UNCERTAINTIES 2014

Proceedings of the 2nd International Symposium on Uncertainty Quantification and Stochastic Modeling
July 7th to July 11th, 2014, Rouen, France

DETERMINING INITIAL CONDITION BY FPGA

Sabrina B.M. Sambatti[1], Vitor C.F. Gomes[1], Helaine C. M. Furtado[1], Eduardo F. P. Luz[1], Haroldo F. de Campos Velho[1], Andrea S. Charao[2]

[1] National Institute of Space Research (INPE), [sabrinabms, vconrado, helaine.furtado, eduardofpl]@gmail.com, haroldo.camposvelho@lac.inpe.br
[2] Federal University of Santa Maria (UFSM), andrea@inf.ufsm.br

Abstract. *Data assimilation is a mathematical tool to compute an appropriate combination between observation and data from a mathematical model in order to determine the best initial condition. Advanced methods are Extended Kalman Filter (EKF), and three and four dimensional variation methods (3D-Var, 4D-Var) employed to perform data assimilation. Artificial neural networks can also be applied, once it is able to emulate the EKF or 3D/4D-Var procedures, reducing the computational complexity. The supervised Multilayer Perceptron Artificial Neural Network (MLP-ANN) is used here to emulate the Kalman filter. The MLP-ANN is implemented in a reconfigurable hybrid system: FPGA (Field-Programmable Gate Array). The linear 1D wave equation is the dynamic system used for testing the framework. Good performance was obtained with neural network emulating the Kalman filter. The neural network is automatically configured using the meta-heuristic Multi-Particle Collision Algorithm (MPCA).*

Keywords. *Data assimilation, Kalman filter, Artificial Neural Network, FPGA, MPCA meta-heuristic.*

1 INTRODUCTION

Physical processes can be described by differential equations using mathematical modelling, and it is an approximation of reality. Therefore, modelling errors are inherent to the process. A strategy to mitigate the modelling errors is adding information to the model. Associated information are observations: data from measurements of the phenomena. This procedure is called data assimilation.

Data assimilation can be defined as a methodology for providing an appropriate combination between mathematical model data with observations to determine the analysis data (Daley, 1993). The analysis is a new initial condition to compute a new prediction period, and the process is repeated systematically – if new observations are available (Daley, 1993; Kalnay,2003). Modern techniques are Ensemble Kalman Filter (EnKF) (Houtekamer and Mitchell,2001; Evensen, 2009), Particle Filter (PF) (Gordon, Salmond and Smith,1993; van Leeuwen, 2010) – two Bayesian techniques, and 3D/4D variational methods (Daley, 1993; Kalnay, 2003; Courtier, 1997). However, these schemes are very expensive, from the computational point of view. The use of Artificial Neural Networks (ANN) can reduce the computational effort.

Nowosad et al. (Nowosad, Rios Neto and Campos Velho, 2000) employed a multilayer perceptron artificial neural network (MLP-ANN) for data assimilation emulating the extended Kalman filter. Härter and Campos Velho (Harter and Campos Velho, 2008) employed a local ANN (applied to each grid-point), instead the strategy used before (Nowosad, Rios Neto and Campos Velho, 2000). The application of neural networks for data assimilation was also tested to emulate particle filter (Furtado, 2008), and variational method (Furtado, Campos Velho and Macau, 2011). Recently, the neural network was applied to 3D general circulation atmospheric model SPEEDY (Simplified Parametrizations primitivE-Equation DYnamics). In the latter study, the MLP-ANN was able to produce a similar analysis to that computed by the Local Ensemble Transform Kalman Filter (LETKF) (Miyoshi and Yamane, 2007), but the neural network was 90 times faster than LETKF (Cintra and de Campos Velho, 2012).

The procedure to configure an appropriate neural network to solve a specific problem is a complex task, and usually requires a great effort from the developer, mostly to determine the best parameters. A prior knowledge about the problem to be treated is necessary. This activity can require a long time from an expert (Carvalho, 2011). Therefore, a difficult task is to find out the best configuration of the ANN for a given application. Some studies are addressed to develop automatic schemes for configuring an ANN. In our approach, the configuration of the multilayer perceptron artificial neural network (MLP-ANN) is formulated as an optimization problem. A new meta-heuristic, the Multi-Particle Collision Algorithm (MPCA) (Luz, Becceneri and Campos Velho, 2008), is used to identify the best configuration (Sambatti et al., 2012b). The MPCA is inspired in the behaviour of neutrons transport in a nuclear reactor, where absorption and scattering are considered.

ANN is an intrinsically parallel algorithm, and software implementation fails to take advantage of the inherent parallelism. Currently, the era of the *Data Science* (or *Big Data*), where we are dealing with large volumes of data, indicating the necessity of alternatives to improve the computational performance. Several works bring high performance solutions for this computation, involving hybrid computation: massively parallel environments (Long and Gupta, 2008), and the use of co-processors (GPU: Graphics Processing Units) (Martinez-Zarzuela et al., 2007), and FPGA: Field-Programmable Gate Arrays (Gomes et al., 2011; Shiguemori, 2007).

