator, within a stipulated mean square error, it will be possible to simulate the dynamics from the following relation:

$${}^{k+1}\hat{y}^i = \tan_{\Delta t} {}^k\alpha^i \cdot \Delta t + {}^ky^i \tag{28}$$

It can be demonstrated mathematically (see [64,67]) that Equation (28) is a discrete and exact general solution for the system of nonlinear ordinary differential equations $\vec{y} = \vec{f}(\vec{y}, \vec{u})$, where $\tan_{\Delta t} {}^k\alpha^i$ is the mean derivatives function with step Δt which approximates the instantaneous derivatives function $\vec{f}(\vec{y}, \vec{u})$ at the starting point ${}^ky^i = \left[{}^ky_1^i \;\; {}^ky_2^i \; \ldots \; {}^ky_{n_1}^i\right]$. It is important to note that the application of Equation (28) will only be allowed if the mean derivatives function is known. Theoretically, any universal approximator of functions can be used for this purpose. In addition, the precision obtained by Equation (28) is the same if a higher order integrator is used (see [64]).

Some comparative considerations between the instantaneous derivatives methodology and the mean derivatives methodology are given below. Thus, the instantaneous derivatives methodology uses variable integration step in the simulation phase and fixed step in the training phase. However, accurate results are achieved only with high order integrators. On the other hand, the mean derivatives methodology uses a fixed integration step both in the training phase and in the simulation phase, being applicable only to the first order Euler type integrator, but whose accuracy is similar to that obtained by the instantaneous derivatives coupled to integrators of the highest order.

To conclude this section, it is important to emphasize that the algorithm described here on mean derivatives uses the direct methodology for the determination of the mean derivatives. This is important because, as mentioned earlier, there are two approaches to the determination of the mean derivatives: the direct approach, as explained in steps 1) to 5), and the indirect or empirical approach. A few words to differentiate these two methodologies are timely. Thus, for this purpose, the direct approach is schematized in Figure 5 and the indirect approach is schematized in Figure 6. In both Figures 5 and 6 we have a simplified scheme of a neural architecture conveniently named Euler neural networks. However, there are significant differences between direct and indirect approaches.

