DOI: 10.14529/jsfi220105

Data Assimilation by Neural Network for Ocean Circulation: Parallel Implementation

Haroldo F. Campos Velho[1]***, Helaine C. M. Furtado***[2]***,***
Sabrina B. M. Sambatti[3]***, Carla O. F. Barros***[4]***, Maria E. S. Welter***[4]***,***
Roberto P. Souto[4]***, Diego Carvalho***[5]***, Douglas O. Cardoso***[6,7]

Data assimilation (DA) is an essential issue for operational prediction centers, where a computer code is applied to simulate physical phenomena by solving differential equations. The procedure to determine the best initial condition combining data from observation and previous forecasting (*background*) is carried out by a data assimilation method. The Kalman filter (KF) is a technique for data assimilation, but it is computationally expensive. An approach to reduce the computational effort for DA is to emulate the KF by a neural network. The multi-layer perceptron neural network (MLP-NN) is employed to emulate the Kalman in a 2D ocean circulation model, and algorithmic complexity to KF and NN is presented. A shallow-water system models the ocean dynamics. Synthetic measurements are used for evaluating the MLP-NN for the data assimilation process. Here, a parallel version for the DA procedure by the neural network is described and tested, showing the performance improvement for a parallel version of the NN-DA.

Keywords: data assimilation, artificial neural network, shallow water equations, parallel processing.

Introduction

An essential step for the prediction centers is identifying the better set of initial conditions for each prediction period. Several issues are involved for producing good predictions from a mathematical model, starting with calculation of the best possible initial condition. In this context, the analysis typically employs data assimilation (DA) procedure, which combines observational data with the previous prediction (*background*) of a numerical model to obtain an optimal estimate of the evolving system state. Weather services were the first centers to employ schemes for data assimilation, using several methods such as optimal interpolation, Kalman filter, variational approaches, and particle filter [8, 11, 15].

The mentioned methods for DA are computing-intensive [11], and one alternative to reduce computer processing time is to adopt a neural network (NN) formulation to emulate or replace parts or the entire method. Here, the artificial neural network architecture based on *multi-layer perceptron* (MLP) formulation will be employed. The MLP is a supervised fully-connected neural network, requiring a training algorithm to compute interconnecting weights. The back-propagation algorithm is applied for training the MLP-NN as a surrogate for the Kalman filter for the DA process.

The literature has registered applications of neural networks for DA in forecasting systems in different areas of geophysics: meteorology [6, 7], hydrology modeling [3], and space weather application [4]. Here, the DA by the neural network is applied to the shallow-water equation

[1]National Institute for Space Research, São José dos Campos, Brazil
[2]Federal University of Western Pará, Santarém, Brazil
[3]Independent researcher, São José dos Campos, Brazil
[4]National Laboratory for Scientific Computing, Petrópolis, Brazil
[5]Federal Center for Technological Education Celso Suckow da Fonseca, Rio de Janeiro, Brazil
[6]Federal Center for Technological Education Celso Suckow da Fonseca, Petrópolis, Brazil
[7]Polytechnic Institute of Tomar, Tomar, Portugal

(SWE) designed to represent ocean circulation dynamics as described in Bennett's book [2]. The shallow water system can be used for different applications in geophysical fluid dynamics, the dimension describing the domain and mainly parameters associated to the simulation are used to determine a good representation of this system for the application expressed in the numerical experiment. Next section will be to describe the SWE configured to the ocean circulation [2].

The supervised MLP-NN is designed to emulate the Kalman filter for DA – see also references [9, 16] for 2D shallow water system and reference [4] for space weather applications. Besides, in this paper, a parallel version of the DA procedure is presented aiming to enhance the computational performance for neural data assimilation. In some previous results, the MLP-NN was used to emulate KF, focusing on data assimilation. The algorithmic complexity of these two schemes is included in this paper.

The following section describes the mathematical model used as a prediction system. Data assimilation methods – Kalman filter and neural network – are presented in Section 2. The strategy for parallel implementation is commented on in Section 3. Results for speed-up and efficiency are shown in Section 4. The last section offers conclusions and final remarks.

1. Shallow-Water Equations as a Model for Ocean Circulation

The shallow-water approach works for fluid dynamics simulation, assuming that the vertical dimension is much smaller than the horizontal dimension. Integrating the NavierStokes equations on vertical coordinate, a 2D system of partial differential equations (PDE) is derived. The latter PDE system is called *shallow water equations* (SWE). The SWE can be solved by applying a numerical algorithm, and it is able to model the ocean circulation dynamics. Here, the same SWE presented in the Bennett's book [2] is employed. The 2D SWE is described with fluid depth (H), coupled to a velocity field (u, v). The three independent variables (u, v, q) are under influence of gravitational force (g) acting on the fluid. Mathematical equations for the worked model are given by:

$$\frac{\partial u}{\partial t} - fv + g\frac{\partial q}{\partial x} + r_u u = F_u \ , \tag{1}$$

$$\frac{\partial v}{\partial t} + fu + g\frac{\partial q}{\partial y} + r_v v = F_v \ , \tag{2}$$

$$\frac{\partial q}{\partial t} + H\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) + r_q q = 0 \ , \tag{3}$$

where $t > 0$ and $(x, y) \in (0, L_x) \times (0, L_y)$, the Coriolis parameter is denoted by f, the damping coefficients are represented by (r_u, r_v, r_q) – these coefficients are linked with the linearization process of the nonlinear terms, (u, v) is the velocity field, F_u and F_v are external forcing, H is the mean depth of the ocean, and the free perturbed ocean surface is defined by q. Boundary conditions are shown in Fig. 1.