Here, the data assimilation is performed by FPGA. It is configured to implement a MLP-ANN. In the beginning, FPGA dealt with small memory and slow processing capacity (Omondi and Rajapakse, 2006). Nowadays, the technology

allows the FPGA works as neurocomputers under high computation demanding.

The experiment for data assimilation by using a FPGA-neurocomputer is carried out with linear 1D wave equation (Bennett, 2002; Furtado, Campos Velho and Macau, 2011). A comparison between data assimilation using neural network by software and hardware is shown, where small differences are reported.

2 ARTIFICIAL NEURAL NETWORK

Artificial Neural Networks (ANN) are distributed parallel systems. They are composed by simple process units (node) that calculate certain mathematical functions (typically non-linear), emulating the human brain function. These units can be divided into one or more layers interconnected by synaptic weights (connections), which store the knowledge represented in the model (Haykin, 1994). Figure 1 shows the schematic representation of an artificial neuron, and their fundamental elements are: (a) entries $\{x_1, x_2, ..., x_n\}$; (b) synaptic connections with weights associated $\{w_1, w_2, ..., w_n\}$; bias b_k; and the activation function φ that relates the internal activity of neuron v_k with the output signal y_k.

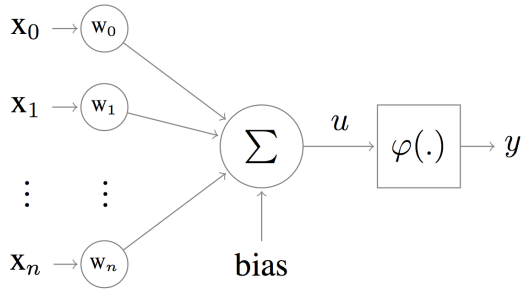

Figure 1: Artificial neuron.

$$y_k = \varphi\left(\sum_{j=1}^{n} w_{kj} x_j + b_k\right) \tag{1}$$

Equation: neuron output.

There are many activation functions that can be applied to create distinct neurons. The following functions are used here: Gaussian function, logistic and hyperbolic tangent function, they are shown in Figure 2:

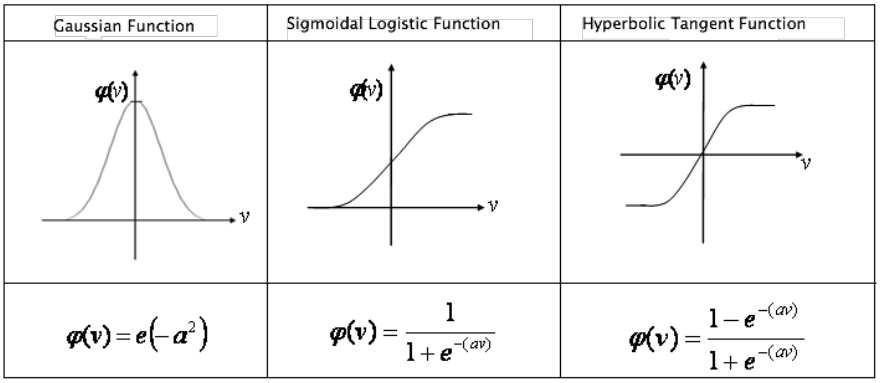

Figure 2: Graphical representation of different activation functions: (a) Gaussian function; (b) logistic function; (c) hyperbolic tangent.

The property of primary significance for a neural network is the ability to learn from the environment, improving its performance through the learning. A neural network learns from its environment through an interactive process of adjustments applied to its synaptic weights and bias levels (Haykin, 1994). In the context of neural networks, the learning process can be defined as a set of well defined rules for solving a specific problem of learning.

The training algorithms are divided into two classes: supervised and unsupervised learning. In the supervised learning, input pattern and the desired output are provided by an external supervisor to set the parameters of the network, in order to find a connection weight between provided pairs of input and output. For this type of training, the output is compared with the desired response and the weights of connections are adjusted by minimizing the error. In the training with unsupervised learning, only the input patterns are presented to the network: there is not an external supervisor to indicate the desired output for input patterns.

2.1 MULTILAYER PERCEPTRON ARTIFICIAL NEURAL NETWORK

It is possible to generate different network architectures by a combination of artificial neurons, and MultiLayer Perceptrons (MLP) is one of the most commonly used topologies, with at least one intermediate layer or hidden. It has been applied successfully to solve several difficult problems by training them with a supervised manner, with a highly popular backpropagation algorithm. This algorithm is based on the error correction learning rule.

The backpropagation learning error consists of two steps through the different layers of the network: a forward step and a backward step. For the forward step, an activity pattern (input vector) is applied thought the nodes of the network, and its effect propagates on the entire the network, layer by layer. Finally, a set of outputs is produced as the actual response of the network. During the backward step, the synaptic weights are all adjusted in accordance with an error correction rule (Haykin, 1994).

