USE OF NEURAL NETWORKS IN SOUNDING ROCKET TRAJECTORY RECONSTITUTION

ALEXANDRE N. BARBOSA

Subdivision of Flight Dynamics, Division of Space Systems, Institute of Aeronautics and Space (IAE)
Praça Marechal Eduardo Gomes, 50, Vila das Acácias, 12228-904 São José dos Campos – SP, Brazil
E-mail: n.barbosa@iae.cta.br

JOSÉ DEMISIO S. DA SILVA, RAFAEL SANTOS

Laboratory for Computing and Applied Mathematics, National Institute for Space Research (INPE)
Av. dos Astronautas, 1758, Jardim da Granja, 12227-010 São José dos Campos – SP, Brazil
E-mails: demisio@lac.inpe.br, rafael.santos@lac.inpe.br

Abstract— The neural networks, which are a model of how the brain works, are well known as universal approximators. This article takes advantage of such characteristic on presenting an alternative way for reconstituting a sounding rocket trajectory, instead of using the conventional approach, which spends too much effort. The basic idea is concerned to the use of a neural network to map the difference between nominal and actual flight curves, solving drop-outs, which are occasioned by malfunctions and atmospheric conditions.

Keywords— neural networks, sounding rocket, trajectory reconstitution.

1 Introduction

A rocket launch mission takes into account three main aspects. The first one is concerned to the payload experiments. The second one is the rocket that meets the mission requirements. The third one is the launch infra-structure that encompasses the systems to track the rocket, such as the telemetry and radar systems. In Brazil, the priority is the radar. There are at least two radar systems, one for proximity and another one for accuracy. Both provide the instantaneous rocket position and velocity. They should carry out without halts, but tracking interruptions could occur due to atmospheric conditions, malfunctions and so on. Such problem happened with the first flight of the Brazilian rocket VSB-30, during its launch operation in Alcantara Launch Center (CLA), in Brazil, October 2004.

When the tracking vanishes simultaneously in both radars, they are guided by the information of the nominal trajectory, in order to retry the tracking.

After the flight, the post-flight analysis starts. The Subdivision of Flight Dynamics of the Institute of Aeronautics and Space, in Brazil, is responsible for studying the trajectory of Brazilian rockets and performing its post-flight analysis. In order to carry out such analysis, computational applications are employed. Among those applications, it is the Rocket Simulation >> ROSI << for trajectory calculation [6] [3]. If the actual trajectory is not quite tracked by the radars, it is performed its reconstitution by using the ROSI and updating its input data with actual conditions.

This paper treats with a typical problem of signal loss, or drop-out. At the time that the radars fails to track the rocket, the nominal characteristics of the flight are not realized in the actual trajectory. Therefore, the ROSI is employed to reconstitute such characteristics. But, the complexity of the reconstitution using the ROSI is high and increases with the presence of noise and when burning and atmosphere phases are taken into account in the drop-out segment of the actual trajectory [7]. Despite the fact that the ROSI takes too much effort in order to reconstitute a trajectory, it is the most accurate approach because it uses analytical models.

2 The Proposed Approach

Searching by an approach with low complexity, this paper introduces the use of neural networks in order to reconstitute the trajectory of a sounding rocket, instead of using the ROSI. The basic idea is to map the difference from both nominal and actual trajectory curves by using a multilayer perceptron, and reconstitute the occluded nominal characteristics, that are not realized in the actual curve (Fig. 1).

The actual curve reconstitution is obtained by an interpolation of the difference between the nominal and actual curves in the drop-out segment. Such interpolation should result in a good generalization of the difference between the curves in that segment. This is the goal to be achieved in the multilayer perceptron training.

The topology of the neural network, the size of the training set and the complexity of the physical problem influence the capability of the neural network for generalizing [4]. This is a reason for training different topologies and sampling data for training and testing in different manners.

In order to set different topologies through the arguments of the neural network algorithm, the neural units are identified sequentially, in an array, from 1 to q (Fig. 2). In this manner, the multilayer perceptron is defined by changing the following arguments: the

number of hidden layers and units per layer. This way, the best one among the topologies, which has the best generalization, can be chosen.

