Self-configured neural network for data assimilation using FPGA for ocean circulation

Sabrina B. M. Sambatti[a 1], Haroldo Fraga de Campos Velhos [a], Helaine C. M. Furtado [b], Vitor C. Gomes [c] and Andrea S. Charão [d]

[a]National Institute for Space Research, São José dos Campos, SP, Brazil
[b]Universidade Federal do Oeste do Par, Santarém, PA, Brazil
[c]Instituto de Estudos Avanados, Instituto de Estudos Avanados do DCTA, São José dos Campos, SP, Brazil
[d]Universidade Federal de Santa Maria, Santa Maria, RS, Brazil

Abstract

Physical processes can be represented mathematically by differential equations whose solutions are obtained employing numerical methods. The model never represent reality because there are disagreement between the simulation model and the real world Adding observational information to the model, the simulation error can be mitigated. This process of combining data with observation data from a mathematical model is known as data assimilation (DA). Several techniques have been developed to DA as Kalman filter, particle filter and variational methods; however, the cited methods involve a high computational cost and an approach to reduce this cost is to make use of Artificial Neural Networks (ANN). The definition of a quasi-optimal topology for a neural network is a major challenge.An auto-configuration approach to identify the best topology of RNA was adopted. The ideal of self-RNA configuration was addressed as an optimization problem and solved by a new metaheuristic: Multiple Particles Collision Algorithm (MPCA). The dynamic model for testing this new approach is described by 2D shallow water equations used to simulate the ocean circulation. The data assimilation tests were performed by an RNA multi-layer perceptron trained to emulate the Kalman filter is implemented in an FPGA (field-programmable gate array).

Keywords: Data Assimilation, Artificial Neural Network, FPGA.

1. Introduction

Data assimilation (DA) is an strategy to mitigate the modelling errors, providing an appropriate combination between data from the mathematical model and observations. The latter data fusion is called the *analysis* [1]. Modern techniques for DA are Ensemble Kalman Filter (EnKF), Particle Filter (PF), and 3D/4D variational methods. However, these schemes are computational expensive. Other algorithms can be applied to reduce the

[1]E-mail Corresponding Author: sabrinabms@gmail.com

computational effort. Here, the Artificial Neural Networks (ANN) is employed. Indeed, the application of ANN was tested for data assimilation to emulate Kalman filter, particle filter, and variational method [2]. Recently, the neural network was applied to 3D general circulation atmospheric model SPEEDY (Simplified Parametrization primitivE-Equation DYnamics) emulating the Local Ensemble Transform Kalman filter (LETKF) [3].

The appropriate configuration of an ANN topology is a complex task, and requires a significant effort from the developer. Some studies are addressed to develop automatic schemes for configuring an ANN [4]. In our approach, the configuration of multi-layer perceptron neural network (MLP-NN) is formulated as an optimization problem. Multiple Particle Collision Algorithm (MPCA)[5] is used to solve the optimization problem.

ANN is an intrinsically parallel algorithm. Software implementation has difficulty to take advantage of the inherent parallelism. Hybrid computers, mixing CPU and co-processors, has emerged using GPU (Graphics Processing Units), FPGA (Field-Programmable Gate Arrays), MIC (Many Integrated Cores).

Here, the DA is performed by FPGA. The results were obtained with the hybrid computer Cray XD1 (12 processors and 6 FPGAs). The FPGA is configured to implement a MLP-NN trained to emulate a Kalman filter. The two dimenson shallow water model applied to simulate the oceanic circulation is the prediction model [6].

2. Artificial Neural Network

Artificial neural network is a machine designed to model the behaviour of brain carrying out a particular task or function of interest [7]. They are composed by artificial neurons for calculating certain mathematical function. The neurons may be arranged within one or more layers, and the neurons are interconnected [8]. Mathematically, an artificial neuron k, could be described according to the following equations [7]:

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

where $x_1, x_2, ..., x_n$ are the input signals, w_{kj} are the synaptic weights of neuron k, v_k is the linear combination among the input signals, b_k is the bias; φ is the activation function, and y_k is the output signal of the neuron.

The activation function, represented by φ, defines the output of a neuron in terms of the induced local field v[7]. There are various types of activation functions that can be used: Gaussian function, logistic, Heaviside function, and hyperbolic tangent function.

2.1. Multi-layer Perceptron Artificial Neural Network

Multi-layer perceptrons have been applied successfully to solve some difficult problems by training them with a popular back-propagation algorithm, a supervised algorithm based on the error correction [7].