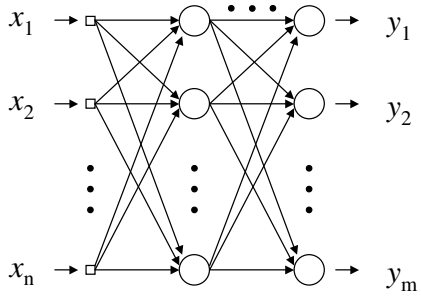

Figure 3: Multilayer Neural Network

Figure 3 shows the architecture of a MLP network comprising: an input layer, where the patterns are presented to the network, an intermediate layer, which works as a recognizer of characteristics that are stored in the synaptic weights and account for most of the processing, and an output layer, where the results are presented.

In order to evaluate the performance of ANN models, the mean square error is used:

$$E_{gen} = \frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y_k})^2 \tag{2}$$

where N is the number of grid points, y_k is the true observational value, and $\hat{y}_k$ is the estimation computed by the neural model.

3 MULTIPLE PARTICLE COLLISION ALGORITHM

The Multiple Particle Collision Algorithm (MPCA) is a stochastic optimization method developed by Luz et al. (Luz, Becceneri and Campos Velho, 2008). The algorithm was prepared to run in a parallel machine. This is a new version of the standard PCA (Particle Collision Algorithm) (Sacco and Oliveira, 2005). The PCA was inspired by the scattering of a particle in a nuclear reactor. The use of the PCA was effective for several test functions and real applications (Sacco and Oliveira, 2005). The MPCA uses multiple particles in a collaborative way, organizing a population of candidate solutions.

The PCA starts with a selection of an initial solution (Old-Config), it is modified by a stochastic perturbation (*Perturbation*$\{.\}$), leading to the construction of a new solution (New-Config). The new solution is compared (function *Fitness*$\{.\}$), and the new solution can or cannot be accepted. If the new solution is not accepted, the scheme of scaterring (*Scaterring*$\{.\}$) is employed. The exploration around closer positions is guaranteed by using the functions *Perturbation*$\{.\}$ and *Small-Perturbation*$\{.\}$. If the new solution is better than the previous one, this new solution is absorbed. If a worse solution is found, the particle can be send to a different location in the search space, it enables the algorithm of escaping a local minimum (Luz, 2012).

The implementation of the MPCA algorithm is similar to PCA, but it uses a set with n particles, where a mechanism to share the particles information is necessary. A blackboard strategy is adopted, where the Best-Fitness information is shared among all particles in the process. This process was implemented in Message Passing Interface (MPI), looking for application into a distributed memory machine (Sambatti, 2004). The pseudo-code for the MPCA is presented by Table 1.

Table 1: MPCA: pseudo-code for the algorithm.

```
Generate an initial solution: Old-Config
Best-Fitness = Fitness{Old-Config}
Update Blackboard
For n = 0 to # of particles
    For n = 0 to # iterations
    Update Blackboard
    Perturbation{.}
        If Fitness{New-Config} > Fitness{Old-Config}
            If Fitness{New-Config} > Best-Fitness
                Best-Fitness = Fitness{New-Config}
            End If
            Old-Config = New-Config
            Exploration{.}
        Else
            Scattering{.}
        End If
    End For
End For
```

4 CONFIGURING THE MLP-ANN BY MPCA

An ANN architecture is not previously known. Usually, the best architecture is empirically determined. However, the problem of identification of an optimized ANN architecture can be formulated as a search in the space of solutions, where each point represents a possible architecture. If a performance value is associated which each point or solution, in such a way that this value is based on some optimality criterion (complexity), it is possible to construct a hyper-surface, where the highest point (or the lowest) is equivalent to the best architecture. Therefore, the problem can be treated as an optimization problem, and the goal is to find the optimum value in this surface, which represents the best combination of variables (Carvalho, 2011).

Many parameters must be evaluated to find the best ANN configuration for a given application: number of hidden layers, number of neurons for each layer, type of activation function, learning ratio, and momentum rate. These parameters are identified by minimizing a cost function. This paper uses an stochastic method called MPCA. The method is enable to make the balance of the behaviour between global search (exploration) and local search (exploitation), this balance is essential to prevent that the search will not be stop in a local optimum, enabling the searching for global optimum (Luz, Becceneri and Campos Velho, 2008).

The optimization problem is formulated by an objective function and a set of restrictions that need to be satisfied. The objective function used in this article is a combination of two factors: square difference between the target values and the ANN output, and a penalty factor. The latter factor is expressed by (Carvalho, 2011):

$$f_{obj} = penalty \times \left(\frac{\rho_1 \times E_{train} + \rho_2 \times E_{gen}}{\rho_1 + \rho_2} \right) \tag{3}$$

where $\rho_1 = 1$ e $\rho_2 = 0.1$ are factors that modify the relevance allowed to the training and generalization error. There is great flexibility in the evaluation of the objective function, because the training error is directly related to the network memory capacity and generalization error refers to the ability of ANN to identify the patterns that are similar to patterns used in training. The function f_{obj} consists of the sum of squared errors for training and generalization multiplied by the penalty, who is responsible by the complexity of neural network architecture in question. The minimum value of f_{obj} corresponds to a simple architecture that displays consistent behaviour in the solution space combined with low training error and generalization. Thus, it is a simple architecture, where the total weights and bias time of learning can be reduced (Sambatti et al., 2012a).

