Implementation of Neural Networks in Foundation Fieldbus Environment Using Standard Function Blocks

Diego R. C. Silva Adrão D. Dória Neto Jorge D. Melo Luiz Affonso Guedes

Department of Computer and Automation Engeneering
Federal University of Rio Grande do Norte
Campus de Lagoa Nova, Natal,RN, Brazil
{diego,adriao,jdmelo,affonso}@dca.ufrn.br

Abstract

This paper presents an implementation approach of artificial neural networks in industrial network environment, through the use of function blocks standardized by Fieldbus Foundation (FF). This enables the implementation of a wide range of applications that involve this mathematical tool, such as intelligent control, failure detection, etc in standard FF system. For validation propose, some examples are presented from of real experiments.

1. Introduction

Industrial automation had evolved greatly since decade of 1960. It first appeared with the pneumatic technology, and then incorporated analogical electronics and the PLCs (Programmable Logic Controllers), with the popularization of microelectronics. From PLCs, physical projects had evolved until leading to industrial automation networks. Finally, impelled by advances in digital communications, we have come to what is called today the FCSs (Field Controller Systems)[3]. Among several existing field control systems, Foundation Fieldbus (FF)[3], can be highlighted for having a foundation that has as one of its main concerns the specification of a open standard, that makes possible the interoperability of equipment from different vendors. A second interesting important point is the organization of application layer in function blocks, which facilitates development of new functionalities once those can be encapsulated, being defined only a interface for input and output signals.

There is a number of function blocks that is already standardized by FF, such as controllers and mathematical algorithms[1], but some advanced functionalities is still missing in those systems, such as fuzzy logic, neural networks, etc. Some computational intelligence technique could be useful in control and supervision of most diverse processes, as in intelligent control[7], indirect measure[5] fault detection[8]. As these "advanced blocks" are not yet standardized by the foundation, there is a chance that vendors implement these funcionalities on its own will, not guaranteeing its interoperability.

The goal of this work is to make available a tool of artificial neural networks in Foundation Fieldbus environment, by using only function blocks standardized by the foundation. This makes it a universal solution, once it can be used in environments with devices from different vendors, and enables a wide range of new applications in processes in control level.

The remaining of this paper is organized as follows: in Section 2, the FF standard is presented along with its particularities, then the function block concept is presented and a few examples are cited. In Section 3 we describe fundamental neural networks concepts and the solutions used for its implementations on Foundation Fieldbus networks, using standardized function blocks. In Section 4 it is described the testing environment and the results obtained from two experiments. Finally, in Section 5 there is a brief conclusion followed by indication for future works.

2. Foundation Fieldbus Networks

Fieldbus Foundation[2] is an independent organization which purpose is to develop and maintain a internationally uniform standard for field networks of automation processes, the Foundation Fieldbus. One of its main characteristics is the organization of the application layer in function blocks, which are open and completely specified, allowing to reach one of the main goals of foundation, the interoperability.

These blocks are divided in three great categories: resource blocks, transducer blocks and funcional blocks. The last ones are responsible for the handling of signal provided by resource blocks, after being treated by the transducer block.

Each functional block can be seen as a piece of software with well-defined interfaces for input and output, making it transparent for the user-level the internal implementation details. This encapsulation allows decen-

tralized and independent development of new function blocks, therefore making it easier the task of enhancing the network with new functionalities.

Other positive point of the funcional-blocks-based architecture is the possibility of linking blocks in order to combine the processing algorithms among themselves and with the input signals of a device. This allows complex tasks and greatly increases the number of possible applications.

Among standardized function blocks, we can cite the following: AI (Analog Input), AO (Analog Output), DI (Discrete Input), DO (Discrete Output), PID (Proportional Integral Derivative), ARTHM (Arithmetic), CHAR (Characterizer), etc. The last two were used in this work and will be further detailed. Figure 1 shows the internal scheme of the standard FF arithmetic block. In it we can see the existence of a sub-block called "Algorithm Type" which can assume different configurations and is chosen by the user.

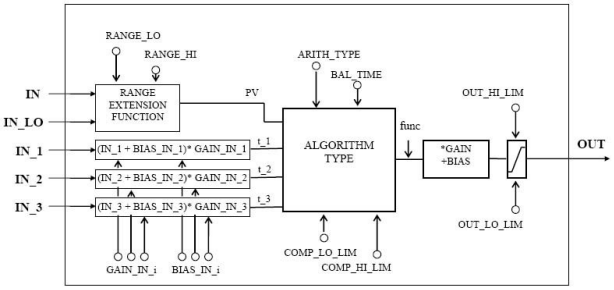

Figure 1. Internal scheme of standard FF arithmetic block.

The various functions taken by this block are listes in table 1 and are mapped through the first parameter ARITH_TYPE.