The overall architecture of a MLP-ANN comprising: an input layer, where the patterns are presented to the network, one or more intermediate layers, 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

Multiple Particle Collision Algorithm (MPCA) is an stochastic optimization method developed by Luz et al. [5]. The MPCA is a new version of the PCA (Particle Collision Algorithm) [9]. The latted was inspired traveling particle in a nuclear reactor. The MPCA was prepared to run in a parallel machine and 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 scattering (*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[10].

The implementation of the MPCA algorithm uses a set with n particles, where a blackboard strategy mechanism is used to share the particle information. The best-fitness information is shared among all particles in the process implemented with Message Passing Interface (MPI).

3.1. Configuring the MLP-ANN by MPCA

ANN architecture is not previously known. The identification of an optimal architecture can be formulated as a search problem in the solution space, where each point represents a possible architecture. If a performance value is associated which each point,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 [4].

The optimization problem is formulated by an objective function, and a set of restrictions needs to be satisfied. The objective function used 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 [4]:

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

where $\rho_1 = 1$ e $\rho_2 = 0.1$ are factors that modify the relevance allowed to the training and generalization error. 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.

The penalty function is given by [4]:

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

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.

The MPCA is employed to identify the best configuration of an ANN, considering: (i) the number of neurons in the intermediate layer, (ii) the learning rate parameter η, (iii) momentum constant α. 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. 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.

4. Data Assimilation

Data assimilation is a set of techniques to have a proper combination of data from a mathematical model prediction with observation data [11].

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 DA to initialize the numerical forecast models.

Mathematically, data assimilation 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.).

4.1. Assimilation: Kalman Filter

The Kalman filter is a well established statistical estimation process under a stochastic Gaussian process. The algorithm for the cycle of DA, 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.

5. Shallow Water Equations

The shallow water equations is a well known model. The system was firstly derived for ocean simulation, but it has also been used in meteorology. The equaions are expressed by [6]:

$$\frac{\partial u}{\partial t} - fv + g\frac{\partial q}{\partial x} + r_u u = F_u \tag{7}$$

$$\frac{\partial v}{\partial t} + fu + g\frac{\partial q}{\partial y} + r_v v = F_v \tag{8}$$

$$\frac{\partial q}{\partial t} + H(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}) + r_q q = 0 \tag{9}$$

on the domain $(x, y) \in (0, X) \times (0, Y)$, with f is the Coriolis coefficient, the gravitational constant is denoted by g, u and v are, respectively, velocity in the direction x and y, q is the sea-level disturbance, H is average depth of the ocean, and the external forcing model are F_u and F_v.

6. Data Assimilation by Hardware Device

Cray XD1 is a hybrid system composed by six interconnected nodes (blades), each one containing two 2.4 GHz AMD Opteron general-purpose processors and one Xilinx Virtex II Pro FPGA. The Cray offers the RapidArray API to allow communication between the FPGA and the processor blade.

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 CPU (Central Processing Unit); 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 [12].

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 (Figure 1a) stores the result of the product between inputs and synaptic weights, adding 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 (Figure 1b). 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 (Figure 1c). The neurons can receive data, and the results are flowing to the Lookup Table (LUT) unit: this is operation to simulate the activation function.

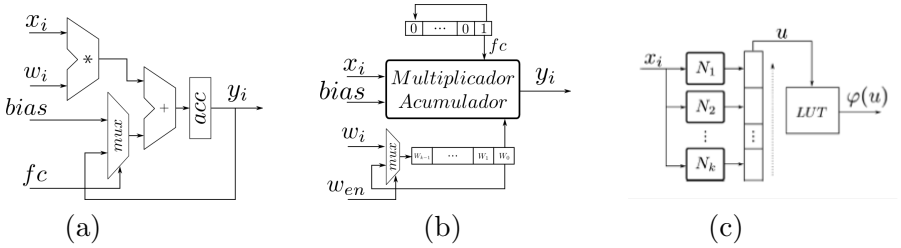


Figure 1: ANN on FPGA: (a) Multiplier and accumulator (MAC), (b) neuron, (c) ANN implemented: the pipeline.

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.

7. Results

The shallow water 2D model was spatially discretized with the Arakawa C-Grid with a forward-backward scheme for time-stepping [13]. The parameters used for the model integration were adopted to reproduce the experiment described in the Bennett's book [6]

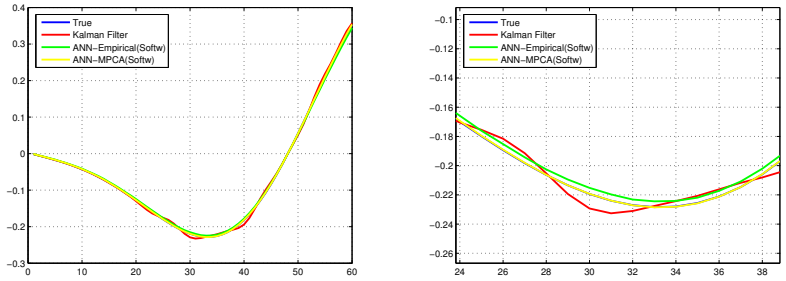
As already mentioned, the MPCA was applied to optimize the parameters of ANN, and the Table 1 shows the obtained results, that corresponding to an average of 15 experiment with seeds generate different random numbers. The parameters used to run the MPCA were: 1 particles per processor, 8 processors. The stopping criterion used was the maximum number of evaluation of the objective function.

