International Journal of Engineering and Technical Research (IJETR)
ISSN: 2321-0869 (O) 2454-4698 (P) Volume-9, Issue-12, December 2019

River Flux Prediction Via RBF Neural Networks And NARMAX Methodology

V. P. R. Arruda, P. M. Tasinaffo, A. M. Cunha, L. A. V. Dias

Abstract— **Use of Radial Basis Function Neural Networks to predict the time series of river flux data was approached via NARMAX methodology. By modifying the network architecture in search of the best performance characteristics, it was observed that the problem of time series analysis has chaotic behavior, which cannot be fully accounted for by use of such technology, suggesting the need for more robust machine learning techniques.**

***Index Terms*— RBF neural network, NARMAX methodology, time series prediction, dynamic systems**

I. INTRODUCTION

Radial Basis Function (RBF) networks are characterized for their use of a unique hidden layer whose neurons' activation functions belong to the class of "radial basis functions". Such a layer augments the dimensionality of input data when it maps input to hidden space, and the network's output is a weighted sum of the hidden layer activations.

Further, the NARMAX methodology refers to a technique for presenting input data to a neural network – be it RBF or not. In the context of a network modelling a dynamical system whose output is $y(t)$ - where t stands for time, NARMAX consists in presenting to the model a history $\{y(t-1), y(t-2), ..., y(t-t_d)\}$ – where t_d is the maximum delay – of past system outputs, supposing this is enough for learning its dynamics.

In the following, we present the theory underlying RBF and NARMAX, followed by the analysis proposed in this work.

Highlight a section that you want to designate with a certain style, and then select the appropriate name on the style menu. The style will adjust your fonts and line spacing. **Do not change the font sizes or line spacing to squeeze more text into a limited number of pages.** Use italics for emphasis; do not underline.

A. *RBF Networks*

The architecture known as RBF consists of two layers: the first one has m hidden units, such that the layer's activation functions are $\{G_1, \ldots, G_m\}$; the second is composed of l neurons, each of which generates a network output, such that each output $\hat{y}_i$ computed from input x is determined by $\hat{y}_i = \sum_{j=1}^{m} w_{ij} G_j(x)$, where $\{w_{i1}, \ldots, w_{im}\}$ are neural weights associated to the i -th output neuron.

Thus, hidden units, by increasing the dimensionality of input data, increase the chance of them becoming linearly separable [1]. In this work, we use a "bias" term w_{i0} in the output, such that we have instead $\hat{y}_i = w_{i0} + \sum_{j=1}^{m} w_{ij} G_j(x)$. This is equivalent to considering an extra hidden unit with constant unitary activation function. We also restrict the analysis to Gaussian activation functions, those with the format:

$$G^{\sigma, x_c}(x) = \exp\left(\frac{\|x - x_c\|^2}{2\sigma^2}\right) \quad (1)$$

Where x_c and σ are respectively termed centroid and standard deviation, both being parameters of the function.

There are two main types of RBF networks: *regularization networks* and *generalized networks.* The former possess as many hidden units as there are training samples, and there is rigorous mathematical support to guarantee the existence of solution to the interpolation problem which they solve [2], however – owing to their large degree of freedom in terms of neural parameters [3] – they are prone to overfit, whence there is considerable body of literature describing how to avoid this tendency [4], [5]. The latter, on the other hand, align to the general principle of the Occam Razor [6], in that their centroid count is usually less than the amount of training samples by several orders of magnitude, thus reducing the likelihood of overfit. From here on, we take "RBF network" to mean "generalized RBF network".

Several training techniques are known [7], [8]. One of the simplest is *K-means* [9], [10] for centroid coordinate computation, and *pseudoinverse method* for neural weight computation.

K-means attemps to define hidden unit centroids $\{t_1, \ldots, t_m\}$ according to the geometry of samples in the training set $I = \{x_1, \ldots, x_N\}$, where $N = |I|$ is the training set size and each x_i is a training sample. To this effect, over a partition $P = \{P_1, \ldots, P_m\}$ of I , we define the "intra-group sum of squares":

$$L(P) = \sum_{i=1}^{m} \sum_{t \in P_i} \left(t - \frac{1}{|P_i|} \sum_{u \in P_i} u \right)^2 \quad (2)$$

Centroids are computed by finding P which minimizes $L(P)$, then defining each t_i as the arithmetical average of all elements belonging to P_i . Since this minimization problem is NP-hard, usually heuristics are applied, which naturally may lead to local optima.