The penalty function is given by (Carvalho, 2011):

$$penalty = c_1 e^{x^2} + c_2 y + 1 \tag{4}$$

where x is the number of neurons, y corresponds to the number of epochs to convergence, c_1 and c_2 are fitting parameters to find the balance between the factors in measuring complexity. This term is used to avoid complex architectures.

The MPCA is employed to identify the best configuration of an ANN, considering: (i) the number of neurons in the intermediate (hidden) layer, (ii) the learning rate parameter η, (iii) momentum constant α. Allowed values for these parameters are shown in Table 2

Table 2: Parameters to define a network architecture.

Parameter	Value
Neuron in the hidden layer	| 1 | ... | 32 |
Learning ratio	| 0.0 | ... | 1.0 |
Momentum constant	| 0.1 | ... | 0.9 |

A set of candidate solutions is generated by MPCA at each iteration, corresponding to different ANN architectures. For each solution, the ANN is activated, and the training process starts until the stopping criterion is satisfied (minimum error, or maximum number of epochs). The ANN output values are obtained, and the MPCA calculates the objective function, up dating the parameters for the ANN. This process is repeated until an optimal value for the objective function is found.

5 DATA ASSIMILATION

Mathematical models describing physical phenomena are imprecise. They have an incomplete scenario of the process. Therefore, the predictions are inexact. The forecast is uncoupled from the reality every time step. The model can be close to the real world by inserting observations. This represents an analysis, a new initial condition. This characterizes a cycle of data assimilation and forecasting. Even if models are considered perfect, observation measures contains errors. For chaotic systems, any small change in initial conditions alter the dynamics, making the data assimilation process necessary. In a simplified manner, it can be said that the data assimilation is a set of techniques to have a proper combination of data from a mathematical model prediction with observation data (Furtado, 2008).

The more accurate is the estimate of the initial condition, the quality of the forecast will be better. For this, it is necessary to use tools of data assimilation to initialize the numerical forecast models.

Mathematically assimilation data is a two step process:
(i) Forecast step:

$$\eta_n^f = M(\eta_{n-1}^a) \tag{5}$$

(ii) Analysis step:

$$\eta_n^a = \eta_n^f + \rho \tag{6}$$

where η_n^f is the vector of state variables of the model provided, the superscripts represent the forecast step and analysis step. M represents the numerical model, ρ is the increment of the analysis or innovation, that is determined according to the technique assimilation used, η_n^a represents the analysis data or initial condition (i.c.).

5.1 KALMAN FILTER

The Kalman filter is a well established statistical estimation process under a stochastic Gaussian process. The algorithm for the cycle of data assimilation, when the observation is available, can be summarized as following:

1. Forecast model for state vector: $\eta_{v,n+1}^f = M_{n+1}\eta_{v,n}^a$, with $\eta_{v,n}^f = [\eta_1^f(t_n)...\eta_{N_x}^f(t_n)]^T$.
2. Update the covariance matrix: $P_{n+1}^f = M_{n+1}P_{n+1}^a M_{n+1}^T + W_n^{Mod}$
3. Compute the Kalman gain: $K_{n+1} = P_{n+1}^f H_{n+1}^T [W_n^{Obs} + H_{n+1}P_{n+1}^f H_{n+1}^T]^{-1}$
4. Compute the analysis (data assimilation): $\eta_v^a = \eta_v^f + K_{n+1}[\eta_v^{Obs} - H_{n+1}\eta_{n+1}^f]$
5. Update the analysis covariance: $P_{n+1}^a = [I - Kn + 1H_{n+1}^T]P_{n+1}^f$

The state value $\eta(x,t)$ is discretized: $\eta(x_i,t_n)$, and the matrix M_n represents the state transition matrix from the state η_n up to η_{n+1} for the discrete dynamical system. Matrices P, H, W^{Obs}, W^{Mod} are the state covariance matrix, observation system matrix, and error covariance matrices for observations and modelling, respectively. The superscript f and a are the predicted values (forecasting, or also background), and the analysis. Subscripts v and n identifies the grid point (x_i) and discrete time (t_n), respectively. Finally, the matrix K is the Kalman gain. Several studies are dedicated to evaluate the matrix W^{Mod}. The ensemble Kalman filter (Evensen, 2009; Cintra and Campos Velho, 2012) is a procedure to identify this matrix. For non-linear and non-Gaussian stochastic process, the particle filter is an alternative estimation procedure (Gordon, Salmond and Smith, 1993; Furtado, Campos Velho and Macau, 2011).

5.2 LINEAR 1D WAVE EQUATION

The dynamical system used in our tests is a linear, first-order partial differential equation, called here linear 1D wave equation:

$$\frac{\partial \eta}{\partial t} + c\frac{\partial \eta}{\partial x} = F(x,t) \tag{7}$$

where η is the unknown variable to be estimated by data assimilation, c is the constant phase speed, $F(x,t)$ is the external forcing, t is time, and x is space. Periodic boundary conditions are assumed. The initial condition is given by the following equation:

$$\eta(x,0) = \eta_0 \frac{1}{\cosh\{2[(x-v)/\Delta]\}} \qquad 0 \le x \le L_x \tag{8}$$

with $\eta_0 = -60$, $v = c + \frac{\alpha\eta_0}{3}$, $\alpha = -1.62 \times 10^{-2}$, $c = 2.42$, and $\Delta = 1340$ (Bennet, 2002). The equation (7) is integrated using finite difference (for space), and Crank-Nicholson method (for time).