The forcing terms are expressed as:

$$\begin{aligned} F_u &= -C_d\,\rho_a\,u_a^2/(H\,\rho_w) \ , \\ F_v &= 0 \ , \end{aligned} \tag{4}$$

where C_d is the drag coefficient, ρ_a and ρ_w are air and ocean water densities – respectively, u_a is the zonal wind forcing.

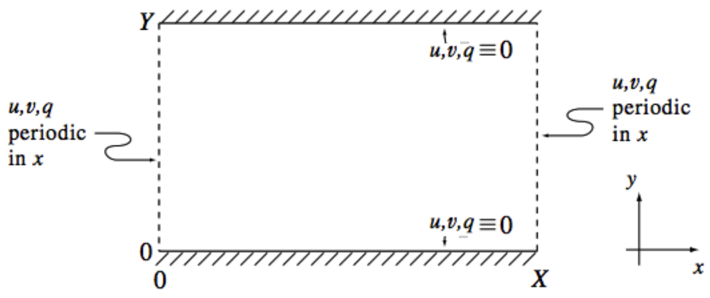


Figure 1. Boundary conditions for the equations (1)–(3)

The spatial discretization for the system (1)–(3) is carried out by finite difference technique, and the forward-backward method is applied for time integration [14]. The space mesh 2D discretization follows the Arakawa grid-C scheme – see Fig. 2.

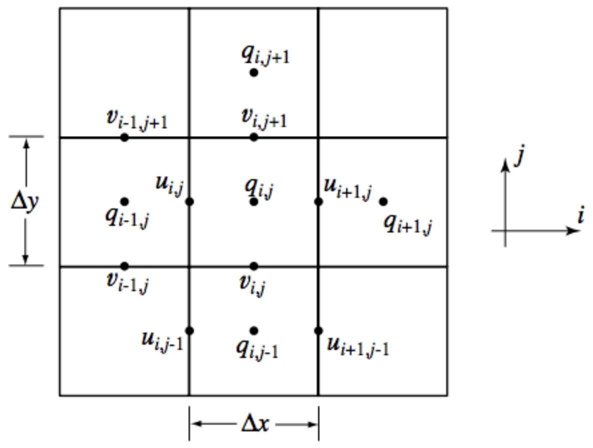


Figure 2. The Arakawa grid-C employed for space discretization

2. Data Assimilation Methods

As already mentioned, data assimilation is a scheme to compute the initial condition by combining the background fields with the available observations, producing the *analysis*. The supervised artificial neural network is self-configured to emulate the Kalman filter. Therefore, both DA methods are presented in this section.

2.1. Kalman Filter

A Kalman Filter is a well-known scheme to estimate the unknown state by least-squares taking into account the Gaussian statistics for the measurement and the modeling errors. Data assimilation algorithm using Kalman filter is written as:

1. Forecast model for state vector (M_N is the matrix of the system): $x^f_{n+1} = M\{x^a_n\} \approx M_n\, x^a_n$.

2. Update the forecasting covariance matrix (W^{mod} is the modeling covariance error matrix, and P^a the analysis covariance matrix – see item 5):
$P^f_{n+1} = M_n P^a_n M^T_n + W^{\text{mod}}_n$.
3. Compute the Kalman gain (W^{obs} is the measurement covariance error matrix):
$K_{n+1} = P^f_{n+1} H^T_{n+1} [W^{\text{obs}}_{n+1} + H_{n+1} P^f_{n+1} H^T_{n+1}]^{-1}$.
4. Compute the analysis (DA) – x^a_{n+1}, with x^{obs} being the state observation vector:
$x^a_{n+1} = x^f_{n+1} + K_{n+1}[x^{\text{obs}}_{n+1} - (H_{n+1} x^f_{n+1})]$.
5. Update the analysis covariance: $P^a_{n+1} = [I - K_{n+1} H_{n+1}] P^f_{n+1}$.

Data assimilation using the Kalman filter has a high computational effort. The processing DA cost can be mitigated by using artificial neural networks trained to emulate the KF. In fact, the MLP-NN has a lower computational complexity than KF, as shown below.