V. P. R. Arruda, Instituto Tecnológico de Aeronáutica, São José dos Campos, São Paulo, Brasil
P. M. Tasinaffo, Instituto Tecnológico de Aeronáutica, São José dos Campos, São Paulo, Brasil
A. M. Cunha, Instituto Tecnológico de Aeronáutica, São José dos Campos, São Paulo, Brasil
L. A. V. Dias, Instituto Tecnológico de Aeronáutica, São José dos Campos, São Paulo, Brasil

Meanwhile, the *pseudoinverse method* is employed to compute output-layer weights after centroids and standard-deviations have been fixed. In such a context, and considering a scalar-output network (generalizing to multidimensional output is immediate), the method defines a cost function:

$$C(w_1,\ldots,w_m)=\sum_{i=1}^{N}\left(y_i-\sum_{j=1}^{m}w_i G^{\sigma_j,t_j}(x_i)\right)^2 \tag{3}$$

Where y_i is the expected output for each input x_i. Minimizing C leads to:

$$w=\left(G^T G\right)^{-1}G^T y \tag{4}$$

Where we assume the following notations:

$$G=\begin{bmatrix} G^{\sigma_1,t_1}(x_1) & G^{\sigma_2,t_2}(x_1) & \ldots & G^{\sigma_m,t_m}(x_1) \\ G^{\sigma_1,t_1}(x_2) & G^{\sigma_2,t_2}(x_2) & \ldots & G^{\sigma_m,t_m}(x_2) \\ \vdots & \vdots & & \vdots \\ G^{\sigma_1,t_1}(x_N) & G^{\sigma_2,t_2}(x_N) & \ldots & G^{\sigma_m,t_m}(x_N) \end{bmatrix} \tag{5}$$

$$y=\left[y_1,y_2,\ldots,y_N\right]^T \tag{6}$$

$$w=\left[w_1,w_2,\ldots,w_m\right]^T \tag{7}$$

B. *NARMAX methodology*

For a variety of natural phenomena, physical laws are known which can describe and predict their evolution quantitatively. When this is the case, physics offers a mathematical model through which the dynamical system may be studied. However, there are complex systems for which either no structure is known or it is too complex to be completely treated computationally. This situation motivates the use of empirical formulations in an attempt to analyse the problem, one of them being NARMAX [11].

In NARMAX formulation, a discrete system with output y and input u is described by [12]:

$$y(t)=F\left(y(t-1),\cdots,y(t-n_y),u(t-1),\cdots,u(t-n_u)\right) \tag{8}$$

Where F is an arbitrary non-linear function, n_y is the maximum output delay and n_u is the maximum input delay. Thus, (8) is an approximation to the system under scrutiny.

C. *Problem proposal*

This work employs and RBF network coupled with NARMAX methodology to predict the flux of Ganges river (MG/Brasil) given the history of its flux in the past. The solution is implemented in MATLAB, and the combination of RBF and NARMAX has precedents in the literature [13].

In possession of a database registering monthly flux of Camargos and Furnas hydroelectric plants (built along the river) during a 82 year interval, we intended to have the network learn the river's dynamics. To this effect, we architected the input, hidden and output layers of the model, defining – for instance – the maximum delay in presenting flux data to the network, the quantity of hidden units, and the training algorithm to be used – provided more than one is known.

Varying the architecture choice, we compared all obtained results to one another, analyzing how each factor impacted network performance.

II. METHODOLOGY

Fig. 1 illustrates the network architecture. A single network processes both power plants' fluxes, each with 2 delayed inputs. Input vector $\mathbf{u}$ is admitted to each of the $m+1$ hidden units, the first of which is constant and equal to 1 (bias) while the remaining ones are Gaussian centered in $\mathbf{t_1},\ldots,\mathbf{t_m}$ with a common standard-deviation σ. Finally, the bidimensional output is $\hat{\mathbf{y}}(\mathbf{u})=\mathbf{w}_0+\sum_{i=1}^{m}\mathbf{w}_i G_i(\mathbf{u})$ - where $\mathbf{w}_i$ is a bidimensional neural weight and G_i is the radial basis function centered on $\mathbf{t_i}$ - and represents the next predicted flux for each power plant. We adopt the notation $\mathbf{W}=\left[\mathbf{w_0}\cdots\mathbf{w_m}\right]^T$ ($m+1$ rows and 2 column) for the neural weights matrix.