6 DATA ASSIMILATON BY HARDWARE DEVICE: FPGA AS AN NEUROCOMPUTER

Cray XD1 is a hybrid system made by a combination of breakthrough interconnect, management and reconfigurable computing technologies. It is a system based on Direct Connected Processor (DCP) architecture, which optimizes message-passing applications by directly linking processors to each other through high performance interconnected fabric, eliminating shared memory contention (Cray, 2005). It is made up by six interconnected nodes (blades), each one containing two 2.4 GHz AMD Opteron general-purpose processors and one Xilinx Virtex II Pro FPGA (Field-programmable Gate Array). Figure 4 shows the architecture of a XD1 node (blade).

Inside the Cray XD1, the FPGA can access different levels of memory. The Dynamic Random Access Memory (DRAM) has up to 8GB and variable reading latency. A board with QDR II SRAM (QDR: Quad Data Rate; SRAM: Static Random- Access Memory) - it is a type of static RAM computer memory with capacity to transfer up to four words of data in each clock cycle - is available. Four independent QDR II SRAM with 4MB can be directly accessed by FPGA, with latency of 8 cycles. There is an internal FPGA memory, with smaller capacity, and it can be accessed every cycle.

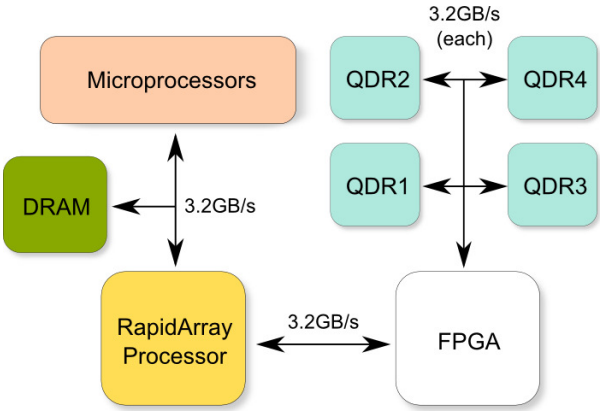

Figure 4: Sketch for the Cray XD1 blade.

The reconfigurable device has direct access to four banks of data transfer between the CPU (Central Processing Unit) and FPGA is performed by the API (Application Programming Interface) RapidArray transport core. This API allows the CPU sending and receiving data to the FPGA, and the FPGA can read and write on the shared memory regions with the CPU.

The *Communication Unit* is an interface between the RapidArray processor, that links the FPGA to the processors. This entity responsible for transferring the data set from blade memory to FPGA QDR banks. To perform this, the Communication Unit has two parallel processes that communicate directly with the RapidArray processor. The first process is responsible for requesting the data set. The requests are sequentially made at each clock cycle. The second process is responsible for handling responses of the RapidArray processor that can come out of order. The data organization is done using the internal memory of the FPGA before being sent to the module responsible for memory management.

The use of FPGAs in HPC (High-performance computing) systems can provide three distinct advantages over conventional compute clusters. Firstly, FPGAs consume less power than conventional microprocessors; secondly, using FPGAs as accelerators can significantly increases compute density; and thirdly, FPGAs can provide a significant increase in performance for a certain set of applications (Wain et al., 2006).

6.1 MLP-ANN FOR THE CRAY XD1

The implementation of the MLP-ANN on FPGA, designed for the data assimilation, has different modules. Each module is embedded into other modules as computation components.

The MAC (Multiplier and Accumulator) unit - see Figure 5a - stores the result of the product between inputs and synaptic weights, adding the bias. Each input x_i is multiplied by a weight w_i and added to the register *acc* or to the bias. For selecting the operation to be done, the signal *fc* is provided. The next module is the artificial neuron, and it uses a MAC and control structures - see the diagram in Figure 5b. In this module, the *fc* signal is generated by a shifting register with only one bit. For the weights management, interconnected registers are used on the circular queue. The weights are shifted at each x_i input. The last computational module is a combination of neurons, with the inputs are connected by a unique bus. The output of each neuron is connected to a position of a shifting register with parallel loading - see Figure 5c. The neurons can receive data, and the results (outputs) are flowing to the Lookup Table (LUT) unit: this is operation to simulate the activation function.

The MLP-ANN design is complete with serial concatenating of layers forming an artificial neural network. The input of each layer is directly concatenated to the output of the previous layer. Considering a layer as a computation module, a pipeline of a operation sequence is performed. The computation for each layer can be independently executed, allowing that multiples data set can be computed with a sequential delay for each computation layer.