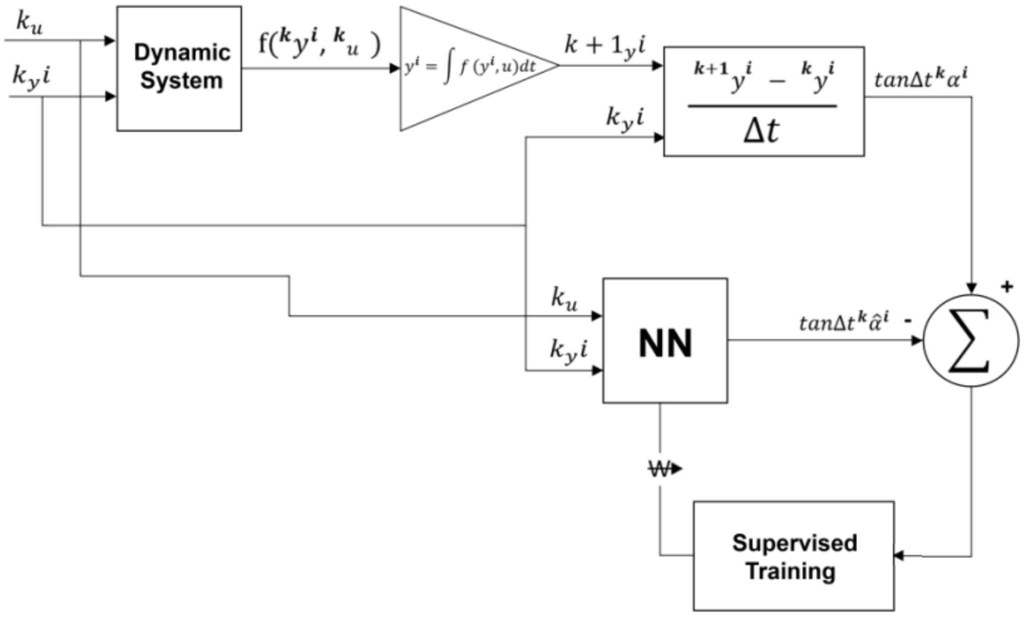

^{k}u
$^{k}y^{i}$
Dynamic System
$f(^{k}y^{i}, {}^{k}u)$
$y^{i} = \int f(y^{i}, u)dt$
$^{k+1}y^{i}$
$\frac{^{k+1}y^{i} - {}^{k}y^{i}}{\Delta t}$
$tan\Delta t^{k}\alpha^{i}$
^{k}u
$^{k}y^{i}$
NN
$tan\Delta t^{k}\hat{\alpha}^{i}$
+
-
$\sum$
W
Supervised Training

FIGURE 5. Direct approach for the determination of the mean derivatives

As shown in Figure 5, the mean derivatives are presented directly to the universal approximator in the direct approach. Already in the indirect or empirical approach, as schematized in Figure 6, the mean derivatives are learned indirectly through the value $^{k+1}\hat{y}^{i}$. Mathematically, this difference is significant, since in [67] considering the particular case of the universal approximator being represented by a neural network with feedforward architecture, it is verified that in the direct approach, the backpropagation remains unchanged. However, in the empirical approach, backpropagation should be slightly modified.

This classification between the direct and empirical approaches is also applied to the instantaneous derivatives methodology. However, in the latter, real world data are applicable only in indirect or empirical methodology. Thus, Wang and Lin in [13] apply the empirical approach, using a Rung-Kutta 4-5 integrator in the instantaneous derivatives methodology and Melo and Tasinaffo in [18] apply the empirical approach using an Adams-Bashforth 4 integrator, also in the instantaneous derivatives methodology. In the

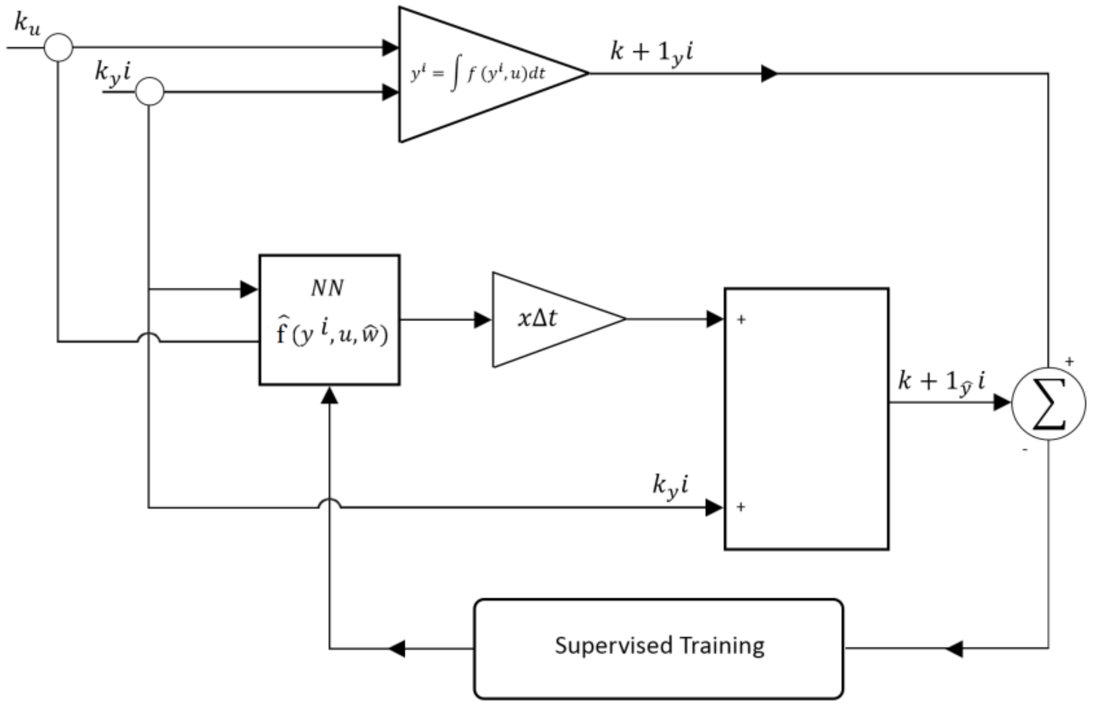


Figure 6. Indirect or empirical approach for the determination of the mean derivatives