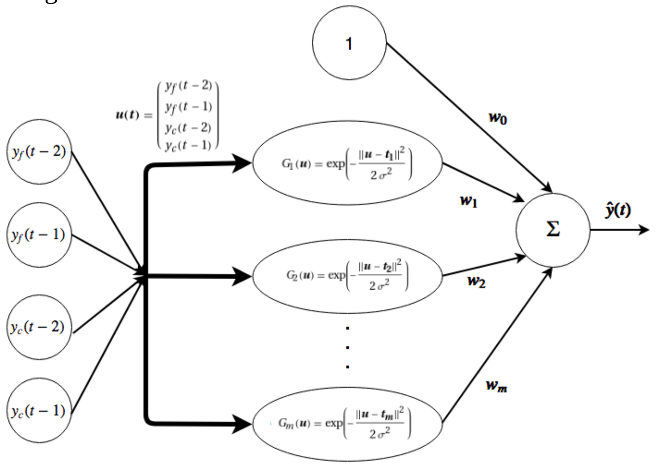
1
$\boldsymbol{u}(t)=\begin{pmatrix}y_f(t-2)\\y_f(t-1)\\y_c(t-2)\\y_c(t-1)\end{pmatrix}$
$y_f(t-2)$
$y_f(t-1)$
$y_c(t-2)$
$y_c(t-1)$
$G_1(\boldsymbol{u})=\exp\left(-\frac{\|\boldsymbol{u}-\boldsymbol{t}_1\|^2}{2\sigma^2}\right)$
$G_2(\boldsymbol{u})=\exp\left(-\frac{\|\boldsymbol{u}-\boldsymbol{t}_2\|^2}{2\sigma^2}\right)$
$G_m(\boldsymbol{u})=\exp\left(-\frac{\|\boldsymbol{u}-\boldsymbol{t}_m\|^2}{2\sigma^2}\right)$
$\boldsymbol{w_0}$
$\boldsymbol{w_1}$
$\boldsymbol{w_2}$
$\boldsymbol{w_m}$
Σ
$\hat{\boldsymbol{y}}(\boldsymbol{t})$

Figure 1. RBF network. Variables y_c and y_f *refer to Camargos and Furnas plants, and boldface symbols are vectors. Parameters* $\mathbf{t}_i$, *σ and* $\mathbf{w}_i$ *must be learned by the network.*

A. *Training and test sets*

Training the network means deciding on values for $\mathbf{t_1},\ldots,\mathbf{t_m}$, σ and $\mathbf{w_0},\ldots\mathbf{w_m}$. To this effect training samples were extracted from aforementioned database, in the following format: there were 984 flux measurements for each power plant, from $\phi_1(f)$ to $\phi_{984}(f)$ (Furnas plant) and from $\phi_1(c)$ to $\phi_{984}(c)$ (Camargos plant); for each triplet $\{\phi_i,\phi_{i+1},\phi_{i+2}\}$, we built a sample such that the input to the network is $\mathbf{u}_i=\{\phi_i(f),\phi_{i+1}(f),\phi_i(c),\phi_{i+1}(c)\}$ and the expected output is $\mathbf{y}_i=\{\phi_{i+2}(f),\phi_{i+2}(c)\}$; the 10 last such samples were reserved for post-training evaluation of network performance (test set), while the remaining $N=982-10=972$ constituted the effective training set. We adopt the notation $\mathbf{Y}=\left[\mathbf{y}_1,\ldots\mathbf{y}_N\right]$ (2 rows and N columns) for the matrix of expected outputs.

B. *Training algorithm*

The first pre-processing procedure was normalizing input and output data. This was carried out dimension-by-dimension, with the usual process of subtracting the mean and dividing by standard-deviation.

To determine the centroids $\mathbf{t_i}$ of the hidden units, we used *K-means* [10].

Once the centroids had been computed, we used a heuristic approach to estimate σ [10]:

$$\sigma = \frac{d_{\max}}{\sqrt{2m}} \tag{9}$$

Where $d_{\max}$ is the maximum centroid-to-centroid distance among all centroid pairs. Finally, the weights $\mathbf{W}$ were determined with the pseudoinverse method [10], according to which:

$$\mathbf{W} = \mathbf{G}^{+}\mathbf{Y}^{T} \tag{10}$$

Where $\mathbf{G}^{+}$ stands for the pseudoinverse of $N \times m+1$ matrix $\mathbf{G}$ given by:

$$\mathbf{G} = \begin{bmatrix} 1 & G_1(\mathbf{u}_1) & \cdots & G_m(\mathbf{u}_1) \\ 1 & G_1(\mathbf{u}_2) & \cdots & G_m(\mathbf{u}_2) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & G_1(\mathbf{u}_N) & \cdots & G_m(\mathbf{u}_N) \end{bmatrix} \tag{11}$$

C. Architecture alternatives

For comparison purposes, all m in the set $\{3,5,10,15,20,50\}$ were utilized so as to evidence the influence of centroid quantity over network generalizability. Further, we searched for the centroid count (restricted to inveral from 2 to 50) which would minimize the squared error E over the test set, where E is defined by:

$$E = \sum_{i} \left\| \hat{\mathbf{y}}(\mathbf{u}_i) - \mathbf{y}_i \right\|^2 \tag{12}$$

Where i loops through the test set, $\hat{\mathbf{y}}(\mathbf{u}_i)$ is the output of the network under input $\mathbf{u}_i$, and $\mathbf{y}_i$ is the expected output for the same input.

