Parallel execution of an artificial neural network for data assimilation of the shallow-water 2D problem

Haroldo Fraga de Campos Velho[1], Helaine Cristina Moraes Furtado[2], Sabrina Bergoch Monteiro Sambatti[3], Carla Osthoff Ferreira de Barros[4], Maria Eugênia Sausen Welter[4], Roberto Pinto Souto[4], Diego Carvalho[5], Douglas O. Cardoso[6]

[1]*National Institute for Space Research (INPE)*
Av. dos Astronautas, 1758, 12227-010, São José dos Campos, SP, Brazil
haroldo.camposvelho@inpe.br
[2]*Federal University of Western Pará (UFOPA)*
Rua Vera Paz, s/n – Salé, 68040-470, Santarém, PA, Brazil
helaine.furtado@ufopa.edu.br
[3]*Independent researcher*
Av. São João, 748, 12242-840, São José dos Campos, SP, Brazil
sabrinabms@gmail.com
[4]*National Laboraty for Scientific Computing (LNCC)*
Av. Getulio Vargas, 333 – Quitandinha, 25651-076, Petrópolis, RJ, Brazil
osthoff@lncc.br, marowelter@gmail.com, rpsouto@lncc.br
[5] *Federal Center for Technological Education Celso Suckow da Fonseca (CEFET/RJ)*
Av. Maracanã, 229 – Maracanã, 20271-110, Rio de Janeiro, RJ, Brazil
d.carvalho@ieee.org
[6] *Federal Center for Technological Education Celso Suckow da Fonseca (CEFET/RJ) - Petrópolis Campus*
Rua do Imperador, 971 – Centro, 25620-003, Petrópolis, RJ, Brazil
douglas.cardoso@cefet-rj.br

Abstract. There will always be some error with reality in computational modeling of physical phenomena, even for the most advanced and sophisticated ones. Techniques that incorporate information from the phenomenon's observational data can be applied to reduce this error's uncertainty. These so-called data assimilation techniques add information from observational data to the modeling result with a reasonable degree of reliability. The Kalman Filter is one of the most widely used data assimilation methods in the operational weather forecast to better estimate the next forecast cycle's initial conditions (analysis). This work uses data assimilation through artificial neural networks, applied to the shallow-water model in two dimensions to emulate the Kalman Filter techniques, using synthetic observations. According to results obtained in previous works, this method presents a significant reduction in the processing time, maintaining an equivalent quality of the analyzes obtained through the Kalman Filter. However, even with this reduction in computational cost, when the spatial domain is discretized by a grid containing many points, the data assimilation by the neural network can still be configured as one of the performance bottlenecks. Since the assimilation by neural networks is carried out independently at each grid point, the parallel strategy employed consists of sub-dividing the domain to execute each in different computational nodes or cores.

Keywords: Data assimilation, artificial neural network, parallel processing.

1 Introduction

One essential issue for the operational prediction centers is to compute the best initial condition for the next forecasting cycle. The initial conditional is computed by a data fusion between a previous prediction – *background fields* – and observation data. The produced data fusion is called *analysis*. This data fusion procedure has started with the numerical weather prediction, and the process is known as *data assimilation* (DA). Several methods have been developed to carry out the DA. Among these methods can be cited optimal interpolation, Kalman filter, variational method, and particle filter [1–3].

However, all mentioned DA techniques present intensive computation. For reducing the computational effort for the DA, a supervised artificial neural network is designed to emulate the applied assimilation method. The multi-layer perceptron neural network with a back-propagation learning algorithm is employed to emulate a Kalman filter.

Neural network has been successfully used for data assimilation for geophysical models with application on meteorological models [4, 5], hydrology [6], and space weather application [7]. Here, the DA by the neural network is applied to the shallow-water equation designed to represent ocean circulation dynamics as described in Bennett's book [8]. The neural network is configured to emulate the Kalman filter – see reference [9]. In order to enhance the computational performance for neural data assimilation, a parallel version of the DA procedure is developed in this paper.

The next section describes the mathematical model used as a prediction system. Data assimilation methods – Kalman filter and neural network – are presented in Section 3. The strategy for parallel implementation is commented in Section 4. Results for speed-up and efficiency are shown in Section 5. The last section addresses conclusions and final remarks.