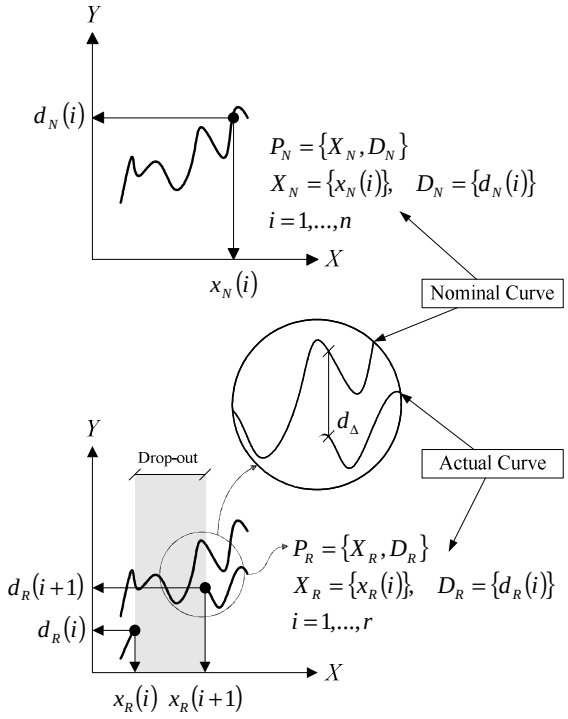

Figure 1. The Definition of the Nominal and Actual Curves.

In order to minimize the mean-squared error in the output of the unit 1 by adapting the synaptic weights of the multilayer perceptron, the backpropagation algorithm is employed for training the neural network. Such algorithm was developed by David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams, in 1986. The term is an abbreviation for "backwards propagation of errors" [4]. The error is calculated with respect to the difference from both nominal and actual trajectory curves.

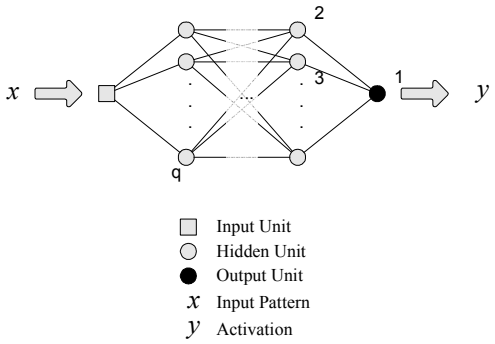

Figure 2. The Neural Network Topology Definition.

The algorithm, which follows, describes in detail the preprocessing into the training steps of the multilayer perceptron that employs a linear interpolation to estimate the difference between nominal and actual curves (not including the drop-out segment).

Such linear interpolation is employed because both curves are given in a discrete set of points. Even though it is possible to require their synchronization, this study takes into account that they are not supposed to match in the same abscissa.

Because of the difference between the curves is addressed to the same abscissa, it is necessary to estimate such difference by an interpolation.

Step 0: The steps from 1 to 10 are repeated until the mean-squared error (5) has started to converge asymptotically or the number of epochs be greater than what was previously defined.

Step 1: For each nominal input pattern $x_N(i)$, $i = 1,...,n$, the steps from 2 to 8 are carried out.

Step 2: The nominal input $x_N(i)$ is presented to the neural network that gives the result of its activation *y*.

Step 3: The actual input data $x_R(j^-)$ and $x_R(j^+)$, where $j^+ = j^- + 1$, are found so that $x_R(j^-) \le x_N(i) \le x_R(j^+)$.

If $x_R(j^-)$ or $x_R(j^+)$ does not exist, the value -1 is assigned to the respective index as a flag to remind that at the step 6. For instance, it might come as the following:

$$x_N(i) \le x_R(1) \Rightarrow j^- = -1 \wedge j^+ = 1, \text{ or}$$
$$x_R(r) \le x_N(i) \Rightarrow j^- = r \wedge j^+ = -1$$

Step 4: The indices i^- and i^+ with respect to the nominal input data, which are respectively closest from the lowest and highest x_R, are found.

$$\begin{aligned} i^- &= \arg\min_k \left\{ \left\| x_R(1) - x_N(k) \right\| \right\} \\ i^+ &= \arg\min_k \left\{ \left\| x_R(r) - x_N(k) \right\| \right\} \end{aligned} \quad k = 1,...,n \qquad (1)$$

Step 5: The nominal output data with respect to the indices that were found in step 4 are assigned to d_N^- and d_N^+.

$$d_N^- = d_N(i^-) \text{ and } d_N^+ = d_N(i^+) \qquad (2)$$

Step 6: The difference from nominal and actual curve d_Δ is calculated, using a linear interpolation.