Finally, we experimented varying how many delayed inputs the network receives: instead of 2 delayed measurements for each power plant (as in Fig. 1), we tried 3 or more measurements.

We note that the quantity of hidden layers was not a theme for study because it is already known that single-hidden-layer RBF networks are universal function approximators [14].

III. RESULTS AND ANALYSIS

First we present performances obtained by varying the quantity of centroids in the hidder layer in the set $\{3,5,10,15,20,50\}$. In what follows, we analyze the effects of augmenting the quantity of delayed inputs available to the network.

A. Network with 3 centroids

In Fig. 2 and Fig. 4, we observe that the network was not capable of capturing the higher frequencies in the time series of both plants, besides being clearly delayed in contrast to the plants' dynamics.

The first mentioned issue refers to the amplitudes of oscillation in the plants's dynamics and in the network's dynamics: the former exhibits peaks that the latter cannot capture in the training set. Further, network outputs in the test set approximate moving averages of the expected outputs (Fig. 3 and Fig. 5).

On the other hand, the second mentioned issue alludes to the time delay between network output peaks and power plant peaks, visible in the training set.

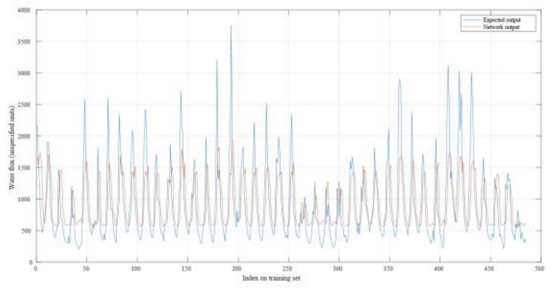

Figure 2. 3-centroid network performance on training set for Furnas plant

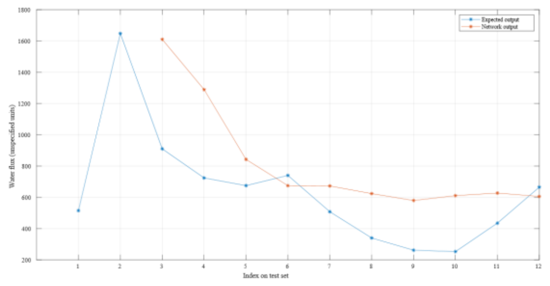

Figure 3. 3-centroid network performance on test set for Furnas plant

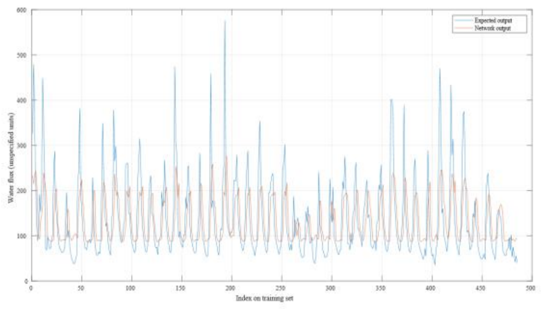

Figure 4. 3-centroid network performance on training set for Camargos plant

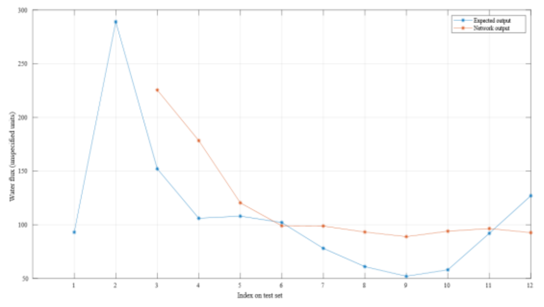

Figure 5. 3-centroid network performance on test set for Camargos plant

B. Network with 5 centroids

We note in Fig. 6 and Fig. 8 a reduction in time delay between expected and predicted peaks. Furthermore, higher oscilation frequencies were captured by the network, which can be perceived by the concave outline between indexes 6 and 12 in the test set.

However, inflexion points in the expected outputs, such as as index 6 of Furnas plant (test set) were still not captured by the model.

