FEEDFORWARD NEURAL NETWORK COMBINED WITH NUMERICAL INTEGRATOR STRUCTURE FOR AN INVERTED PENDULUM MODELING AND SIMULATION.

Daniel van Berghem Motta
Divisão de Ciência da Computação – Instituto Tecnológico da Aeronáutica - ITA,
Pç. Marechal Eduardo Gomes, 50 – Vila das Acácias
CEP 12228-901 - São José dos Campos – SP – Brasil
dvbmotta@bol.com.br

Paloma Maria Silva Rocha
Departamento de Engenharia Elétrica – Universidade Estadual Paulista - UNESP
Av. Dr. Ariberto Pereira da Cunha, 333 – Pedregulho
paloma@feg.unesp.br

Paulo Marcelo Tasinaffo
Divisão de Ciência da Computação – Instituto Tecnológico da Aeronáutica - ITA,
Pç. Marechal Eduardo Gomes, 50 – Vila das Acácias
CEP 12228-901 - São José dos Campos – SP – Brasil
tasinaffo@ita.br

Abstract: This article aims to make the simulation of non-linearized inverted pendulum models using the set neural artificial network combined with high accuracy numerical integration. To achieve this goal we used a feedfoward neural network and the gradient and Kalman algorithms to train this net. Using these algorithms, we addressed the question of computational costs to train and simulate these models. These costs includes CPU time to process the neural network training and allocation of memory to dispose data. As the gradient training method in the neural network gives the derivative function of the system ( \( y' = f(y,t) \) ), we used numeric integrators to obtain the output of the system. So, a fourth order Runge-Kutta and an Adams-Bashforth integrator were matched with the output of the neural network ( \( y' \) ) so that we could have \( y \). Then, it was possible to analyze graphically the performance of these integration algorithms, compared with the true output of the physical dynamic system, given by the solution of the model’s analytic differential equation.

Key words: FeedForward Neural Network, Inverted Pendulum Model, Numerical Integration, non-linear system, Kalman Filter learning, gradient method.

1. Introduction

Neural networks have been used widely for identification of non-linear dynamic systems (Carrara, 1997; Chen et al., 1992 and Hunt et al., 1992). A feedforward neural network can be used in the structure of an ordinary differential equation numerical integrator algorithm to approximate and to replace the derivative function of the ordinary differential equations dynamic system mathematical model, a discrete model is obtained which has quite advantageous characteristics (Rios Neto, 2001 and Wang and Lin, 1998). With this approach, the difficulty with too many inputs in the training of the neural network is alleviated, since it is only necessary to learn an algebraic and static function, and the inputs are occurrences of the state and control variables in their envelope of variation.

It is possible, for example, to train a feedforward neural network in instantaneous derivative method, with was proposed by Wang and Lin in 1998, to represent non-linear dynamic systems. In this paper we used a feedfoward neural network and instantaneous derivative method to represent a non-linear and a linear inverted pendulum model, using Gradient and Kalman algorithms to train of the net. All this training used the software implemented by Tasinaffo in 2004. This software enabled us to vary several parameters of the net configuration, such as integration step, and so we could observe their influence in the MEQ (Average Squared Error) and how they differently affect linear and non-linear model training. Using this software, we addressed the question of computational costs to train and simulate these models. These costs includes CPU time to process the neural network training and location of memory to dispose data.

Though this paper is focused in testing the procedure in an application of interest, it is organized to guarantee a comprehensive presentation for the benefit of those not familiar with the tested procedure. In what follows, Section 2 presents the fundamentals of feedforward neural networks. Section 3 presents the concepts of numerical integrator of discrete forward model and neural numerical integrator. In section 4 the simulation and the results of the application to the problem of modeling the inverted pendulum are presented. In the last section some conclusions and suggestions to future researches are drawn.
2. Fundamentals: FeedForward Neural Networks.

It is not easy to find out a unique definition about what is an artificial neural network (Zurada, 2002). 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 (Braga et al, 2000) (Figure 1) 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(.)), 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.

\[ x^k_i = f^k_i \left( \sum_{j=1}^{n_{k-1}} w_{ij}^k x_{j}^{k-1} - b_i^k \right) \quad i = 1, 2, ..., n_k \]  

Figure 1 – The Perceptron Neuron.

The Feedforward neural network is made up of columns of Perceptrons (f’s) and so the output vector \( \mathbf{x}^k \) of a layer \( k \) is illustrate and simplification as in figure 2.

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

3. Numerical Integrator Discrete Forward Model and Neural Numerical Integrator

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

\[ \dot{x} = f(x, u) \]  

where \( x \in \mathbb{R}^n \) is the state vector; \( u \in \mathbb{R}^m \) is the control vector; \( f(x, u) \) is the derivative function, coming from physical laws governing the dynamic system. Consider now an ordinary differential equation (ODE) numerical integrator (Stoer et al, 1980) to get a discrete approximation of the system of Eq.(3):

\[ x = x(t + \Delta t) \equiv f_n(x(t), x(t - \Delta t), ..., x(t - n_0 \Delta t); u(t), ..., u(t - n_0 \Delta t); \Delta t) \]
where \( f_a(x(t), x(t-\Delta t),..., x(t-n_0\Delta t); u(t),..., u(t-n_0\Delta t); \Delta t) \) is the function that results from using evaluations of the derivative function of Eq. (2) in the numerical integrator algorithm; \( n_0 \) is related to the order of the approximation; if its value is greater than zero, one has the situation where a finite difference type of integrator is used (for example, an Adams-Bashforth method); if it is zero, a single step type of integrator is used (for example, a Runge-Kutta method); and \( \Delta t \), the step size, is assumed sufficiently small to assure \( u(t) \) constant along the discretization interval.