mean derivatives methodology, both the direct approach and the indirect approach can be applied to real-world data, but it is experimentally verified that the direct approach is the most indicated (see [59]). The direct mean derivatives methodology applies to real-world problems, since the mean derivatives are easily calculable from the input/output patterns. However, the direct methodology of instantaneous derivatives cannot apply to real world problems, since the instantaneous derivatives cannot be measured or calculated directly. However, it is still possible to apply the direct methodology of the instantaneous derivatives to theoretical problems, when we know the analytic function $f(y, u)$ of the differential equation $\dot{y} = f(y, u)$. This classification between direct and indirect approach does not apply to the NARMAX methodology.

In [66] an approximate continuous solution mathematically is developed to allow the variation of the integration step in the mean derivatives methodology. However, this approximate solution does not apply to dynamic systems with control variables. In addition, finite difference approximations for instantaneous derivatives were already in use by Euler (see [74]) in 1768. This procedure is known as Euler's merhod which is the origin of the mean derivatives methodology.

6. **Classification of Major Neural Control Structures.**

Basically there are three control structures, which can be used in conjunction with a universal numeric integrator. These structures are IMC (Internal Model Control) structures, predictive control structures and adaptive control structures (e.g., [68]). However, there are currently very few theoretical or practical applications of universal numerical integrators combined with control theory, and therefore the possibilities for exploration within this context are many. Almost all existing work on control theory, involving universal approximator of functions, is using the NARMAX methodology. Studies on control theory involving universal numerical integrators of type (ii) and (iii) as described in Section 4 are rare.

7. **Conclusions.** The possibility of using numerical integrators of differential equations systems combined with universal approximators of functions was considered. It has been shown that the structure of these universal numerical integrators can be exploited to obtain universal discrete models where the universal approximator of function has only to learn and approximate the instantaneous derivatives functions or the mean derivatives functions, which are simply of static algebraic nature. Some conclusions can be highlighted (see also Table 1):

a) it is a simpler task to train a universal approximator of functions to learn an algebraic and static function than to train it to learn the discrete dynamic model;

b) the architecture of the universal numerical integrator is simpler with regard to the number of layers and number of neurons since it does not have to learn the law of dynamics but only the instantaneous derivatives or mean derivatives functions;

c) the use of the universal numerical integrator for systems of ordinary differential equations, as an approximate model of discrete time, does not destroy the parallel processing characteristic, since some algorithms of numerical integration only involve calculations and evaluations of linear combinations of the universal approximator trained;

d) the flexibility of being able to vary the step size and the order of the universal numerical integrator can be used to control the desired accuracy for the output response of the system. However, one must train the universal approximator of functions with a tolerance compatible with that desired in the universal numerical integrator;

e) when the dynamic response times are not too small and the mathematical model of ordinary differential equations evaluated is reasonably good, then the structure of the universal numerical integrator can be directly used as a discrete internal model;

f) even in situations where a theoretical mathematical model cannot be evaluated and there are only pairs of dynamic input/output information obtained experimentally, yet the structure of the universal numerical integrator can be trained to obtain a discrete internal model in schemas of control.

It can also be concluded that there are three distinct ways of solving the problem of nonlinear programming required by control structures:

a) use only a universal approximator of functions to approximate the dynamics of the system through the criterion of delayed inputs (NARMAX methodology);

b) to use only the numerical integrator as discretized model of the dynamic system (applicable only to theoretical models);

c) jointly use the numerical integrator and a universal approximator of functions that will represent the instantaneous derivatives functions or mean derivatives functions to represent the dynamic system (applicable to theoretical and empirical models of the real world).