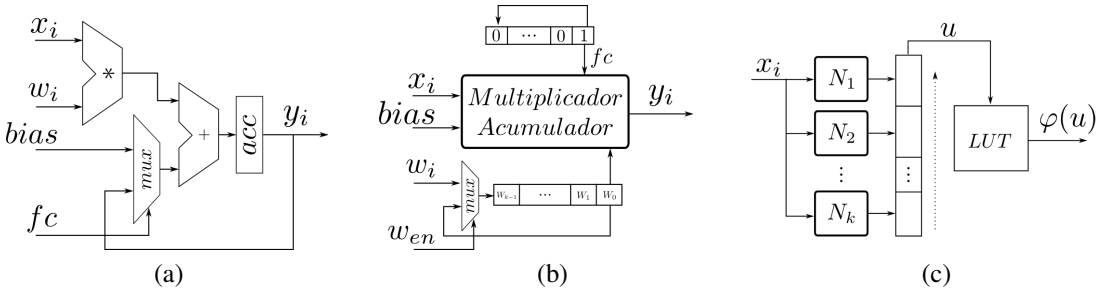

Figure 5: ANN on FPGA: (a) Multiplier and accumulator (MAC), (b) neuron, (c) ANN implemented: the pipeline.

7 RESULTS

The linear 1D wave equation was discretized using $N_x = 128$ grid points for the space mesh ($L_x = N_x$), with time integration up to 200 time-steps ($t_{max} = N_t$ seconds). For the Kalman filter implementation, the following values were

assumed (Furtado, Campos Velho and Macau, 2011):

$$W_n^{Mod} = 0.1I, \qquad W_n^{Obs} = 0.5I, \qquad H_n = I \tag{9}$$

with I being the identify matrix. The observations were synthetically generated, and the assimilation cycle is carried out every 10 time-steps. The synthetic observations were computed from time integration of equation (7), and adding to the results a Gaussian random noise with 5% for level of noise.

As mentioned earlier, one of the goal is to identify an artificial neural network topology for emulating the assimilation process carried out by Kalman filter.

Parameters used by the MPCA were: 6 particles (one particle per processor, with total of six processors), maximum number of iterations = 20. It was adopted as stopping criterion the maximum number of evaluations of the objective function. The results are presented considering the of 4 experiments, with different seeds to generate random numbers.

The MPCA was applied to optimize the parameters of ANN, identifying: number of neurons in the hidden layer, activation function, learning rate, and momentum and the number of layers was fixed at one.

The data assimilation by ANN can be expressed as:

$$\eta_n^a = F_{ANN}(\eta_n^{Obs}, \eta_n^f) \tag{10}$$

where F_{ANN} represents the multilayer Perceptron artifidial neural network. Such neural network requires supervised training. The desired output neural network is the analysis estimate with Kalman filter.

The numerical experiment was performed with two different initial guesses for connection weights and bias. In the first experiment, the initial guess was the fixed value 0.5 for all weights and bias. To the second experiment, weights and bias were initialized with random value. The best architectures found with MPCA correspond to the parameters showed in the following table 3.

Table 3: ANN Topologies

Parameters	NN1-MPCA	NN2-MPCA	ANN-Empirical
Hide layer	1	1	1
Hide layer neuron	3	23	1
Learning tax	0.53	0.4	0.9
Momentum	0.20	0.6	0.0
Activation Function	*Tanh*	*Tanh*	*Tanh*
Quadratic Error	0.5264	0.1583	0.4212

Two different topologies of ANN were implemented on FPGA: the MLP-NN described by (Furtado, Campos Velho and Macau, 2011) with two inputs (forecasting and observation), three neurons in the hidden layer, and one neuron for the output layer, and the MLP-NN2: the best topology defined by MPCA. The MLP-ANN project was implemented for the hybrid system Cray XD1, the VHDL (VHSIC Hardware Description Language) implementation for mentioned ANNs was compiled. Due to the low affinity of FPGAs with floating point operations, the computations were done using 16-bit fixed point, being divided in half for the integer and fractional parts. The activation function is computed and storage at the Lookup Table (LUT) unit.

To evaluate the results of FPGA-neurocomputer implementations, a version was implemented on software (CPU: Central Processing Unit). The results are shown in Figure 6, corresponding on software and FPGA implementations, and the square difference between these two implementations. The results for the ANN defined by an expert is shown in Figure 6 (left), and the results for ANN self-configured by MPCA is also depicted in Figure 6 (right). The values showed in Table 4 correspond to average square error, and variance.

Table 4: Results

	ANN2-MPCA	ANN-Empirical
Average square error	$1.68 * 10^{-5}$	$5.18 * 10^{-5}$
Variance	$1.69 * 10^{-9}$	$1.79 * 10^{-8}$

8 CONCLUSION

Artificial neural networks can be designed as a method for data assimilation. Here, the MLP-ANN was applied to emulate the Kalman filter to the wave 1D dynamical system. The implementation on FPGA works well, where the fixed point arithmetic was adopted for avoiding memory constraints. In a future work, we want to repeat the experiment, but employing the floating-point arithmetic. In the FPGA implementation, the activation function is not codified as a mathematical function, a lookup table approach was employed. The strategy for the automatic configuration of the MLP-ANN using MPCA meta-heuristic was effective, with application for data assimilation. Actually, the computed ANN topology produced better results than a configuration defined by an expert.