N.	Algorithm Description
1	Flow Compensation, linear
2	Flow Compensation, root square
3	Flow Compensation, approximation
4	BTU Flow
5	Traditional Multiple Divisor
6	Average
7	Traditional Adder
8	Fourth-order Polynomial
9	HTG Level Compensator

Table 1. Arithmetic block functionalities.

Figure 2 illustrates the internal scheme of characterizer function block.

As its own name suggests, this block has the function of characterize functions, in other words, with the limitation of 21 points, it tries to build the wished function by interpolating them. These points, chosen by the user, are in form of parameter and are filled with the help of a configuration software.

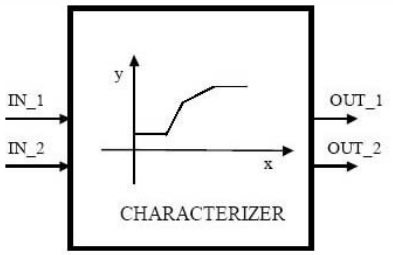

Figure 2. Internal scheme of characterizer block.

3. Neural Networks and its Implementation in FF Environment

Neural networks are formed by cells called neurons, which are responsible for the information processing in the brain. Artificial neural networks, in turn, are formed by sets of artificial neurons, which consist of a mathematical model a natural neuron. Figure 3 shows this model[6].

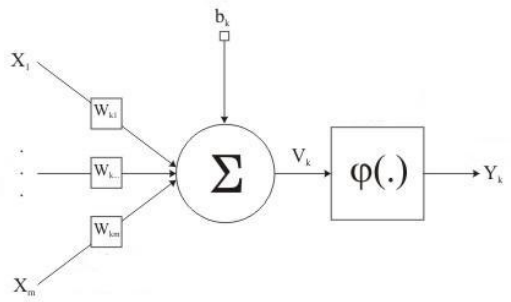

Figure 3. Mathematical Model of an Artificial Neuron.

Neurons are connected through link known as synapsys, or synaptyc gains. To each synaptic gain a weight is associated, which generates a transformation of the input signal. All input signals are added and a bias is applied in the end in order to increase or decrease the effective input for the activation function, which in turn is the responsible for the network to reach the linearity or not.

There is two main types of neural network structures: direct-feed networks and recurrent networks[4]. A direct-feed network represents a function of its current input, in other words, there is no internal states besides the gains itself. In a recurrent network there is feedback loops, in other words, network outputs are used to feed its own inputs. This means that the activation levels of the network makes a dynamic system that can exhibit stable, oscillatory or even chaotic behavior.

3.1 Implementation of Artificial Neural Networks in FF Environment

The presented solution involves two function blocks: arithmetic and characterizer, shown in figures 1 and 2 respectively. They must be configured and linked in a way that the set behaves as a artificial neuron, as can be seen in figure 3. In arithmetic block, the Algorithm Type parameter must be chosen as Traditional Adder and fill the gains of inputs according to off-line training performed.

In characterizer block, we can choose up to twenty points to simulate the activation function. In this work a sigmoid function has been chosen, shown in figure 4.

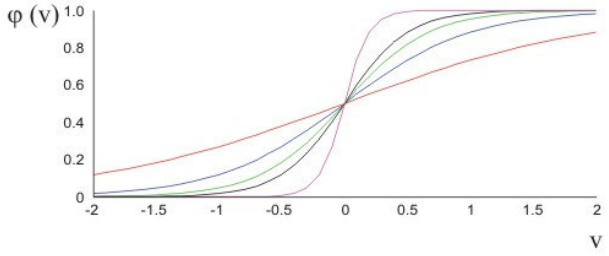

Figure 4. Function chosen for the activation function.

The selection of points must have as goal the minimization of the error between original sigmoid function and the approximated function. Genetic Algorithms was chosen for this selection. To measure the quality of the approximation, the used technique relies on a evaluation function, defined here as the quadratic mean error between original function and the function formed by the interconnection of chosen points. The set of resulting chosen points is shown in figure 5.

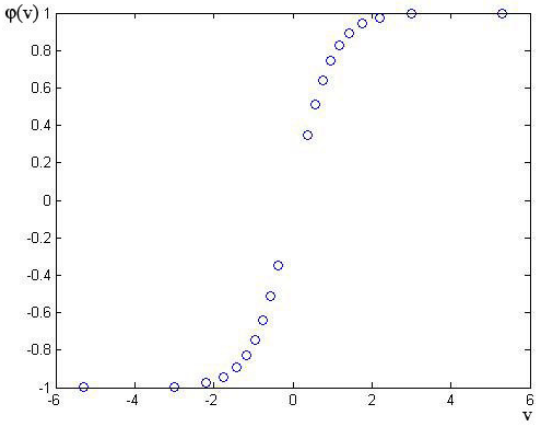

Figure 5. Points chosen to form the evaluation function.