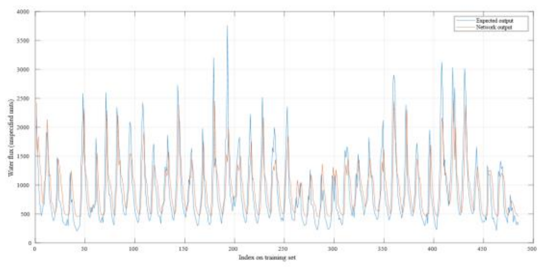

Figure 6. 5-centroid network performance on training set for Furnas plant

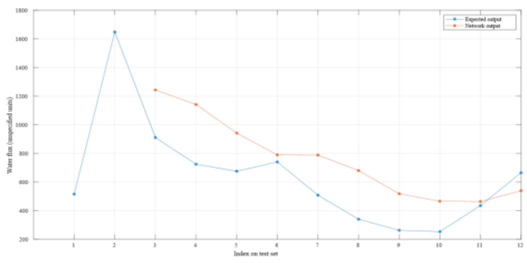

Figure 7. 5-centroid network performance on test set for Furnas plant

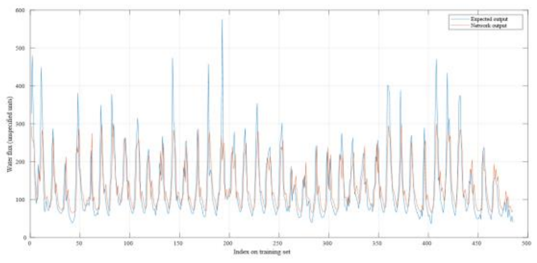

Figure 8. 5-centroid network performance on training set for Camargos plant

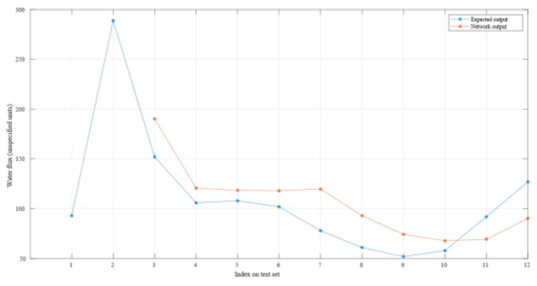

Figure 9. 5-centroid network performance on test set for Camargos plant

C. *Network with 10 centroids*

We again notice a reduction in temporal delay between expected and predicted fluxes. Further, the network could capture another aspect of the chaotic dynamics: it emulated the inflexion noticeable in index 6 of the testing set for Furnas plant, even though with temporal delay. Quantitatively there was progress as well: the squared error in the test set was reduced in contrast to previous models.

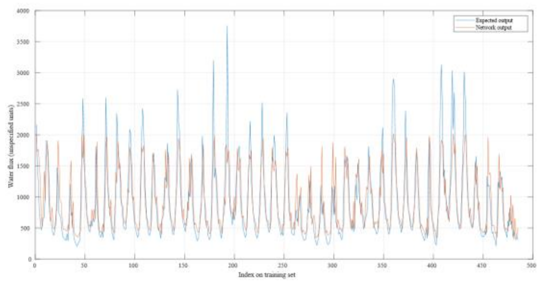

Figure 10. 10-centroid network performance on training set for Furnas plant

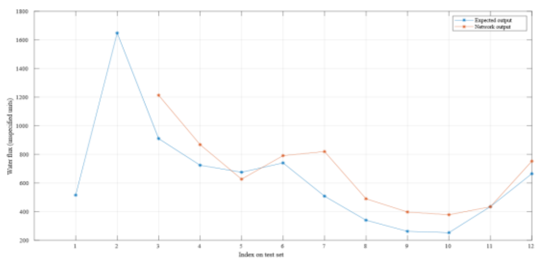

Figure 11. 10-centroid network performance on test set for Furnas plant

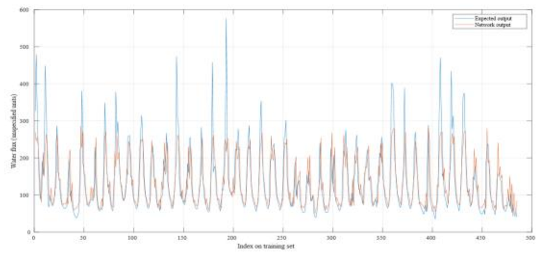

Figure 12. 10-centroid network performance on training set for Camargos plant

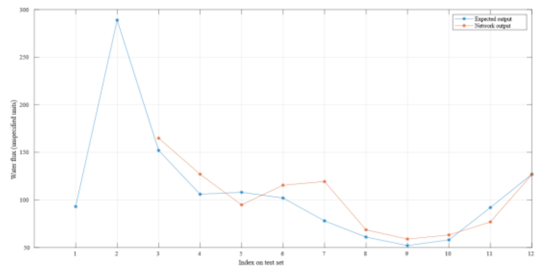

Figure 13. 10-centroid network performance on test set for Camargos plant

D. *Network with larger centroid counts*

The differences between using 15, 20 or 50 centroids are no longer noticeable, but we list remaining data for completeness (Fig. 14 to Fig. 25).