P_Δ is the set of patterns that estimates the difference from nominal and actual curves (Fig. 3).

If $j^\pm \neq -1$ is true, the difference $d_\Delta(i)$ is calculated by a linear interpolation (3).

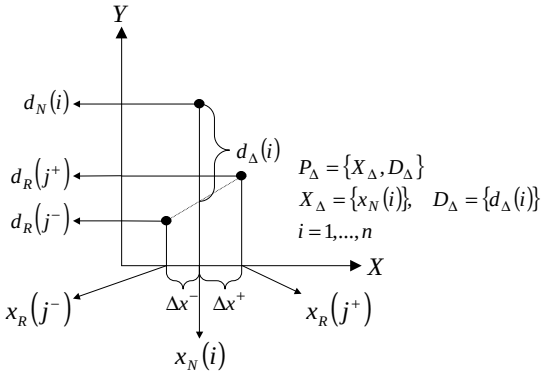

Figure 3. The Definition of the Difference Between Nominal and Actual Curves.

$$\left.\begin{aligned}\Delta x^- &= x_N(i) - x_R\left(j^-\right)\\ \Delta x^+ &= x_R\left(j^+\right) - x_N(i)\end{aligned}\right\} \Rightarrow$$
$$d_\Delta = d_N(i) - \frac{\Delta x^- \cdot d_R\left(j^+\right) + \Delta x^+ \cdot d_R\left(j^-\right)}{x_R\left(j^+\right) - x_R\left(j^-\right)} \tag{3}$$

On the other hand, if $j^+ = -1$ or $j^- = -1$ is true, d_Δ is the difference between the outputs of the closest nominal and actual inputs as follows:

$$d_\Delta = \begin{cases} d_N^- - d_R\left(j^+\right), & j^+ \neq -1 \wedge j^- = -1 \\ d_N^+ - d_R\left(j^-\right), & j^+ = -1 \wedge j^- \neq -1 \end{cases} \tag{4}$$

Step 7: The error in the output of the neural network is calculated.

Step 8: The backpropagation algorithm is carried out to adapt the synaptic weights of the neural network.

Step 9: The mean-squared error is calculated (5).

$$E = \frac{1}{2 \cdot n} \cdot \sum_{k=1}^{n} e_k^2 \tag{5}$$

Where e_k is the error in the output unit with respect to the output pattern of index k.

Step 10: The terminal condition is verified.

The presented algorithm describes the preprocessing. But, its implementation should take into account the computational efficiency. This way, the estimated output patterns P_Δ for training the multilayer perceptron should be obtained before carrying out the training algorithm by finding and keeping the indices $j^-(i)$ and $j^+(i)$, $i = 1,...,n$, as well as d_N^- and d_N^+.

Besides, it is emphasized that the preprocessing does not interpolate the drop-out segment. This is done by the neural network.

3 Application

In order to demonstrate the proposed approach and verify for which conditions it diverges, a drop-out has been provoked on a quite covered flight, such as the second one of the VSB-30 (Fig. 4), so that it is possible to calculate the error with respect to the omitted part of the actual trajectory.

In addition, an enhanced approach is briefly mentioned, in the ending of this topic, for further studies.

As a first case study, a single curve has been taken from the trajectory of the VSB-30, which is relevant at all for post-flight analysis, the total velocity of the rocket as a function of time (Fig. 5).

The reconstitution of the total velocity data is preferable to the position data because the second one is obtained by a numerical integration of the first one. It follows that a numerical integration is more desirable than a numerical differentiation [1], which is what would occur in the opposite chosen. Nevertheless, the difference of signal quality between the position and velocity data can define what data should be used in the reconstitution.

Figure 4. VSB-30.

The post-flight analysis uses the position and velocity data in order to provide the following magnitudes: altitude, downrange, flight azimuth and elevation, Mach number, dynamic pressure, splashdown impacts and so on.