Using only the universal approximator of function leads to a more robust situation and therefore more difficult to be trained, besides not allowing the variation of the integration step. Using only the structure of the numerical integrator with the instantaneous derivatives functions makes it difficult to adapt the theoretical model of the dynamic system with that of the real plant.

Using the numerical integrator structure with the instantaneous derivatives functions or mean derivatives functions, estimated by the universal approximator of functions, allows us to construct a universal numerical integrator less robust and therefore easier to train, besides being able to design, in this case, a dynamic model with variable integration step and that can be adapted in real time with respect to the original system of the plant.

The major disadvantage of the latter method is that the computation of the partial derivatives becomes more complex and complicated, and for each type of integrator to be used we will have a different expression for the partial derivatives and, in general, these

TABLE 1. Comparative description between the methods of the instantaneous derivatives and the mean derivatives

Influence of the numerical integrators theoretical development, when they are coupled to the architecture of a universal approximator of functions.			
	Instantaneous Derivatives Methodology		**Mean Derivatives Methodology**
	Runge-Kutta Type Simple Step Integrators	**Adams-Bashforth Type Multi-Step Integrators**	**Euler Type Simple Step Integrators**
1	Chain rules increasingly complex with increasing the order of the integrator used in neural training as proposed by Wang and Lin in [13].	Only the presence of a linear combination in the backpropagation algorithm with the growth of the integrator order and as proposed by Melo and Tasinaffo in [18].	Simpler algorithms. Simply the increased backpropagation algorithm of the product Δt and as proposed by Tasinaffo et al. (see [67]).
2	Good numerical precisions only achievable with higher order integrators.	Good numerical precisions only achievable also with higher order integrators.	Good precisions attainable simply with Euler-type integrator designed with the concept of mean derivatives.
3	The number of training patterns does not increase linearly with increasing order of the integrator used.	The number of training patterns always increases linearly with increasing order of the integrator (see [18,59]).	The number of training patterns is always of the same magnitude as single-step integrators.
4	Variable step method and less sensitive to step variation. To vary the integration step it is not necessary to train a new network.	Variable step method and more sensitive to step variation. To vary the integration step it is not necessary to train a new network.	Unconditionally fixed step method. To change it you need to perform a new supervised training.
5	Runge-Kutta is obtained by the Taylor series development on the original instantaneous derivatives function, and therefore, the larger the order of the integrator used then the greater the numerical precision of the solution propagation. Wang and Lin in [13] do not carry out a study predicting control variables in their mathematical model.	The Adams-Bashforth is obtained by replacing the original derivative function with a polynomial interpolation. Thus, not necessarily, the higher the order of the integrator the greater the numerical precision of the propagated solution. This occurs because polynomials of higher degrees oscillate around the original function.	It is mathematically proved that the Euler integrator, designed with mean derivatives, obtains equivalent precision to any higher-order integrator designed with instantaneous derivatives. See [64,67].
6	It is difficult to obtain mathematical and open its generalization in relation to the order of integration. Wang and Lin in [13] apply this methodology only to the fourth-order Runge-Kutta. There is no such methodology, for example, combined with fuzzy logic.	Easily obtained and generalized from the mathematical point of view by Melo and Tasinaffo in [18]. Found in literature only combined with neural network with MLP architecture in [18,56-59]. There is also no such methodology combined with fuzzy logic.	Easily obtained from a mathematical point of view. In [61] the mean derivatives methodology is combined with neural networks and in [65] it is combined with fuzzy logic.
7	Application in predictive control extremely difficult from the mathematical point of view and still open. Open also its application in adaptive control and in IMC structures.	More easily applicable in control theory than the Runge-Kutta method. However, there is little known in the literature in this regard (see [69,70]).	Easily applicable in control theory. In [67,71] the mean derivatives methodology is used in predictive control. In adaptive control and in IMC structures the knowledge is still open.
8	Real world models obtained only by empirical (or indirect) methodology.	Real world models obtained only by empirical (or indirect) methodology.	Real world models obtained both by empirical methodology and by direct methodology.

expressions will be as more complex and difficult to obtain as the larger the order of the integrator employed. To conclude this article, Table 1 presents a comparative description between the methods of the instantaneous derivatives and the mean derivatives.