2 Shallow-water equations as a model for ocean circulation

Assuming that the vertical dimension is much smaller than the horizontal dimension, the shallow-water system can be applied. The system of equations can be derived by vertical integration of the Navier–Stokes equations. The equations are solved by a numerical approximation, producing fields of oceanic circulation. Two-dimensional linearized wave equation has the fluid depth (H) and the two-dimensional fluid velocity field (u and v) as independent variables, where the gravitational force (g) is the only force acting on the fluid. Following the model described by Bennett's book [8], the mathematical model equations are expressed as:

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

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

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

where $(x, y) \in (0, L_x) \times (0, L_y)$ and $t > 0$, f is the Coriolis parameter, $(r_u,\ r_v,\ r_q)$ are the damping coefficients, (u, v) are velocity components, F_u and F_v are external forcing, H is the mean depth of the ocean, and q is the free surface disturbance. Boundary conditions are shown in Figure 1.

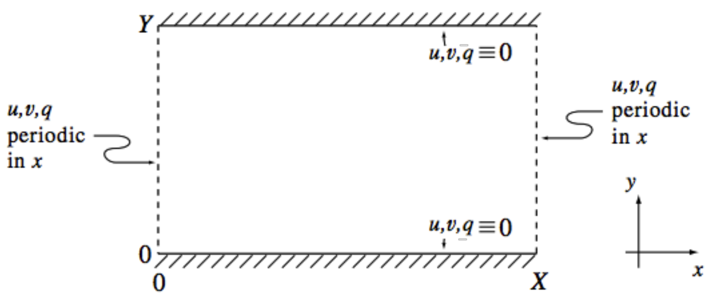


Figure 1. Boundary conditions for the equations (1)–(3).

The finite difference is the method for space discretization, and the forward-backward method was used for time integration [10]. The Arakawa grid-C is adopted for spatial discretization – see Figure 2.

3 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.

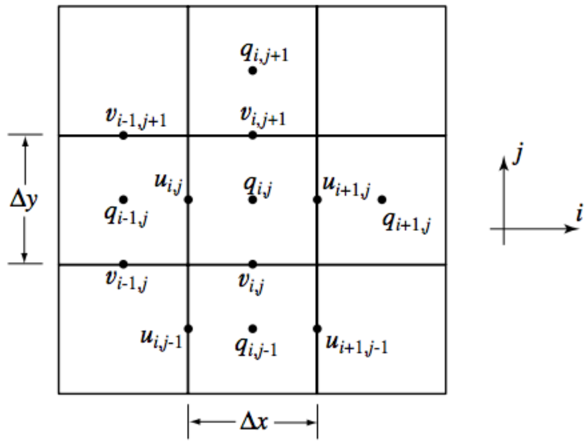


Figure 2. The Arakawa grid-C used for space discretization.

3.1 Kalman filter

A Kalman filter provides a recursive solution to the linear optimal filtering problem. It is an optimal estimator, infers parameters of interest from indirect, inaccurate and uncertain observations. It is recursive so that new measurements can be processed as they arrive. Consider the nonlinear stochastic discrete-time dynamical system:

$$x^f_{t+1} = M_t x^f_t + \mu_t$$
$$y^f_t = H_t x^f_t + \nu_t$$

where M_t is a mathematical model, μ_t is the model error, H_t is the mesurement function, ν_t is the white-noise sequence associetaded to observations. Under the specified hypotheses the optimal way, in the least squares sense, to assimilate sequentially the observations is given by the Kalman filter defined by recurrence over the observation times:

1. Forecast model for state vector (M_t is the matrix of the system):

$$x^f_{t+1} = M\{x^a_t\} \approx M_t\, x^a_t$$

2. Update the forecasting covariance matrix ($W^{\rm mod}$ is the modeling covariance error matrix):

$$P^f_{t+1} = M_t P^a_t M^T_t + W^{\rm mod}_t$$

3. Compute the Kalman gain ($W^{\rm obs}$ is the measurement covariance error matrix):

$$K_{t+1} = P^f_{t+1} H^T_{t+1} [W^{\rm obs}_{t+1} + H_{t+1} P^f_{t+1} H^T_{t+1}]^{-1}$$