2.2. Artificial Neural Network (ANN)

Artificial neural networks (ANN) have been used for many applications. However, The neural network employed to DA is a relatively recent application. The supervised multilayer perceptron (MLP) [10], using a back-propagation algorithm for the training phase, is used here to emulate the Kalman filter by reducing the computational effort during the DA process [16]. The best topology for the MLP-ANN is found by solving an optimization problem with cost functional:

$$f_{obj} = penalty \times \left[\frac{\rho_1 \times E_{train} + \rho_2 \times E_{gen}}{\rho_1 + \rho_2} \right] , \tag{5}$$

where the errors associated to the training and generalization evaluations are denoted by E_{train} and E_{gen}, balancing parameters for the generalization and training errors are given by: $\rho_1 = \rho_2 = 0.5$. The penalty term is a degree of neural network complexity. The procedure searches for a neural network with a fewer neurons and faster training iteration. The penalty factor is expressed by:

$$penalty = c_1 e^{(n_{neurons})^2} + c_2(n_{epochs}) + 1 \tag{6}$$

with $c_1 = 5 \times 10^8$ and $c_2 = 5 \times 10^5$ as used by Anochi and co-authors [1]. The objective function (5) is solved by a meta-heuristic called multi-particle collision algorithm (MPCA) [12]. Figure 3 shows the isovalues for q-variable in the shallow water system (1)–(3) at time-step $t = 30$, where TRUE, KF, and ANN are reference, Kalman filter, and neural network configured by MPCA (emulating Kalman filter) [16], respectively.

2.3. KF and MLP-NN: Algorithmic Complexity

Unlike in Section 2.2, the expression complexity is not linked to the number of artificial neurons or how fast the convergence is for the training phase. The *algorithmic complexity* is related to the number of arithmetic operations of an algorithm. Cintra and co-authors [5] have applied MLP-NN to reduce the Kalman filter complexity, where the NN was trained as an operator to matrix inversion. Indeed, depending on the application, the Kalman filter complexity can be reduced when the system matrix can be partitioned into smaller dimension matrices [17].

For general applications, the Kalman filter complexity has order $O(M^3)$ in terms of a number of floating-point multiplications [13] – M being the state-vector dimension, due to the matrices operations in the Kalman filter algorithms.

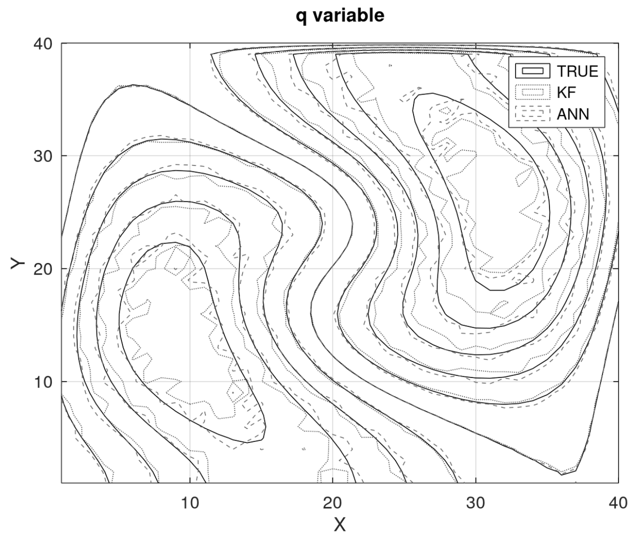


Figure 3. Data assimilation for shallow water system 2D applying Kalman filter and neural network at $t = 30$ – see also reference [16]

In the analysis for neural network algorithm, we are going to consider the $M_0 = 2 \times M_2$ inputs – with: M_2 is the number of observations and background grid points, L the number of layers (including output layer), $M_{\ell,1}$ ($\ell = 1, 2, \ldots, L-1$) the number of artificial neurons for the ℓ-th hidden layer – for simplicity, we are considering: $M_{1,1} = M_{2,1} = \ldots = M_{L-1,1} = M_1$. For each neuron, a linear combination of M inputs has the complexity:

$$s = \sum_{i=1}^{M} \theta_i \, X_i \;\sim\; O(M) \;, \tag{7}$$

where θ_i are the connection weights and X_i the inputs, respectively. The value s feeds the activation function $\varphi(s)$, representing the non-linear term in the supervised neural mapping. The activation function can be computed from a polynomial approximation by Horner's rule (*nested multiplication*):

$$\begin{aligned} \varphi(s) &= a_0 + a_1\, s + a_2\, s^2 + \ldots + a_{N-1}\, s^{N-1} + a_N\, s^N \\ &= a_0 + s(a_1 + s(a_2 + \ldots + s(a_{N-1} + a_N\, s)\ldots)) \end{aligned} \tag{8}$$

with $O(N)$ of multiplications and additions. Denoting by C_{NN} the complexity of the MLP-NN, this value can be estimated by:

$$C_{NN} \sim M_1\,[O(M_0) + O(N)] + (L-1)\,M_1\,[O(M_1) + O(N)] + M_2\,[O(M_1) + O(N)] \;. \tag{9}$$

Considering the entries of inputs and outputs, one can consider: $O(M_0) \sim O(M_1) \sim O(M_2) \sim O(M)$, where the stronger assumption is $O(M_1) \sim O(M)$. For standard applications, $L \ll M$, and $O(N) < O(M)$. Therefore, $C_{\mathrm{NN}} \sim O(M^2)$. For applications: $C_{\mathrm{KF}} = O(M^3) > O(M^2) = C_{\mathrm{NN}}$, showing that multilayer perceptron neural network has a smaller complexity than Kalman filter.