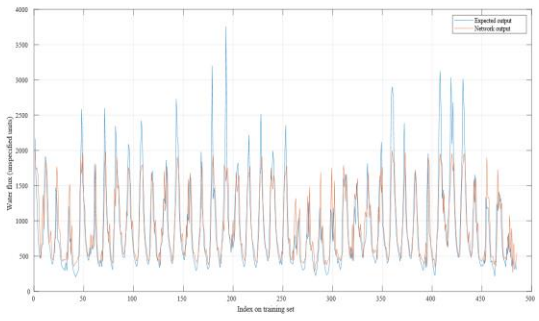

Figure 14. 15-centroid network performance on training set for Furnas plant

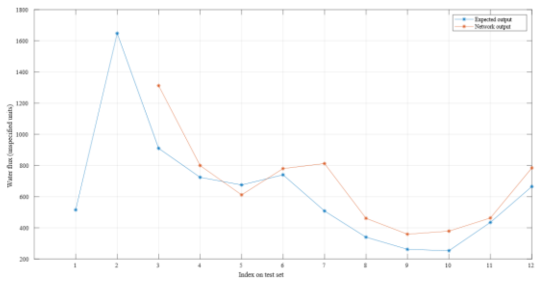

Figure 15. 15-centroid network performance on test set for Furnas plant

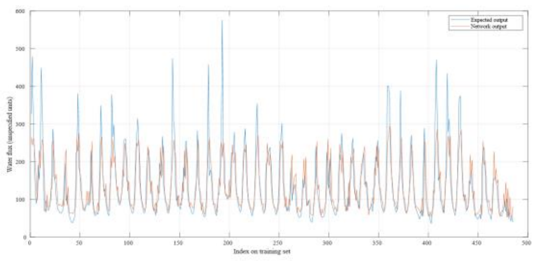

Figure 16. 15-centroid network performance on training set for Camargos plant

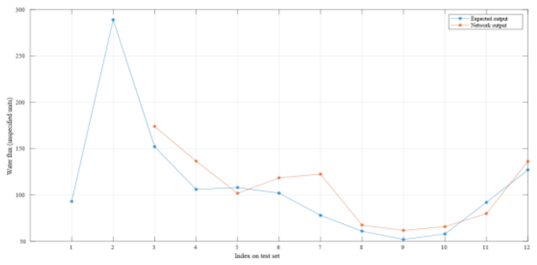

Figure 17. 15-centroid network performance on test set for Camargos plant

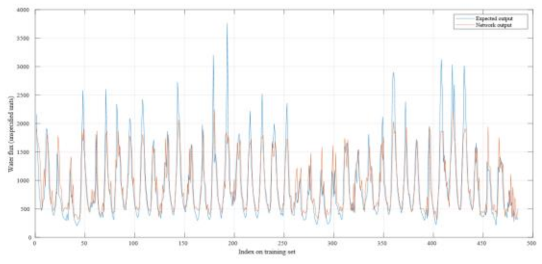

Figure 18. 20-centroid network performance on training set for Furnas plant

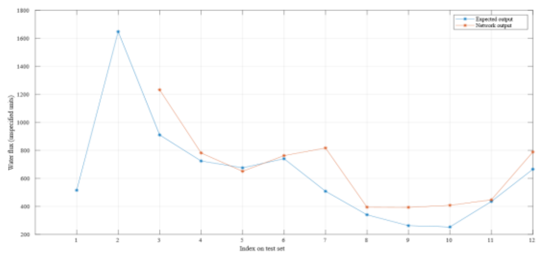

Figure 19. 20-centroid network performance on test set for Furnas plant

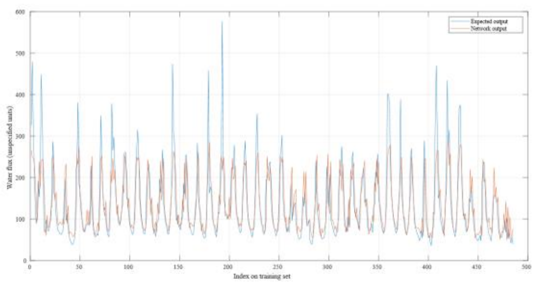

Figure 20. 20-centroid network performance on training set for Camargos plant

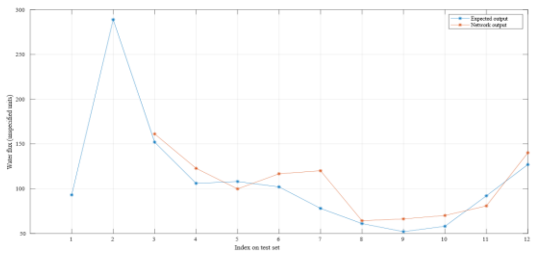

Figure 21. 20-centroid network performance on test set for Camargos plant

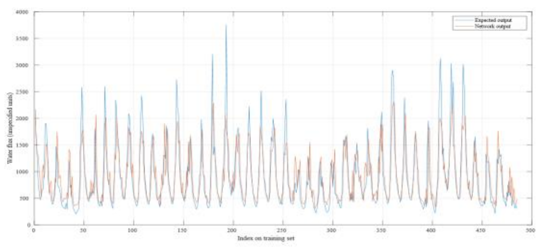

Figure 22. 50-centroid network performance on training set for Furnas plant

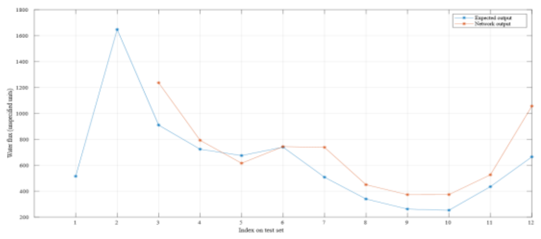

Figure 23. 50-centroid network performance on test set for Furnas plant

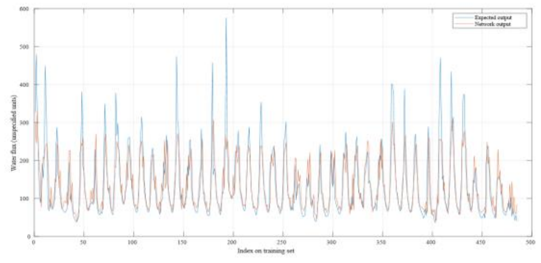

Figure 24. 50-centroid network performance on training set for Camargos plant

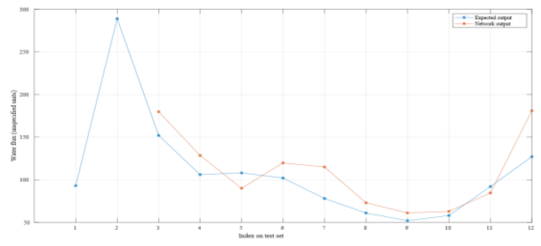

Figure 25. 50-centroid network performance on test set for Camargos plant

E. *Effects of centroid count*

To evaluate the effect of centroid count m on network performance, we employed the metric of squared error (12) on the test set for comparison. Using this metric, we searched the interval from 2 to 50 in an attempt to find the optimum value for the parameter, with results summarized by Fig. 26:

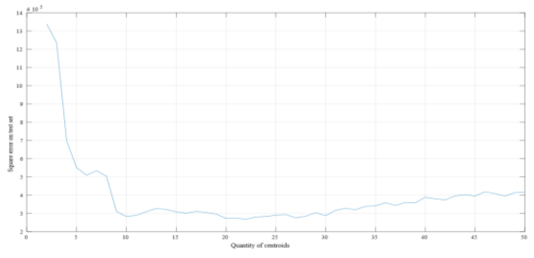