Acknowledgments. Initially, the authors of this article would like to thank the reviewers for the excellent technical work and good suggestions, especially for the friendship we have for Mr. Yan Shi. We would also like to thank Professor Atair Rios Neto, the Technological Institute of Aeronautics (ITA), Casimiro Montenegro Filho Foundation (FCMF) and, The Ecossistema Negocios Digitais Ltda for its financial support of this research. And finally, we thank God that He has helped us through this hard and pleasant work.

REFERENCES

[1] J. M. S. Telles, *Lições de Equações Diferenciais Ordinárias*, IMPA – Instituto de Matemática Pura e Aplicada e CNPq, Rio de Janeiro, 1979.

[2] R. E. Kalman, A new approach to linear filtering and prediction problems, *Transaction of the ASME – Journal of Basic Engineering*, pp.35-45, 1960.

[3] P. Henrici, *Elements of Numerical Analysis*, John Wiley & Sons, New York, 1964.

[4] L. Lapidus and J. H. Seinfeld, *Numerical Solution of Ordinary Differential Equations*, Academic Press, New York and London, 1971.

[5] J. D. Lambert, *Computational Methods in Ordinary Differential Equations*, John Wiley & Sons, New York, 1973.

[6] M. Vidyasagar, *Nonlinear Systems Analysis*, Networks Series, Electrical Engineering Series, Prentice-Hall, New Jersey, 1978.

[7] M. Braun, Differential equations and their applications: An introduction to applied mathematics, in *Applied Mathematical Sciences*, 3rd Edition, Springer-Verlag, New York, 1983.

[8] K. R. Rao, *A Review on Numerical Methods for Initial Value Problems*, INPE-3011-RPI/088, São José dos Campos/SP, Brazil, 1984.

[9] D. Sarafyan, Improved sixth-order Runge-Kutta formulas and approximate continuous solution of ordinary differential equations, *Journal of Mathematical Analysis and Applications*, vol.40, pp.436-445, 1972.

[10] A. Kumar and T. E. Unny, Application of Runge-Kutta method for the solution of non-linear partial differential equations, *Appl. Math. Modelling*, pp.199-204, vol.1, 1977.

[11] A. Rios Neto and K. R. Rao, A stochastic approach to global error estimation in ODE multistep numerical integration, *Journal of Computational and Applied Mathematics*, vol.30, no.3, pp.257-281, 1990.

[12] T. E. Simos, Runge-Kutta-Nyström interpolants for the numerical integration of special second-order periodic initial-value problems, *Computers Math. Applic.*, vol.26, no.12, pp.7-15, 1993.

[13] Y.-J. Wang and C.-T. Lin, Runge-Kutta neural network for identification of dynamical systems in high accuracy, *IEEE Trans. Neural Networks*, vol.9, no.2, pp.294-307, 1998.

[14] J. C. Chiou and S. D. Wu, On the generation of higher order numerical integration methods using lower order Adams-Bashforth and Adams-Moulton methods, *Journal of Computational and Applied Mathematics*, vol.108, pp.19-29, 1999.

[15] D. Goeken and O. Johnson, Fifth-order Runge-Kutta with higher order derivative approximations, *The 15th Annual Conference of Applied Mathematics*, Univ. of Central, Oklahoma, pp.1-9, 1999.

[16] B. Cockburn and C.-W. Shu, Runge-Kutta discontinuous Galerkin methods for convection-dominated problems, *Journal of Scientific Computing*, vol.16, no.3, pp.173-261, 2001.