Before showing the results with parallel runs, a sequential execution was carried out for a comparison considering CPU-time between data assimilation performed by Kalman filter and neural network [9] with $N_x^{(1)} = N_y^{(1)} = 40$. Two numerical experiments were considered with 25

(Exp-1) and 100 (Exp-2) observation points. Table 1 shows the CPU-time (hr:min:sec) for two simulations with different number of observations for the data assimilation process.

Table 1. CPU-time for data assimilation using Kalman filter (KF) and multi-layer perceptron neural network (MLP-NN) with 25 (Exp-1) and 100 (Exp-2) observation points per data assimilation cycles

Expriments	KF	MLP-NN
Exp-1	00:42:02	00:01:39
Exp-2	01:19:03	00:05:01

From the results shown in Tab. 1, data assimilation process with neural network was about 25 and 15 times faster than Kalman filter for Exp-1 and Exp-2, respectively. These results for CPU-time are in agreement with results obtained with global atmospheric models. Cintra and Campos Velho did data assimilation experiments with SPEED global model (3D spectral model with very simplified physical parameterizations) [6], where the NN approach was 95 times faster than ensemble Kalman filter (EnKF). Another result with comparison with NN and EnKF with global atmospheric model was carried out with Florida State University (FSU) model. Data assimilation by the NN was 55 times faster than applying EnKF for the FSU model [7].

For parallel version implementation, the data assimilation by NN is much more effective than KF. In the analysis computed by NN, the procedure is performed at each variable for a specific grid point, combining background value with observation available. In the KF, the analysis is calculated by estimating the Kalman gain (item 3, Section 2.1), involving matrix multiplication and inversion, with another step for producing the analysis from a product between a matrix (Kalman gain) and a vector (difference between background and observation vectors) – see item 4, Section 2.1. Therefore, the parallel implementation is much more effective for NN than KF.

3. Parallel Version Strategy

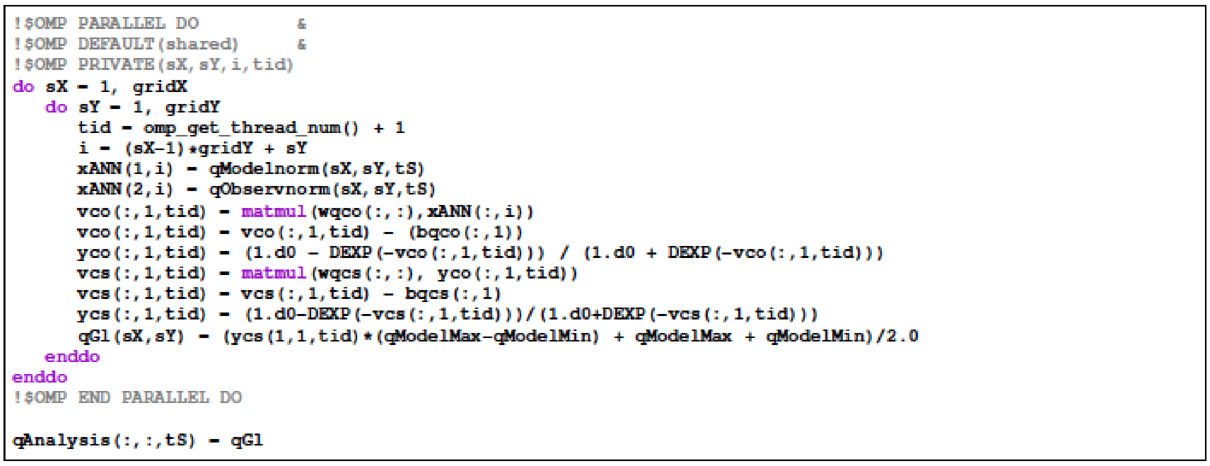
The data assimilation process described in Section 2, is summarized by algorithm SW2D_DA (Algorithm 1). For the worked example here, only assimilation for the q-variable is assimilated. The algorithm (or function) to implement the shallow-water model (SW2D_Model) is called at all N_t timesteps. In contrast, the Kalman filter data assimilation algorithm (KF_DA) or the neural networks (ANN_DA, showed in Algorithm 2) is triggered at regular intervals of timesteps (data assimilation cycles), represented by $freqObsT$, called here as the *frequency of observation.*

In this article, the Kalman filter algorithm will not be shown in detail since the focus of the work was the parallelization of data assimilation by neural networks for a space domain with a high number of grid points. In Algorithm 2, the DA is carried out independently for each grid point. Therefore, the parallel strategy is to compute the DA for each grid point in parallel. Considering N_g the number of the grid points and N_p the number of processors, the analysis is computed by a trivial parallel approach, executing N_g/N_p computation cycles for completing the DA on the entire space domain.