The numerical integrator in Eq.(3) can be used recursively as an approximate discrete predictive model of the dynamic system of Eq.(2) in internal model control schemes. The error in each step can be controlled by varying step size and or the order of numerical integrator; and the resulting numerical algorithm can be processed in parallel for each component of the state of the dynamic system. In the dynamic system of Eq. (2), the derivative function in the ODE mathematical model is an algebraic function that can be approximated by a feedforward neural network:

\[
\dot{x} = f(x,u) \cong \hat{f}(x,u,\hat{w})
\]  

(4)

where \( \hat{f}(x,u,\hat{w}) \) is to represent the neural network trained; and \( \hat{w} \) the neural network learned weights. Consider now this neural approximation of the derivative function in the structure of an ordinary differential equation (ODE) numerical integrator (Stoer et al, 1980) to get a discrete approximation of the system of Eq.(2):

\[
x(t+\Delta t) \cong \hat{f}_a(x(t), x(t-\Delta t),..., x(t-n_0\Delta t); u(t),..., u(t-n_0\Delta t); \Delta t; \hat{w})
\]  

(5)

The neural numerical integrator in Eq.(5) is an approximation of the ODE numerical integrator of Eq. (3) and thus an approximate discrete predictive model of the dynamic system of Eq.(2) which can be used as an internal model in control schemes.

4. Simulation and Results

As shown below in figure 3, our model consists of an inverted pendulum. We modeled it using the approaches of a non-linear model. The table 1 gives the nominal values used for modeling the body considerate.

![Inverted pendulum](image)

**Table 1 - Inverted pendulum parameters.**

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>mass of the cart</td>
</tr>
<tr>
<td>m</td>
<td>mass of the pendulum</td>
</tr>
<tr>
<td>b</td>
<td>friction of the cart</td>
</tr>
<tr>
<td>l</td>
<td>length to pendulum center of mass</td>
</tr>
<tr>
<td>I</td>
<td>inertia of the pendulum</td>
</tr>
<tr>
<td>F</td>
<td>force applied to the cart</td>
</tr>
<tr>
<td>x</td>
<td>cart position coordinate</td>
</tr>
<tr>
<td>( \theta )</td>
<td>pendulum angle from vertical</td>
</tr>
</tbody>
</table>

The non-linear model can be described by the following equations, where \( x \) and \( \theta \) are as above.

\[
(M + m)\ddot{x} + bx + ml\dot{\theta} \cos \theta - ml\dot{\theta}^2 \sin \theta = F
\]

\[
(I + ml^2)\ddot{\theta} + mgl \sin \theta = -ml\ddot{x} \cos \theta
\]  

(6)
Our objective with this paper is divided in two tasks: train a Neural Network (NN) to learn the differential equation in term of state variables that fits the non-linear model of the inverted pendulum and after simulate the time response of the model to different initial conditions. To train the NN, we adjusted the number of training patterns given to the net and the learning rate so that we could have a model good enough and minimize the costs in terms of processing. Besides, we varied:

- $x$ from -5 to 5 meters;
- $\frac{dx}{dt}$ from -14 to 14 m/s;
- $\theta$ from $-\pi/12$ to $\pi/12$;
- $\frac{d(\theta)}{dt}$: from -1 to 1 rad/s
- $F$ (external excitation of the system) from -10 to 10 Newtons

We attempt to reach MEQ in the training of the NN about $10^{-7}$. Unfortunately, the minimum MEQ we got in non-linear case was $1.2812 \times 10^{-5}$. We used 3500 patterns points generated randomly within the state-space described above. In addition, we used two algorithms to train the net: Kalman Filter and Gradient. The functioning of this algorithms aren’t in the scope of our work. After setting the main parameters of the simulation, as the step of integration, initial condition of state variables e time interval of simulation, we could get some results, as shown below.

As shown in Figure 4, the time interval of simulation was reduced from 0 (as the initial time of simulation) until 1 (end point of simulation). It’s easy to see that this simulation was much more successful than the first one. It’s due to the capability of approximation of the non-linear model. Thus, $[0,1]$ is a reliable interval to this model. The propagation MEQ was drastically decreased with this change in simulation interval.

Figure 5 – Graphics showing the evolution in time interval $[0,3]$ of state variables of the non-linear model.
In Figure 5, it’s clear that we have reached a good result with the non-linear approach, with the new interval \([0, 3]\) second. This confirms the importance of the non-linearity in the inverted pendulum model that itself can be applied besides the linear confimation. It’s very important to notice that the step of integration considerably affects the precision of the neural network estimation. Nevertheless, decreasing the step of integration means an increase in the costs of computer processing time.

5. Conclusions and suggestions to future researches

This article reached the our aims, namely: the limitations of a mathematical model has to represent a physical one; the implementations and simulations of a non-linear general model; the potentialities and limitations of Feedforward neural network with a single internal layer to represent and learn a non-linear model and the influence of NN parameters and settings in the its performance and throughput.

So, we hope we have contributed to the understanding of FeedForward NN and its implementation and as suggestion it would be very interesting to develop this problem using a RBF Neural Network architecture. In this case, the RBF is a good estimator of linear parameters, and possibly the results of the linear model estimation would be better. Afterwards, it would be interesting as well as to implement a FeedForward NN with more than one internal layer to see how it would behave in terms of non linear estimation and computational costs.

6. References