Figure 26. Squared error on test set for various centroid counts

Above 10, m has few influence on network generalizability.

F. *Networks with more delayed inputs*

We investigated another dimension of architectural variation: the quantity of network inputs. In other words, we changed the structure illustrated by Fig. 1 to increase the maximum delay of inputs to the network.

While varying t_d (the maximum delay) and m (centroid count) simultaneously, we searched for the network with least mean squared error (squared error divided by quantity of samples) in the test set. The results are illustrated by Fig. 27.

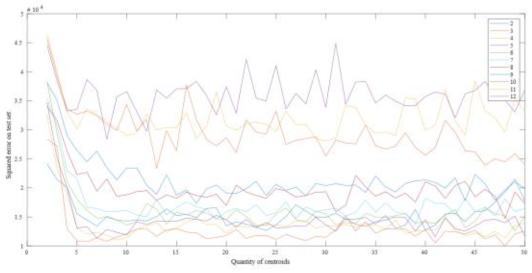

Figure 27. Mean squared error on test set as a function of the quantity of delayed inputs for each power plant

We observe that increasing maximum delay from 2 to 4 increased performance, but the opposite happened with further increases of delay. Due to the noisy data, however, we conjecture that the apparent evolution brought about by varying t_d does not generalize, i.e. it was verified for the test et but does not equate to a real improvement in network generalizability. The results presented here can be considered an alternative method to the one described in [15], where Multilayer Perceptrons are employed for the problem of flux prediction.

IV. Conclusion

By using time series data about monthly flux measurements of two hydroelectric power plants in Grande river, in a 82 year interval, we employed RBF networks coupled with NARMAX methodology to predict future flux data. The network performance indicated that the model cannot replicate the highest frequencies of oscillation that characterize the real data, even though it can approximate the data in periods of low oscillation.