According to (Cintra, 2010), the neural network used to emulate the LETKF for the SPEEDY model (3D meteorological spectral code), during one month simulation, was able to reduce the computational effort for the data assimilation from 4 hours and 20 min to less than 3 min (reduction of 95% for the CPU-time). In that simulation, 6 ANNs were defined for different regions in the globe. This indicates a necessity of an automatic process to configure the ANN. The implementation on FPGA can represent an improvement for the performance to the assimilation module.

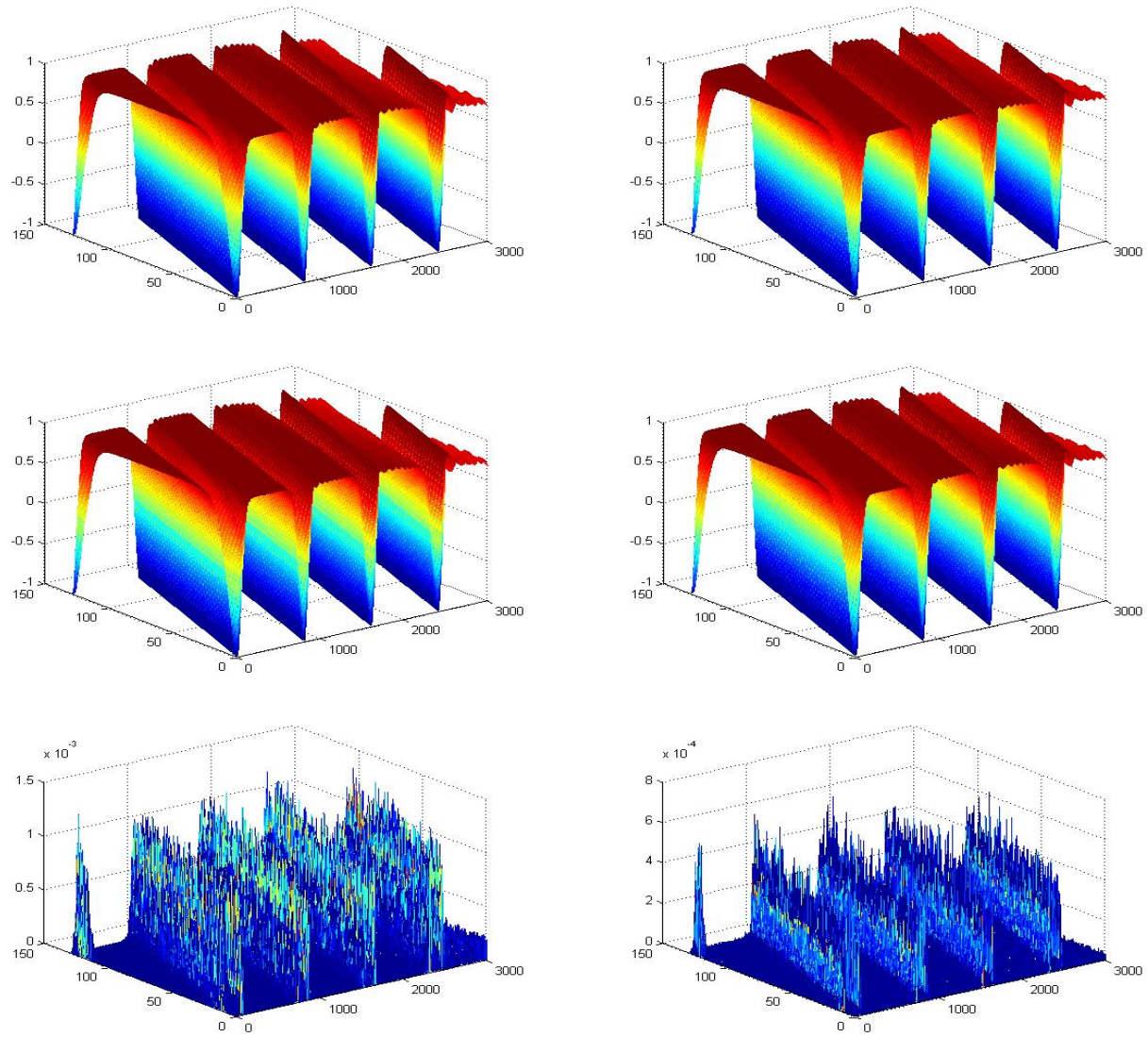

Figure 6: Left: empirical ANN (upper: software implementation, middle: FPGA implementation, bottom: square difference). Right: MPCA-ANN2 (upper: software implementation, middle: FPGA implementation, bottom: square difference).

ACKNOWLEDGEMENTS

The authors would like to thank CAPES (Coordination of Improvement of Higher Education Personnel) for the financial support.

9 REFERENCES

Bennett, A. F., 2002, “Inverse Modeling of The Ocean and Atmosfere,” Cambridge University Press.

Carvalho, A. R., 2011, “Uso de redes neurais para recuperação do perfil de concentração de gases traço atmosfficos a partir de dados de satélites,” PhD thesis, National Institute of Space Research (INPE), São José dos Campos, Brasil.

Cintra, R. S. C., 2010, ”Assimilação de dados com redes neurais em modelo de circulação geral da atmosfera,” PhD thesis, National Institute of Space Research(INPE), São José dos Campos, Brasil.