[17] D. J. L. Chen, J. C. Chang and C. H. Cheng, Higher order composition Runge-Kutta methods, *Tamkang Journal of Mathematics*, vol.39, no.3, pp.199-211, 2008.

[18] R. P. Melo and P. M. Tasinaffo, Uma metodologia de modelagem empírica utilizando o integrador neural de múltiplos passos do tipo Adams-Bashforth, *SBA. Sociedade Brasileira de Automática*, vol.21, no.5, pp.487-509, 2010.

[19] J. Peinado, J. Ibáñez, E. Arias and V. Hernández, Adams-Basforth and Adams-Moulton methods for solving differential Riccati equations, *Computers and Mathematics with Applications*, vol.60, pp.3032-3045, 2010.

[20] Z. Bin and Z. Bo, Comparison of different order Adams-Bashforth methods in an atmospheric general circulation model, *Acta Meteorologica Sinica*, vol.25, no.6, pp.754-764, 2011.

[21] E. Misirli and Y. Gurefe, Multiplicative Adams Bashforth-Moulton methods, *Numerical Algorithms*, vol.57, pp.425-439, 2011.

[22] Q. Ming, Y. Yang and Y. Fang, An optimized Runge-Kutta method for the numerical solution of the radial Schrödinger equation, *Mathematical Problems in Engineering*, vol.2012, pp.1-12, 2012.

[23] G. Polla, Comparing accuracy of differential equation results between Runge-Kutta Fehlberg methods and Adams-Moulton methods, *Applied Mathematical Sciences*, vol.7, no.103, pp.5115-5127, 2013.

[24] G. Cybenko, Continuous valued networks with two hidden layers are sufficient, *Technical Report*, Department of Computer Science, Tufts University, 1988.

[25] K. Hornik, M. Stinchcombe and H. White, Multilayer feedforward networks are universal approximators, *Neural Networks*, vol.2, no.5, pp.359-366, 1989.

[26] N. E. Cotter, The Stone-Weierstrass and its application to neural networks, *IEEE Trans. Neural Networks*, vol.1, no.4, pp.290-295, 1990.

[27] J. M. Zurada, *Introduction to Artificial Neural System*, West Pub. Co., St. Paul, MN, USA, 1992.

[28] J.-S. R. Jang, C.-T. Sun and E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Prentice-Hall, Inc., 1997.

[29] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd Edition, Prentice-Hall, Inc., New Jersey, 1999.

[30] J. T. Spooner, M. Maggiore, R. Ordónez and K. M. Passino, *Stable Adaptive Control and Estimation for Nonlinear Systems Neural and Fuzzy Approximator Techniques*, Wiley-Interscience, New York, 2002.

[31] L. Zadeh, Fuzzy sets, *Information and Control*, vol.8, pp.338-353, 1965.

[32] E. H. Mamdani, Application of fuzzy algorithms for control of simple dynamic plant, *Proc. of the IEE (Control and Science)*, vol.121, pp.298-316, 1974.

[33] E. H. Mamdani, Applications of fuzzy logic to approximate reasoning using linguistic synthesis, *IEEE Trans. Computers*, vol.c-26, no.12, pp.1182-1191, 1977.

[34] M. Sugeno, *Industrial Application of Fuzzy Control*, Elsevier Science Pub, New York, 1985.

[35] T. Takagi and M. Sugeno, Fuzzy identification of systems and its applications to modelling and control, *IEEE Trans. System, Man & Cybernetics*, vol.15, pp.116-132, 1985.

[36] L. Zadeh, Fuzzy logic, *IEEE Computer*, pp.83-92, 1988.

[37] B. Kosko, Fuzziness vs. probability, *International Journal of General Systems*, vol.17, no.2, pp.211-240, 1990.

[38] B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, Prentice-Hall, Inc., New Jersey, 1992.

[39] R. dos Santos Guimaraes, V. S. Junior and P. M. Tasinaffo, Multipolar-valued fuzzy sets to deal with the cognitive ambiguities, *International Journal of Innovative Computing, Information and Control*, vol.11, no.6, pp.1965-1985, 2015.