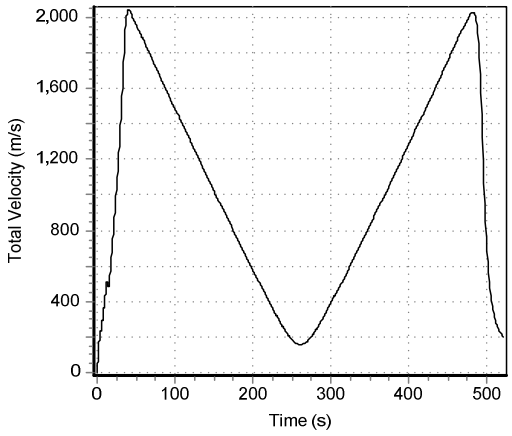

Figure 5. The Nominal Total Velocity.

The second flight of the VSB-30 has spent 550 seconds. In the current case study, a drop-out on the total velocity actual data, which is slightly different from the nominal data, has been provoked from 100 to 300 seconds of the flight. In fact, such simulation is more than what is expected if it is taken into account the radar system capability to recover the tracking upon 200 seconds of drop-out. This simulates a condition that is itself an upper limitation for the radar system (Fig. 7).

The neural network training has been successfully carried out (Fig. 8). It was used a multilayer perceptron with three hidden layers and eight units per layer (Fig. 6).

In order to have good results in such application, a good generalization is important. This is the reason of trying network topologies with more than a single hidden layer. Although the universal approximation theorem states that a single hidden layer is sufficient for a multilayer perceptron to compute an approximation, the theorem does not say that is sufficient to obtain a good generalization [4].

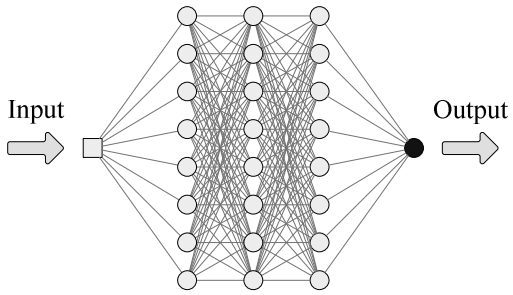

Figure 6. The Neural Network Topology.

In order to allow the network to learn more quickly when there are plateaus in the error surface the backpropagation algorithm has been used with momentum (6). The rate of learning η was 0.1 and the momentum α was 0.5. Synaptic weights w were initialized from a uniform distribution whose mean was zero and whose interval of variance was (-1,1).

$$w = w_{old} + \alpha \cdot \Delta w_{old} + \Delta w, \text{ where } \Delta w = -\eta \cdot \frac{\partial E}{\partial w} \quad (6)$$

The activation function $\varphi(\cdot)$ associated with the neural units was the hyperbolic tangent function (7), with $a = 1$ and $b = 2$.

$$\varphi(v) = a \cdot \tanh(b \cdot v),\ (a,b) > 0 \quad (7)$$

v: activation potential of the neuron

In addition, 351 patterns were used for training and 350 for its validation.

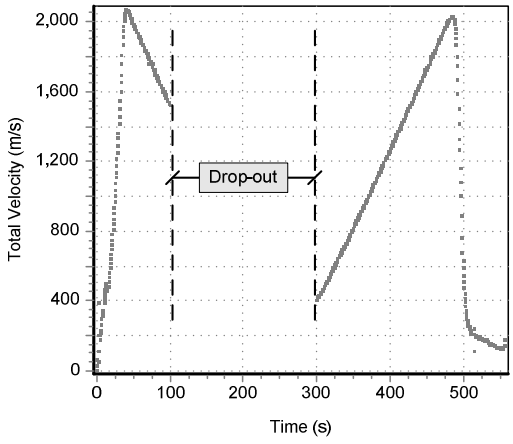

Figure 7. The Actual Curve with a Drop-out.

In order to obtain a good generalization, the neural network topology and the training set size have been considered, but the complexity of the physical process has not because it is addressed to the sinuosities of the curve that represents the physical process. In some cases, a complexity reduction may be done by a logarithmic representation of the data as it is seen in a log-log plot. But, here, such approach has not worked.

The relative error, which was achieved after 13347 epochs of training, was about 3% (with respect to the actual curve, Fig. 9). It is figured out that the ROSI has better results if compared with the proposed approach. It would be pretty good to compare both methods, but it still remains as a suggestion for further studies. Despite this, the relative error, which was achieved by the neural network, is acceptable for a preliminary post-flight analysis.