Cintra, R. S. C. and Campos Velho, H. F., 2012, ”Global data assimilation using artificial neural networks in seedy mode,” In 1st International Symposium Uncertainty Quantification and Stochastic Modeling, pages 648–654, Maresias. International Symposium Uncertainty Quantification and Stochastic Modeling.

Courtier, P., 1997, ”Dual formulation of four-dimensional variational assimilation,” Quarterly Journal of the Royal Meteorological Society, 123(544):2449–2461.

Daley, R., 1993,“Atmospheric Data Analysis,” Cambridge University Press.

Evensen, G., 2009, ”Data assimilation: the emsemble Kalman filter, ” Springer.

Furtado, H. C. M., Campos Velho, H. and Macau, E., 2011, ”Assimilação de dados com redes neurais artificiais em equações diferenciais”, DINCON, pages 595–598.

Furtado, H. C. M., 2008, "Redes neurais e diferentes métodos de assimilação de dados em dinâmica não linear," Masters thesis, National Institute of Space Research, São José dos Campos, Brasil.

Gomes, V. C. F., Shiguemori, E. H., Charão, A. S. and Campos Velho, H. F., 2011, "Rede perceptron de múltiplas camadas para sistema híbrido reconfigurável," Workshop Applied Computing (Worcap).

Gordon, N. J., Salmond, D. J. and Smith, A. F., 1993, "Novel approach to nonlinear/non-gaussian bayesian state estimation," In IEE Proceedings F (Radar and Signal Processing), v 140, pages 107–113. IET.

Harter, F. P. and Campos Velho, H. F., 2008, "New approach to applying neural network in nonlinear dynamic model," Applied Mathematical Modelling, 32(12):2621–2633

Haykin, S., 1994, "Neural networks: a comprehensive foundation," Prentice Hall Inc.

Houtekamer, P. L. and Mitchell, H. L., 2001, "A sequential ensemble kalman filter for atmospheric data assimilation,' Monthly Weather Review, 129(1):123–137.

Kalnay, E., 2003, "Atmospheric modeling, data assimilation and predictability," Cambridge Univ. Press. Cambridge.

Long, L. N. and Gupta, A., 2008, " Scalable massively parallel artificial neural networks," Journal of Aerospace Computing, Information and Communication, 5(1):3–15.

Luz, E. F. P., 2012, "Meta-heurísticas paralelas na solução de problemas inversos," PhD thesis, National Institute of Space Research (INPE), São José dos Campos, Brasil.

Luz, E. F. P., Becceneri, J. C. and Campos Velho, H. F., 2008, "A new multi-particle collision algorithm for optimization in a high performance environment. Journal of Computational Interdisciplinary Sciences, 1(1):3–10.

Martínez-Zarzuela, M., Pernas, F. J. D., Higuera, J. F. D. and Rodríguez, M. A., 2007, "Fuzzy art neural network parallel computing on the gpu," In Computational and Ambient Intelligence, pages 463–470. Springer.

Miyoshi, T. and Yamane, S., 2007, "Local ensemble transform kalman filtering with an agcm at a t159/l48 resolution," Monthly Weather Review, 135(11):3841–3861.

Nowosad, A., Rios Neto, A. and Campos Velho, H. F., 2000, "Data assimilation in chaotic dynamics using neural networks," In Third International Conference on Nonlinear Dynamics, Chaos, Control and Their Applications in Engineering Sciences, pages 212–221.

Omondi, A. R. and Rajapakse, J. C., 2006, "FPGA implementations of neural networks," Springer New York, New York, NY, USA.

Sacco, W. F. and Oliveira, C. R. E., 2005, "A new stochastic optimization algorithm based on a particle collision metaheuristic," Proceedings of 6th WCSMO.

Sambatti, S. B. M., 2004, "Paralelização de um algorítmo genético em problema inverso de conduçã de calor. Master's thesis, National Institute of Space Research (INPE), São José dos Campos, Brasil.

Sambatti, S. B. M., Anochi, J. A., Luz, E. F. P., Carvalho, A. R., Shiguemori, E. H. and Campos Velho, H. F., 2012a, "Automatic configuration for neural network applied to atmospheric temperature profile identification," International Conference on Engineering Optimization.

Sambatti, S. B. M., Anochi, J. A., Luz, E. F. P., Carvalho, A. R., Shiguemori, E. H. and Campos Velho, H. F., 2012b, "MPCA meta-heuristics for automatic architecture optimization of a supervised artificial neural network," 10th World Congress on Computational Mechanics.

Shiguemori, E. H., 2007, "REcuperação de perfis de temperatura e umidade da atmosfera a partir de dados de satélites - abordagens com redes neurais artificias e implemetação em hardware,' PhD thesis, National Institute of Space Research (INPE), São José dos Campos, Brasil.

van Leeuwen, P. J., 2010, "Nonlinear data assimilation in geosciences: an extremely efficient particle filter,' Quarterly Journal of the Royal Meteorological Society, 136(653):1991–1999.

Wain, R., Bush, I., Guest, M., Deegan, M., Kozin, I. and Kitchen, C., 2006, "An overview of FPGAs and FPGA programming: Initial experiences at Daresbury," Council for the Central Laboratory of the Research Councils.

RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.