[40] R. S. Guimarães, V. Strafacci Júnior and P. M. Tasinaffo, Implementing fuzzy logic to simulate a process of inference on sensory stimuli of deaf people in an e-learning environment, *Computer Applications in Engineering Education*, vol.24, no.2, pp.320-330, 2016.

[41] B. E. Boser, I. M. Guyon and V. N. Vapnik, A training algorithm for optimal margin classifiers, *Proc. of the 5th Annual Workshop on Computational Learning Theory*, Pittsburgh, Pennsylvania/USA, pp.144-152, 1992.

[42] C. Cortes and V. Vapnik, Support-vector networks, *Machine Learning*, vol.20, pp.273-297, 1995.

[43] C. J. C. Burges, A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery*, vol.2, pp.121-167, 1998.

[44] J. C. Platt, Sequential minimal optimization: A fast algorithm for training support vector machines, *Technical Report MSR-TR-98-14*, p.21, 1998.

[45] S. Gazzah and N. B. Amara, Neural networks and support vector machines classifiers for writer identification using arabic script, *The International Arab Journal of Information Technology*, vol.5, no.1, 2008.

[46] Y. Tang, Deep learning using linear support vector machines, *International Conference on Machine Learning*, Atlanta, Georgia/USA, 2013.

[47] G. E. Hinton, S. Osindero and Y.-W. Teh, A fast learning algorithm for deep belief nets, *Neural Computation*, vol.18, pp.1527-1554, 2006.

[48] J. Lamos-Sweeney, *Deep Learning Using Genetic Algorithms*, Master Thesis, Rochester Institute of Technology, 2012.

[49] X.-W. Chen and X. Lin, Big data deep learning: Challenges and perspectives, *IEEE Access*, vol.2, pp.514-525, 2014.

[50] J. Schmidhuber, Deep learning in neural network: An overview, *Technical Report, IDSIA-03-14*, p.88, 2014.

[51] J. Heaton, *Artificial Intelligence for Humans Volume 3: Deep Learning and Neural Networks*, Heaton Research, Inc., St. Louis, MO, USA, 2015.

[52] S. Abrahams, D. Hafner, E. Erwitt and A. Scarpinelli, *Tensorflow for Machine Intelligence*, Bleeding Edge Press, Santa Rosa, 2016.

[53] G. Zaccone, *Getting Started with Tensorflow*, Packt Publishing Ltd., Birmingham, 2016.

[54] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, MIT Press, Cambridge, MA, 2017.

[55] K. J. Hunt, D. Sbarbaro, R. Zbikowski and P. J. Gawthrop, Neural networks for control systems – A survey, *Automatica*, vol.28, no.6, pp.1083-1112, 1992.

[56] A. Rios Neto, Dynamic systems numerical integrators in neural control schemes, *V Congresso Brasileiro de Redes Neurais*, Rio de Janeiro-RJ, Brazil, pp.85-88, 2001.

[57] A. Rios Neto and P. M. Tasinaffo, Modeling with ODE neural numerical integrators applied to the dynamics of an orbit transfer problem, *XI Colóquio Brasileiro de Dinâmica Orbital*, Viçosa/MG, Brazil, 2002.

[58] A. Rios Neto and P. M. Tasinaffo, Feedforward neural networks combined with a numerical integrator structure for dynamic systems modeling, *XVII COBEM – International Congress of Mechanical Engineering*, São Paulo/SP, Brazil, 2003.

[59] R. P. Melo, *Metodologia de Modelagem Empírica Utilizando Integradores Neurais de Múltiplos-Passos Aplicado a Sistemas Dinâmicos Não-Lineares*, Master Thesis, Instituto Tecnológico de Aeronáutica (ITA), São José dos Campos/SP, Brazil, 2008.

[60] P. M. Tasinaffo and A. Rios Neto, Modelagem de sistemas dinâmicos com integrador neural de Euler baseado em derivadas médias, *CBA 2004 – XV Congresso Brasileiro de Automática*, Gramado/RS, Brazil, 2004.