Furthermore, the divergence between network and expected outputs seemed non-reducible through refining the model's architecture. Therefore it is plausible that the investigated dynamical system requires more than just its past immediately-visible behavior for successful forecasting.

References

[1] Cover, T.M. Geometrical and Statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*. vol EC-14(3), pp. 326–334. 1965.

[2] Micchelli, Charles A. *Interpolation of scattered data: distance matrices and conditionally positive definite functions*. In *Approximation theory and spline functions*, pp. 143-145. Springer, Dordrecht, 1984.

[3] LeCunn, Y. et al. Generalization and network design strategies. *Connectionism in perspective*. pp. 143-155. Citeseer, 1989.

[4] Bishop, C.M. Training with Noise is equivalent to Tikhonov Regularization. *Neural Computation.* vol 7, pp 108-116. 1995.

[5] Bishop, C. Improving the Generalization Properties of Radial Basis Function Neural Networks. *Neural Computation*. vol 3(4), pp. 579-588. 1991.

[6] Russel, S.J. et al. *Artificial Intelligence: A Modern Approach*. Pearson; 2 ed. p. 652. ISBN: 0130803022. 2002.

[7] Orr, M.J.L. Introduction to radial basis function networks. *Technical Report, Center for Cognitive Science, University of Edinburgh*. 1996.

[8] Rippa, S. An algorithm for selecting a good value for the parameter c in radial basis function interpolation. *Advances in Computational Mathematics*. vol 11(2-3), pp. 193-210. 1999.

[9] Hartigan, J.A; Wong, M.A. Algorithm AS 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*. vol 28(1), pp. 100-108. 1979.

[10] Haykin, S. *Neural Networks and Learning Machines*. Pearson; 3 ed. ISBN: 9789332570313. 2016.

[11] Billings, S.A; Daniel, C. Identification of NARMAX and related models. *Research Report – University of Sheffield Deparment of Automatic*

[12] Chen, S; Billings, S.A. Representations of non-linear systems: the NARMAX model. *International Journal of Control*. vol 49(3), pp. 1013-1032. 1989.

[13] Chen, S. et al. Practical identification of NARMAX models using radial basis functions. *International Journal of Control*. vol 52(6), pp. 1327-1350. Taylor & Francis, 1990.

[14] Park, J; Sandberg, I.W. Universal approximation using radial-basis-function networks. *Neural computation.* vol 3(2), pp. 246-257. 1991.

[15] D. B. De Lima, M. D. C. E Lima and R. M. Salgado. "An Empirical Analysis of MLP Neural Networks Applied to Streamflow Forecasting". IEEE Latin America Transactions, Vol. 9, No. 3, June 2011.

Vitor Pimenta dos Reis Arruda finishes undergraduate course on Computer Engineering in 2020 by Technological Institute of Aeronautics (ITA). Acted on the design and implementation of three-tier web applications and of proof-of-concept distributed systems.

Paulo Marcelo Tasinaffo Graduated in Mechanical Engineering from the Federal University of Itajubá (UNIFEI, 1996), Master's Degree in Mechanical Engineering from the Federal University of Itajubá (UNIFEI, 1998) and PhD in Space Engineering and Technology by the National Institute for Space Research (INPE, 2003). He is currently a full professor at the Technological Institute of Aeronautics (ITA). Has experience in Aerospace Engineering and Computing, focusing on Artificial Neural Networks, acting on the following subjects: modeling of nonlinear dynamic systems, computational mathematics, artificial intelligence, intelligent agents, expert systems, neural control systems, computation evolutionary and stochastic processes.

Adilson Marques da Cunha received the B.Sc. degree in pilot training from the Brazilian Air Force Academy - AFA, in 1970, the B.Sc. degree in business administration from the Brasilia Uni_ed Learning Center (CEUB), in 1977, the M.Sc. degree in information systems from the United States Air Force Institute of Technology - AFIT, USA, in 1984, and the D.Sc. degree in information systems from George Washington University (GWU), USA, in 1987. He has worked with Information Systems, Software Engineering, and Artificial Intelligence Applications. He is currently a Full Professor with the Computer Science Department, Brazilian Aeronautics Institute of Technology [Instituto Tecnológico de Aeronáutica (ITA)].

Luiz Alberto Vieira Dias holds a degree in Electrical Engineer from the Pontifical Catholic University of Rio de Janeiro (1966), a Masters in Space and Atmosphere Science from the National Institute for Space Research (1968), a Masters in Space Sciences - Rice University (1971) and a PhD in PhD. Space Physics and Astronomy - Rice University (1973). He is currently Collaborating Professor at the Technological Institute of Aeronautics (ITA).