In order to visualize the quality of approximation, the functions were overlapped, as can be seen in figure 6.

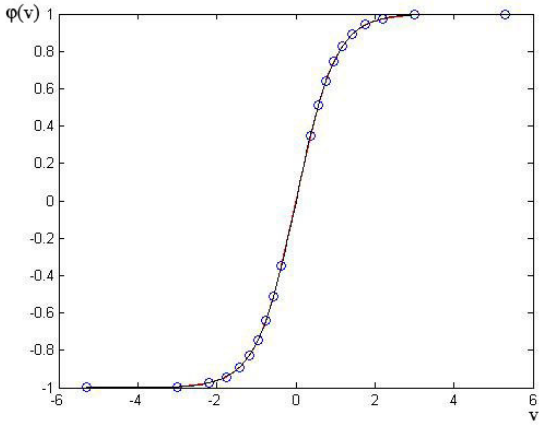

Figure 6. Functions overlapped.

By linking the output of arithmetic function block to the input of signal characterizer function block, configured as described above, we have a artificial neuron in the FF environment. And by the linking of these neurons it is build the neural networks, as will be shown in following section.

4. Test Environment and Experimental Results

The test environment used consist in a industrial Foundation Fieldbus network with six field devices and a bridge, which makes the interconnection among the industrial network and the regular personal computers connected through an Ethernet network, where the configuration softwares are installed. Figure 7 represents the architecture of the didactic network used.

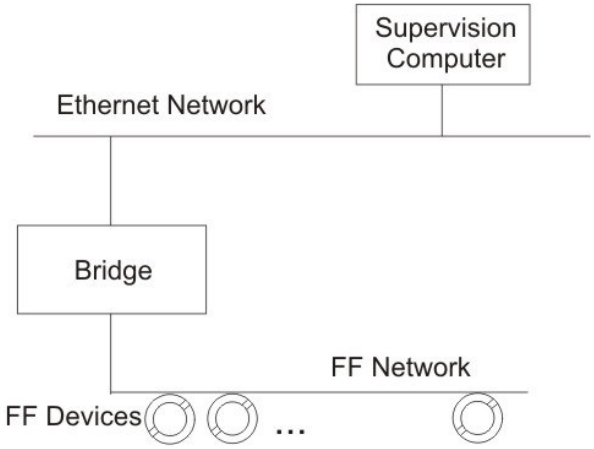

Figure 7. FF network architecture.

We can choose any device to implement the procedure described on previous section. Here we show the implementation in two examples.

4.1 Example 1

In this first example we have trained a neural network with a 1-3-1 architecture, as shown in figure 8, to learn the input-output training set illustrated in figure 9.

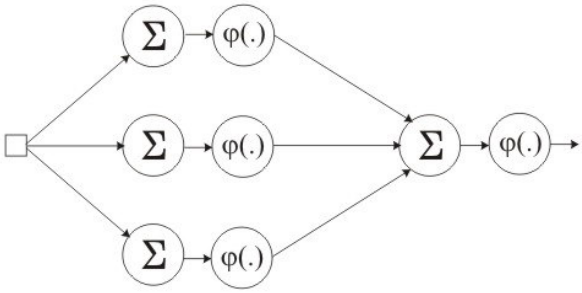

Figure 8. Neural network implemented for example 1.

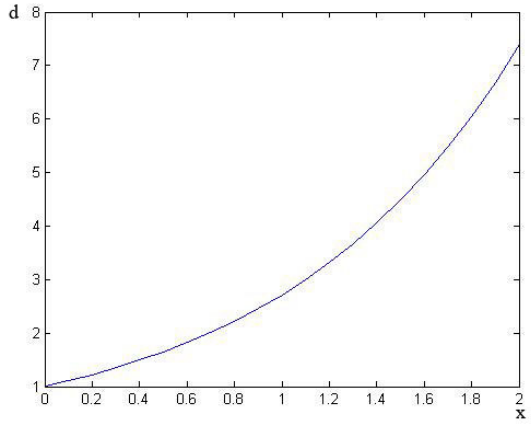

Figure 9. Training data set.

For validating the implementation on the FF environment, a aleatory data set was used. Its respective outputs, as long as the original data set can be checked in figure 10.

Quadratic mean error between output of neural network in FF environment and the original data set was:

$$error = 2,7 \times 10^{-3}$$

4.2 Example 2

The second example presented trains a neural network with a 2-3-1 architecture, shown in figure 11, to learn a 3-dimensional data set shown in figure 12.