The loops traversing the grid points in the horizontal and vertical directions are parallelized with OpenMP directives, and the FORTRAN source code where the parallel strategy was implemented is in Fig. 4. The same approach was employed to a parallel version of the shallow-water function (SW2D_MODEL), as can be seen in Fig. 5.

```
Algorithm: SW2D_DA
input  :
         $q^{Model}$: reference SW2D model values (true)
         $q^{Observ}$: observed SW2D values (true + noise)
         $N_t$: number of timesteps
         $freqObsT$: frequency of observation
                     (defines number of assimilation cycles)
         $N_x$: number of grid points in horizontal direction
         $N_y$: number of grid points in vertical direction
         $assimType$: data assimilation type (KF or ANN)
output:
         $q^{Analysis}$: result of data assimilation
begin
    for $t \leftarrow 1$ to $N_t$ do
        SW2D_MODEL($N_x$, $N_y$, $u$, $v$, $q$)
        $q^{Analysis}_{(t)} = q$
        if $mod(t, freqObsT) = 0$ then
            switch $assimType$ do
                case 1 do
                    KF_DA($N_x$, $N_y$, $q^{Analysis}_{(t)}$)
                end
                case 2 do
                    ANN_DA($N_x$, $N_y$, $q^{Model}_{(t)}$, $q^{Observ}_{(t)}$, $q^{Analysis}_{(t)}$)
                end
            end
    end
end
```

Algorithm 1. Shallow-Water 2D Data Assimilation (SW2D_DA)



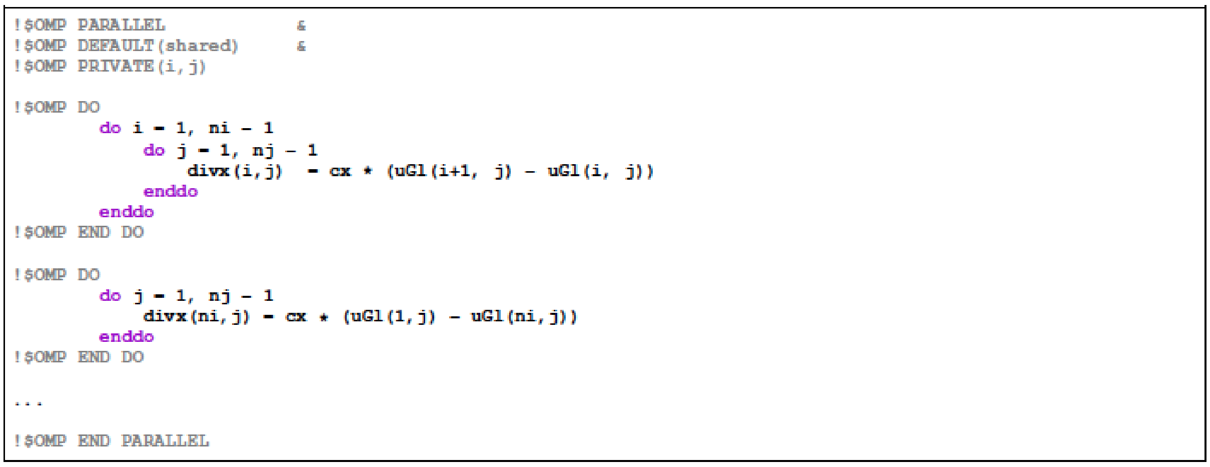
```
!$OMP PARALLEL DO          &
!$OMP DEFAULT(shared)      &
!$OMP PRIVATE(sX,sY,i,tid)
do sX = 1, gridX
   do sY = 1, gridY
      tid = omp_get_thread_num() + 1
      i = (sX-1)*gridY + sY
      xANN(1,i) = qModelnorm(sX,sY,tS)
      xANN(2,i) = qObservnorm(sX,sY,tS)
      vco(:,1,tid) = matmul(wqco(:,:),xANN(:,i))
      vco(:,1,tid) = vco(:,1,tid) - (bqco(:,1))
      yco(:,1,tid) = (1.d0 - DEXP(-vco(:,1,tid))) / (1.d0 + DEXP(-vco(:,1,tid)))
      vcs(:,1,tid) = matmul(wqcs(:,:), yco(:,1,tid))
      vcs(:,1,tid) = vcs(:,1,tid) - bqcs(:,1)
      ycs(:,1,tid) = (1.d0-DEXP(-vcs(:,1,tid)))/(1.d0+DEXP(-vcs(:,1,tid)))
      qGl(sX,sY) = (ycs(1,1,tid)*(qModelMax-qModelMin) + qModelMax + qModelMin)/2.0
   enddo
enddo
!$OMP END PARALLEL DO

qAnalysis(:,:,tS) = qGl
```