The shallow water equations were integrated at 60 time steps, the q variable was initialized with Gaussian function and the $u = v = 0$ at $t = 0$, and the data assimilation process was made each 10 time step. In order to training the MLP-ANN, the data set was made until 40 time steps, and the remaining time steps were used to the generalization phase of ANN.

Table 1: ANN Topologies

Parameters	ANN-Empirical	ANN-MPCA
Hidden layer	1	1
Neuron hidden layer	10	10
Activation Function	tanh	tanh
Quadratic Error	0.5264	0.1583

The Figure 2(a) shows the evolution of variable q and makes a comparison between the results obtained with: Kalman Filter, ANN defined by an expert and ANN self-configured by MPCA. In Figure 2(b) can be seen that the result obtained with ANN self-configured It can be seen that the result obtained with ANN-MPCA is closer to the truth.



(a) (b)

Figure 2: Temporal evolution of point $q(59, 59)$: software results

An other comparison was made: results obtained with software and hardware implementation of ANN Empirical and ANN-MPCA, the results can be seen in Figure 3.

Conclusion

Artificial neural networks can be designed as a method for data assimilation. Here, the MLP-NN was applied to emulate the Kalman filter to the 2D shallow water equations. The implementation on FPGA works well, where the fixed point arithmetic was adopted for avoiding memory constraints. In the FPGA implementation, the activation function is not codified as a mathematical function, instead a look at table approach was employed. The strategy for the automatic configuration of the MLP-NN using MPCA metaheuristic was effective, with application for data assimilation. Actually, the

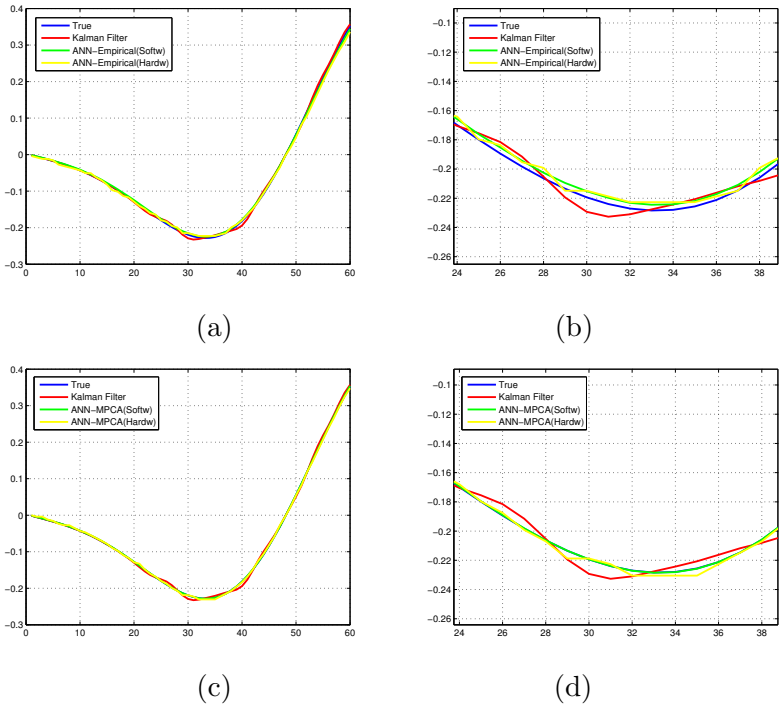


Figure 3: Temporal evolution of point $q(59, 59)$: software and hardware results

computed ANN topology produced better results than a configuration defined by an expert.

Acknowledgements:

The authors thank to the CNPq (Conselho Nacional de Pesquisa e Desenvolvimento), Brazilian agency for research support.

References

[1] R. Daley, Atmospheric data analysis, Vol. 2, Cambridge university press, 1993.

[2] H. F. d. C. Helaine C. M. Furtado, E. E. N. Macau, Anais do DINCON.

[3] R. Cintra, H. F. Campos Velho, Global data assimilation using artificial neural networks in seedy mode, International Symposium Uncertainty Quantification and Stochastic Modeling, Maresias, 2012.