Quadratic mean error reached in this example was:

$$error = 2,8 \times 10^{-3}$$

And the graphics that show the validation measures is in figure 13.

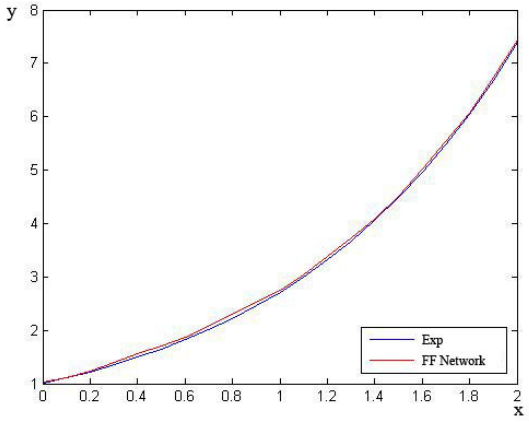

Figure 10. Output of neural network to the validation data set.

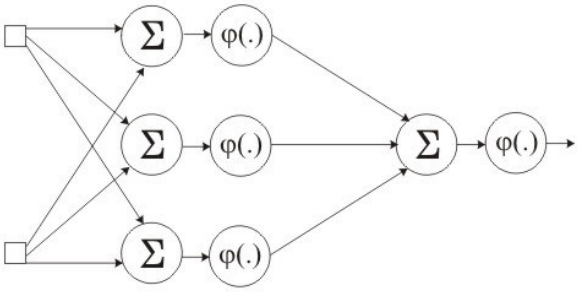

Figure 11. Implemented neural network to learn example 2.

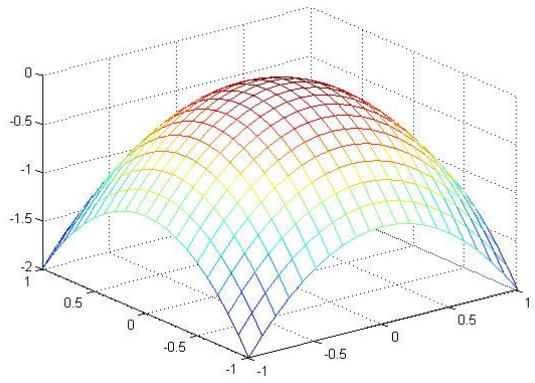

Figure 12. 3D training data set.

5. Conclusions

This work presents the importance of the mathematical tool that are the neural networks and enumerate some possible applications for it in industrial environment. Besides, presents a solution for its implementation in Foundation Fieldbus network environment, which uses only standardized function blocks. That means the implemen-

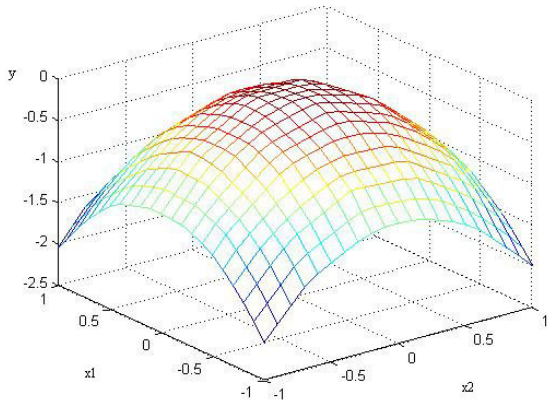

Figure 13. Example 2 validation measures.

tation is possible in any equipment from any vendor, assuming that equipment have the function blocks used here, and also means that the interoperability is guaranteed, a consequence of the use of standardized blocks.

The validation, shown on previous section, allows us to evaluate network performance and helps decide about viability of certain applications. Error rates obtained were considered acceptable for several applications, which proves the success of the presented solution.

References

[1] *Tecnical Information: Foundation Fieldbus*. Samson.

[2] Foundation fieldbus home page. http://www.fieldbus.org, 2005.

[3] J. Berge. *Fieldbus for Process Control: Enginneering, Operation and Maintenance*. ISA - The Instrumentation, System and Automation Society, 2001.

[4] N. K. Bose. *Neural Network Fundamentals with Graphs, Algorithms and Applications*. McGraw-Hill, 1996.

[5] L. Fortuna, S. Graziani, and M. G. Xibilia. Soft sensors for product quality monitoring in debutanizer distillation columns. *Control Engineering Pratice*, 2004.

[6] S. Haykin. *Neural Networks: Principles and Applications*. Bookman, 2001.

[7] Y.-Z. Lu. *Industrial Intelligent Control : Fundamentals and Applications*. John Wiley & Sons, 1996.

[8] S. H. Yang, C. B. H., and X. Y. Wang. Neural network based fault diagnosis using unmeasurable inputs. *Engineering Application of Artificial Intelligence*, 2000.