Figure 4. Parallel OpenMP Fortran code for the Algorithm-2

```
Algorithm: ANN_DA
input :
        $q^{Model}$: reference SW2D model values (true)
        $q^{Observ}$: observed SW2D values (true + noise)
        $N_x$: number of grid points in horizontal direction
        $N_y$: number of grid points in vertical direction
output:
        $q^{Analysis}$: result of data assimilation
begin
    for $i \leftarrow 1$ to $N_x$ do
        for $j \leftarrow 1$ to $N_y$ do
            $v_{1,2}(i,j) = \sum_{l=1}^{\#\text{neurons}} \left[ w_{1l}(i,j)\, q^{Model} + w_{2l}(i,j)\, q^{Observ}(i,j) + b(i,j) \right]$
            $q^{Analysis}(i,j) = \tanh[v_1(i,j)] + \tanh[v_2(i,j)]$
        end
    end
end
```

Algorithm 2. Artificial Neural Network Data Assimilation (ANN_DA)



```
!$OMP PARALLEL              &
!$OMP DEFAULT(shared)       &
!$OMP PRIVATE(i,j)

!$OMP DO
        do i = 1, ni - 1
            do j = 1, nj - 1
                divx(i,j)  = cx * (uGl(i+1,  j) - uGl(i,  j))
            enddo
        enddo
!$OMP END DO

!$OMP DO
        do j = 1, nj - 1
            divx(ni,j) = cx * (uGl(1,j) - uGl(ni,j))
        enddo
!$OMP END DO

...

!$OMP END PARALLEL
```

Figure 5. Parallel OpenMP Fortran code of shallow-water 2D model

4. Results

The shallow-water system was defined by considering the ocean circulation. The numerical values for the parameters are shown in Tab. 2, with $t_{\max} = N_t\,\Delta t$, the spatial domain discretization given by Δx and Δy, and N_x and N_y are, respectively, the number of grid points in horizontal and vertical directions, and the upper indexes $^{(1)}$ and $^{(2)}$ are related to the 40-point and 2560-point grid sizes. Finally, the data assimilation cycle (the frequency of observation in Algorithm 2) is performed at each 10 time-steps ($freqObsT = 10$).

The executions were made in one compute node of the Santos Dumont supercomputer (an ATOS machine). The computer node has two CPU Intel Xeon E5-2695v2 with 48 cores and 384 Gigabytes of RAM. Initially, for serial performance comparison purposes between the original assimilation method with Kalman Filter and the method with neural networks, a 40-point grid size was used. According to results from Furtado and co-authors [9], the KF method

Table 2. Parameters used in the integration for the SW-model

Parameter	Value	Parameter	Value
Δt (h)	180	r_u (s^{-1})	1.8×10^4
N_t	200	r_v (s^{-1})	1.8×10^4
$t_{\max}$ (h)	3.6×10^4	r_q (s^{-1})	1.8×10^4
Δx (km)	10^5	ρ_a (kg/m^{-3})	1.275
Δy (km)	10^5	ρ_w (kg/m^{-3})	1.0×10^3
$N_x^{(1)}$	40	C_d	1.6×10^{-3}
$N_y^{(1)}$	40	H (m)	5000
$N_x^{(2)}$	2560	g (m/s^{-2})	9.806
$N_y^{(2)}$	2560	f (s^{-1})	1.0×10^{-4}

is much more computing expensive than the ANN method. In addition to the ANN method being considerably faster, the final result obtained is relatively close to that of KF, and also to the reference solution (TRUE) for the q shallow-water variable at grid position $(8, 8)$, as can be seen in Fig. 6a. Similar comparisons between KF and ANN methods have already previously been done – see references [4, 9, 16].

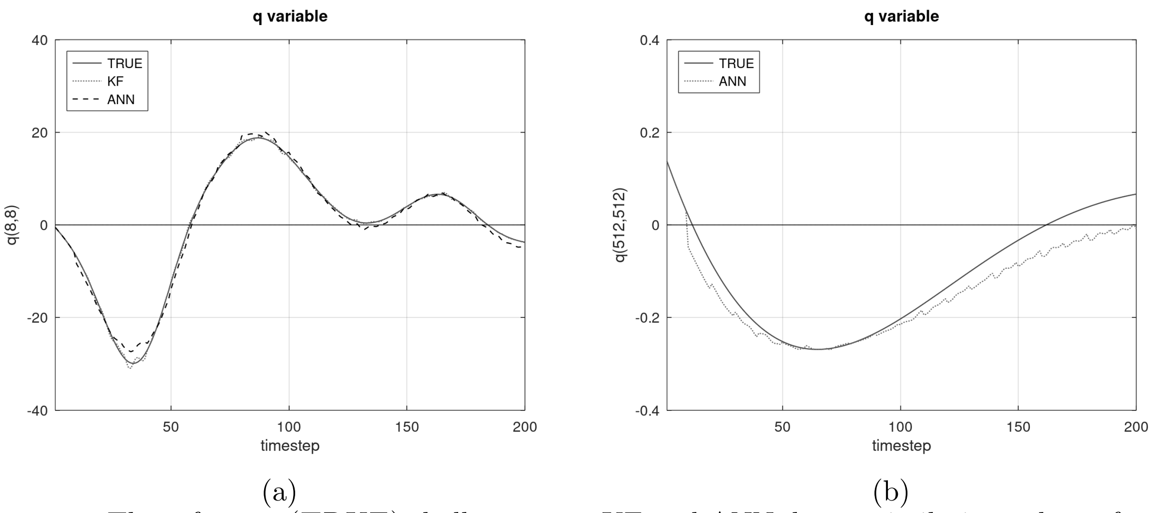


Figure 6. The reference (TRUE) shallow-water, KF and ANN data assimilation values of variable q at grid position $(8, 8)$ for 40-point grid size (a), and at grid position $(512, 512)$ for 2560-point grid size (b), using weights and bias obtained for the 40-point grid size neural network