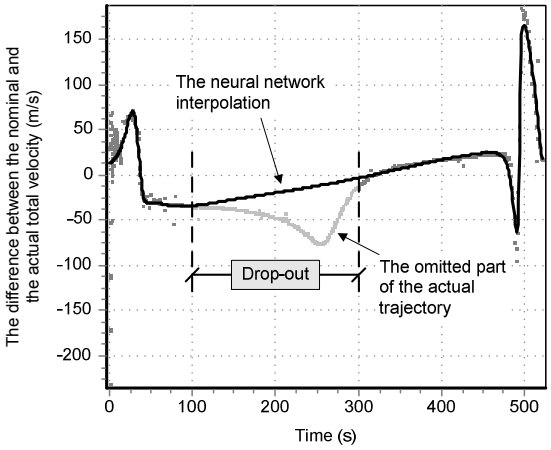

Figure 8. The Neural Network Interpolation.

One of the characteristics of this problem is that the difference from nominal and actual curves may present some sinuosities (Fig. 8). This way, the reconstitution depends on how the peakedness of those sinuosities is, because the neural network is not supposed to map them (Fig. 9).

In a second case study, the actual curve has been multiplied by 0.6, a dispersion of 5% has been provoked as a simulation of noise, and the same drop-out has been provoked so that the transformed curve could be understood as an anomalous flight (Fig. 10).

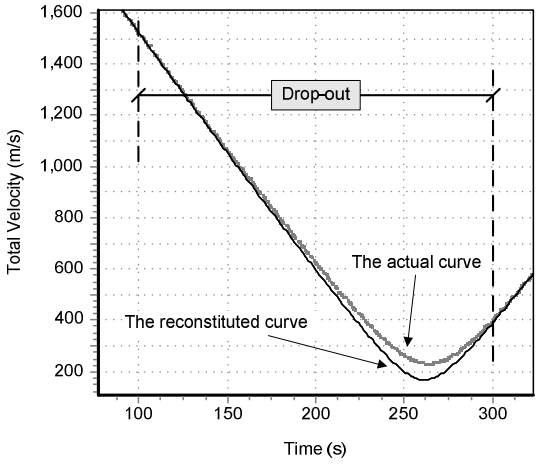

Figure 9. The Reconstituted Curve.

The relative error was about 5% (Fig. 13). If the same problem is solved by interpolating linearly the drop-out segment, instead of using the neural network, the relative error increases to 13%.

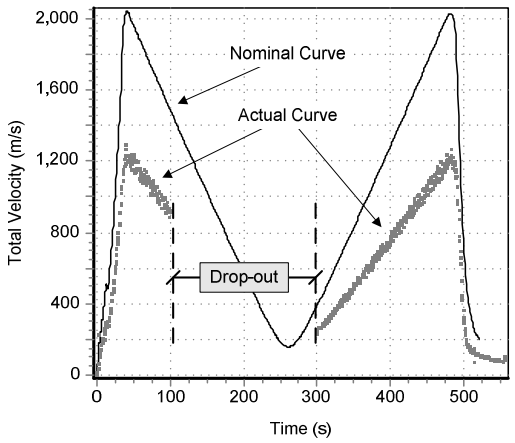

Figure 10. An Anomalous Actual Curve.

In order to decide when the training should be actually stopped, it was used the cross validation method (Fig. 11).

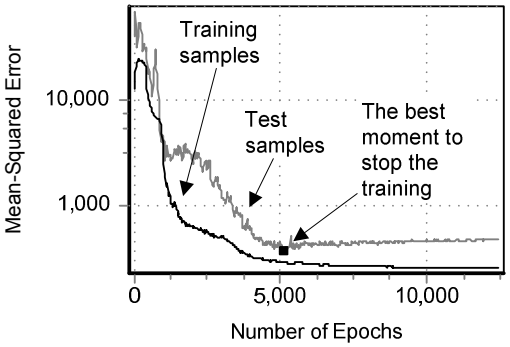

Figure 11. The Cross Validation.

This approach diverges if the difference between the nominal and actual curves is strongly nonlinear in the drop-out segment (Fig. 12).