[61] P. M. Tasinaffo and A. Rios Neto, Mean derivatives based neural Euler integrator for nonlinear dynamic systems modeling, *Learning and Nonlinear Models*, vol.3, pp.98-109, 2005.

[62] R. P. Melo and P. M. Tasinaffo, Uma metodologia de modelagem empírica utilizando o integrador neural de Euler, *Congresso Brasileiro de Automática (CBA)*, Juiz de Fora/MG, Brazil, 2008.

[63] M. O. Figueiredo, *Modelagem de Sistemas Dinâmicos Não-lineares Utilizando Lógica Fuzzy e Algoritmos Genéticos*, Undergraduate Work, Instituto Tecnológico de Aeronáutica (ITA), São José dos Campos/SP, Brazil, 2012.

[64] P. M. Tasinaffo, R. S. Guimarães, L. A. V. Dias, V. S. Júnior and F. R. M. Cardoso, Discrete and exact general solution for nonlinear autonomous ordinary differential equations, *International Journal of Innovative Computing, Information and Control*, vol.12, no.5, pp.1703-1719, 2016.

[65] M. O. de Figueiredo, P. M. Tasinaffo and L. A. V. Dias, Modeling autonomous nonlinear dynamic systems using mean derivatives, fuzzy logic and genetic algorithms, *International Journal of Innovative Computing, Information and Control*, vol.12, no.5, pp.1721-1743, 2016.

[66] P. M. Tasinaffo, R. dos Santos Guimaraes, L. A. V. Dias and V. Strafacci Junior, Mean derivatives methodology by using Euler integrator improved to allow the variation in the size of integration step, *International Journal of Innovative Computing, Information and Control*, vol.12, no.6, pp.1881-1891, 2016.

[67] P. M. Tasinaffo, *Estruturas de Integração Neural Feedforward Testadas em Problemas de Controle Preditivo*, Ph.D. Thesis, INPE-10475-TDI/945, Instituto Nacional de Pesquisas Espaciais – INPE, São José dos Campos/SP, Brazil, 2003.

[68] M. Norgaard, O. Ravn, N. K. Poulsen and L. K. Hansen, *Neural Networks for Modelling and Control of Dynamic Systems*, Spring, London, 2000.

[69] A. Rios Neto and P. M. Tasinaffo, Neural numerical integrators in predictive control tested in an orbit transfer problem, *DINCON-2003 or II Congresso Temático de Dinâmica, Controle e Aplicações*, São José dos Campos/SP, Brazil, 2003.

[70] P. M. Tasinaffo and A. Rios Neto, *Integração Neural em uma Estrutura de Controle Preditivo*, Internal Report, INPE-11484-RPQ/778, Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos/SP, Brazil, 2004.

[71] P. M. Tasinaffo and A. Rios Neto, Predictive control with mean derivative based neural Euler integrator dynamic model, *Sociedade Brasileira de Automática (SBA)*, vol.18, no.1, pp.94-105, 2006.

[72] K. J. Hunt and D. Sbarbaro, Neural networks for nonlinear internal model control, *IEE Proceedings D – Control Theory and Applications*, vol.138, no.5, pp.431-438, 1991.

[73] S. Chen and S. A. Billings, Neural networks for nonlinear dynamic system modelling and identification, *Int. J. Control*, vol.56, no.2, pp.319-346, 1992.

[74] L. Euler, *Institutiones Calculi Integralis*, St. Petersburg, 1768.

[75] H. Poincaré, Mémoire sur les courbes définies par une équation différentielle (I), *Journal de Mathématiques Pures et Appliquées*, 3^e série, tome 7, pp.375-422, 1881.

[76] V. V. Nemytskii and V. V. Stepanov, *Qualitative Theory of Differential Equations*, Dover Publications, New York, 1989.

[77] W. McCulloch and W. Pitts, A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics*, vol.5, pp.115-133, 1943.