The evaluation of parallel performance of the ANN assimilation method for a computational problem with a high number of grid-points was tested. The number of grid-points used for this purpose was one with 2560 points in the horizontal and vertical coordinates, i.e., $N_x^{(2)} = N_y^{(2)} = 2560$. For this grid size, it was unfeasible to obtain the KF data assimilation result, with the actual source code. Figure 6 shows the comparison result only between the reference q values (TRUE) and the ANN assimilation result (ANN_DA) at grid position $(512, 512)$, but using the same neural network employed for the 40-point grid size. As mentioned, the neural network for the finer resolution problem is the same of that configured to emulate the Kalman filter with the a coarser computational mesh. Even so, we can observe the neural network dynamics close to the curve of the true solution – see Fig. 6b.

The serial execution profiling of the Fortran implementation of Shallow-Water 2D Data Assimilation algorithm (SW2D_DA, in Algorithm 1) is shown in Tab. 3. The biggest hotspot is the function SW2D_Model, which integrates the 2D shallow-water model in all 200 timesteps. The second hotspot is the ANN_DA function, which emulates data assimilation obtained by Kalman filter using an artificial neural network. Important to note that this function is activated only at the end of each 10-timesteps cycle. Therefore, it is called in only 20 times from a total of 200 timesteps.

Table 3. Serial performance profiling

Function	Time (s)	Time share (%)
SW2D_Model	217.1	74.7
ANN_DA	41.2	14.2
Others	32.3	11.1
Total time	290.6	100.0

The parallel performance of the functions SW2D_DA and ANN_DA, obtained using up to 32 OpenMP threads, is presented in Tab. 4. A reduction about ten times from the serial time was achieved in the first function (SW2D_Model), while a less significant reduction was observed in the second function (ANN_DA).

The processing time reduction in the shallow-water function SW2D_DA results from the good parallel efficiency achieved, especially with up to 16 threads. Using 32 OpenMP threads, the runtime reduces from 217.1 seconds to 34.7 seconds. However, we believe the speed-up obtained with 32 threads could be even better.

In order to have a better understanding for the results with 16-threads and 32-threads, the number of grid points were increased to $N_x = N_y = 4000$ (in Tab. 5), just to verify if there is a saturation for processing demand, i.e., enhancing the computational load, a better speed-up should be obtained. However, we got a worse performance than 2500 grid points. Therefore, such behavior from the results show other issues are acting. Probably the performance results are linked to the cache misses and/or synchronization among the processing cores. Further investigation is needed to improve the parallel efficiency with this number of threads.

The parallel performance obtained with the ANN_DA function was better than 2D shallow-water function. Using 32 OpenMP threads, the runtime for ANN_DA function is reduced from 41.2 seconds to 2.1 seconds for the 2560-grid size, obtaining a speed-up of almost 20 times concerning the serial execution, according to values presented in Tab. 4. A similar speed-up was also reached for the 4000-grid size, shown in Tab. 5. In this case, the runtime is reduced from 91.9 seconds to 5.0 seconds, obtaining a speed-up about 18 times.

After the parallelization performed in the two main hotspots, the remaining code (Others in Tab. 3), not listed here, are instructions used to prepare the memory for SW2D_Model and ANN_DA routines. Since it amounts to 11.1% of the total time, improving these functions' performance is not mandatory in future developments.

Table 4. Parallel performance of shallow-water 2D model and ANN assimilation for 2560-grid points in both X and Y coordinates

SW2D_Model				ANN_DA			
#threads	Time (s)	Speed-up	Eff	#threads	Time(s)	Speed-up	Eff
1	217.1	1.0	1.00	1	41.2	1.0	1.00
2	119.8	1.8	0.91	2	21.4	1.9	0.96
4	70.0	3.1	0.78	4	11.2	3.7	0.92
8	45.6	4.8	0.60	8	6.0	6.9	0.86
16	31.1	7.0	0.44	16	3.1	13.3	0.83
32	34.7	6.3	0.20	32	2.1	19.6	0.61

Table 5. Parallel performance of shallow-water 2D model and ANN assimilation for 4000-grid points in both X and Y coordinates

SW2D_Model				ANN_DA			
#threads	Time (s)	Speed-up	Eff	#threads	Time(s)	Speed-up	Eff
1	555.5	1.0	1.00	1	91.9	1.0	1.00
2	312.5	1.8	0.89	2	49.7	1.8	0.92
4	195.8	2.8	0.71	4	28.0	3.3	0.82
8	134.3	4.1	0.52	8	15.4	6.0	0.75
16	114.6	4.8	0.30	16	8.1	11.3	0.71
32	155.5	3.6	0.11	32	5.0	18.4	0.57