Such situation could occur if the actual trajectory is highly anomalous. In this case, the radar system is not supposed to recover the tracking and the proposed approach should not be used. In spite of this, it has been observed that the neural network interpolation is more suitable than a linear interpolation on the drop-out segment, and, in this case, about half of the omitted part of the actual trajectory has been correctly recovered (Fig. 13).

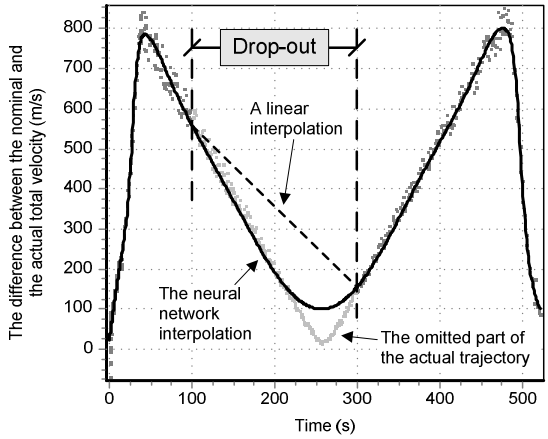

Figure 12. The Neural Network versus a Linear Interpolation.

Looking forward to enhancing the proposed approach, one could try another neural network. Instead of using a multilayer perceptron to estimate the difference between the nominal and actual trajectories, a spatiotemporal neural network could be wanted to carry out it. Such approach is probably more suitable for reconstituting a rocket trajectory.

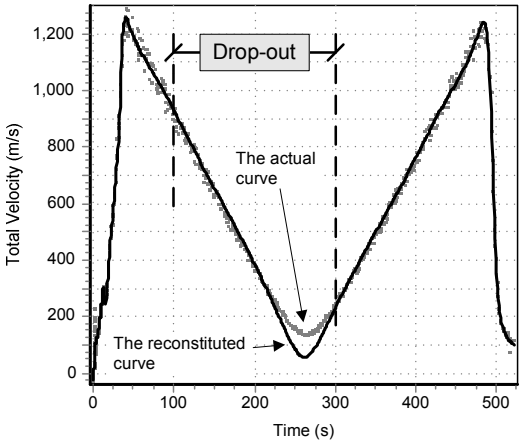

Figure 13. The Reconstituted Anomalous Actual Curve.

A spatiotemporal neural network deals with inputs and outputs that are explicit functions of time [5], such as a rocket trajectory that can be viewed as a time-parameterized trajectory or path in R^3. This kind of neural network might reconstitute an actual trajectory that is more than slightly different from the nominal.

The nominal and actual trajectories increase their differences when the rocket is in a burning and atmosphere's phase. The deviation of the trajectory of the rocket may be caused by steady wind, wind-gusts, atmospheric turbulence, but also by fin misalignment and production inaccuracies [2]. With all of this, the rocket trajectory is not quite predictable. This is why the trajectory reconstitution is complex.

4 Conclusion

Despite the fact that this study is in a beginning and the result is not quite satisfactory yet, the proposed approach could be an alternative way for reconstituting the rocket trajectory. It takes advantage of its simple algebraic structure when compared to the ROSI software, which uses complicated analytical models. It is preferable to the ROSI when reconstituting an actual trajectory with noise. In addition, it could be extended to others categories of problems that are quite similar to the problem of signal loss.

References

Conte, S. D. and Boor, C. (1981). Elementary Numerical Analysis – An Algorithmic Approach, 3rd ed., Singapore: McGraw-Hill, ch. 7.

Cornelisse, J. W., Schöyer, H. F. R. and Wakker, K. F. (1979) Rocket Propulsion and Spaceflight Dynamics. Northern Ireland: Pitman, ch. 14.

Gomes, R. M. “Trajectory Calculation using the ROSI Program”, unpublished.

Haykin, S. (1999). Neural Networks: A Comprehensive Foundation, 2nd ed., Prentice Hall.

Hecht-Nielsen, R. (1990). Neurocomputing, Addison-Wesley.

Kramer, H. J., Craubner, A. and Ziegltrum, W. (1976). ROSI Rocket Simulation, DFVLR TN 12/76.

Louis, J. E. “The Reconstitution of the Trajectory of the Brazilian Rocket VSB-30 V01,” unpublished.