5. Final Remarks

The parallel processing techniques were applied to reduce data assimilation processing time with neural networks for domains with an increased grid points density, presenting a more significant speeding-up. However, a deeper study must be carried out to obtain a better parallel efficiency of the function implemented to the shallow-water 2D algorithm. A preliminary strategy got implemented using OpenMP. Thus, one way to improve parallel performance can be through a better choice of thread scheduling. Looking at a higher level of parallelism, one can also use MPI to execute the code in a distributed memory machine, a cluster p.ex., through a subdivision of the spatial domain. In this case, one can even run costly instances of the shallow-water problem, using a grid containing an even more significant number of points.

Acknowledgements

The authors acknowledge the National Laboratory for Scientific Computing (LNCC/MCTI, Brazil) for providing HPC resources of the SDumont supercomputer, which have contributed to the research results reported within this paper. URL: `http://sdumont.lncc.br`. The authors would also like to thank the Brazilian agencies for their research support. Author HFCV thanks the National Council for Scientific and Technological Development (CNPq, Portuguese) for the research grant (CNPq: 312924/2017-8).

References

1. Anochi, J.A., Campos Velho, H.F., Hernandez Torres, R.: Two geoscience applications by optimal neural network architecture. Pure and Applied Geophysics 1776(1), 1–21 (2019). `https://doi.org/10.1007/s00024-019-02386-y`
2. Bennett, A.F.: Inverse Modeling of The Ocean and Atmosphere. Cambridge University Press (2002)
3. Boucher, M.A., Quilty, J., Adamowski, J.: Data assimilation for streamflow forecasting using extreme learning machines and multilayer perceptrons. Water Resources Research 56 (2020). `https://doi.org/10.1029/2019WR026226`
4. Campos Velho, H.F., Härter, F.P., Rempel, E.L., Chian, A.: Neural networks in auroral data assimilation. Journal of Atmospheric and Solar-Terrestrial Physics 70(10), 1243–1250 (2008). `https://doi.org/10.1016/j.jastp.2008.03.018`
5. Cintra, R.C., Campos Velho, H.F., Todling, R.: Redes neurais artificiais na melhoria de desempenho de métodos de assimilação de dados: filtro de Kalman. TEMA: Trends in Computational and Applied Mathematics 11(1), 29–39 (2010). `https://doi.org/10.5540/tema.2010.011.01.0029`
6. Cintra, R.S.C., Campos Velho, H.F.: Data assimilation by artificial neural networks for an atmospheric general circulation model. In: El-Shahat, A. (ed.) Advanced Applications for Artificial Neural Networks, chap. 14, pp. 265–285. Intech (2018). `https://doi.org/10.5772/intechopen.70791`
7. Cintra, R.S.C., Campos Velho, H.F., Cocke, S.: Tracking the model: data assimilation by artificial neural network. In: IEEE International Joint Conference on Neural Networks – IJCNN, Vancouver, Canada, July 24-29, 2016. vol. 4, pp. 403–410 (2016). `https://doi.org/10.1109/IJCNN.2016.7727227`
8. Daley, R.: Atmospheric Data Analysis. Cambridge University Press (1993)
9. Furtado, H.C., Cintra, R.S.C., Campos Velho, H.F., et al.: Neural network for data assimilation method applied to shallow water equation. In: 2nd International Symposium on Uncertainty Quantification and Stochastic Modeling, Rouen, France, July 7-11, 2014. pp. 299–311 (2014)
10. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice Hall Inc. (1994)
11. Kalnay, E.: Atmospheric Modeling, Data Assimilation, and Predictability. Cambridge University Press (2003)
12. Luz, E.F.P., Becceneri, J.C., de Campos Velho, H.F.: A new multi-particle collision algorithm for optimization in a high performance environment. Journal of Computational Interdisciplinary Sciences 1(1), 3–10 (2008). `https://doi.org/10.6062/jcis.2008.01.01.0001`

13. Mendel, J.: Computational requirements for a discrete Kalman filter. IEEE Transactions on Automatic Control 16(6), 748–758 (1971). https://doi.org/10.1109/TAC.1971.1099837

14. Mesinger, F., Arakawa, A.: Numerical methods used in atmospheric models. Global Atmospheric Research Program – World Meteorological Organization (1976)

15. Reich, S., Cotter, C.: Probabilistic Forecasting and Baysian Data Assimilation. Cambridge University Press (2015)

16. Sambatti, S.B.M., Campos Velho, H.F., Furtado, H.C.M., et al.: Self-configured neural network for data assimilation using FPGA for ocean circulation. In: 3Conference of Computational Interdisciplinary Science (CCIS 2016), São José dos Campos (SP), Brazil (2016)

17. Vaidehi, V., Krishnan, C.N.: Computational complexity of the Kalman tracking algorithm. IETE Journal of Researcht 44(3), 125–134 (1998). https://doi.org/10.1080/03772063.1998.11416038