4. Compute the analysis (DA) ($x^{\rm obs}$ is the state observation vector):

$$x^a_{t+1} = x^f_{t+1} + K_{t+1}[x^{\rm obs}_{t+1} - (H_{t+1} x^f_{t+1})]$$

5. Update the analysis covariance:

$$P^a_{t+1} = [I - K_{t+1} H_{t+1}] P^f_{t+1}\ .$$

The dimension of the matrix M_t is related to number of points used for the spatial discretization. In addition to, the dynamic model matrix M_t we have the error covariance matrices: modeling ($W^{\rm mod}_t$) and observation ($W^{\rm obs}_t$), which are updated at each time step. The matrix operations, such as multiplication, inverse matrix calculation, are carried out to compute the Kalman gain matrix, yielding a very costly method. These difficulties motivate an investigation of ANN tool as a method of data assimilation, with obtained gain reaching up to 30 times faster for the 2D shallow-water problem [9]. In the next section we describe the Artificial Neural Networking as a data assimilation technique.

3.2 Artificial neural network (ANN)

Nowadays, there are multiple applications for artificial neural networks (ANN). The use of the neural network for DA is relatively new. The supervised multilayer perceptron (MLP) [11] with a back-propagation algorithm for the learning phase was employed to substitute the Kalman filter by enhancing the computational performance for DA [12]. The best configuration of the MLP-ANN is computed by minimizing the following objective function:

$$f_{obj} = penalty \times \left(\frac{\rho_1 \times E_{train} + \rho_2 \times E_{gen}}{\rho_1 + \rho_2} \right) \quad (4)$$

where E_{train} and E_{gen} are errors during the training and generalization phases, ρ_1 and ρ_2 are parameters for balancing the generalization and training errors – here: $\rho_1 = \rho_2 = 0.5$ is used. Penalty factor indicates searching for an ANN with the smallest number of neurons and faster learning convergence. The penalty term evaluates the ANN complexity, and it is given by:

$$penalty = c_1 e^{(n_{neurons})^2} + c_2(n_{epochs}) + 1 \quad (5)$$

where $c_1 = 5 \times 10^8$ and $c_2 = 5 \times 10^5$ are the parameters to computed the ANN complexity [13]. The cost function (4) is minimized by the multi-particle collision algorithm (MPCA) [14].

4 Parallel version strategy

The data assimilation process described in the Section 3, is summarized by algorithm SW2D_DA (**Algorithm** 1). For the worked example here, only assimilation for the q-vriable is assmilated. Note that 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 the **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 Listing 1. The same approach was employed to a parallel version of the shallow-water function (SW2D_MODEL), as can be seen in Listing 2.

```
!$OMP PARALLEL DO          &
!$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
```

Listing 1. Parallel OpenMP Fortran code for the **Algorithm 2**

```
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 ← 1 to N_t do
        SW2D_MODEL(N_x, N_y, q_(t))
        if mod(t, freqObsT) = 0 then
            switch assimType do
                case KF do
                    KF_DA(N_x, N_y, q_(t)^Analysis)
                end
                case ANN do
                    ANN_DA(N_x, N_y, q_(t)^Model, q_(t)^Observ, q_(t)^Analysis)
                end
            end
        q_(t+1) = q_(t)^Analysis
    end
end
```

Algorithm 1: Sallow-Water 2D Data Assimilation (SW2D_DA)

```
Algorithm: ANN_DA
input :
        N_x: number of grid points in horizontal direction
        N_y: number of grid points in vertical direction
        q^Model: reference SW2D model values (true)
        q^Observ: observed SW2D values (true + noise)
output:
        q^Analysis: result of data assimilation
begin
    for i ← 1 to N_x do
        for j ← 1 to N_y do
```

$$v_{1,2}(i,j) = \sum_{l=1}^{\#\text{neurons}} \Big[w_{1l}(i,j)\, q^{Model} + w_{2l}(i,j)\, q^{Observ}(i,j) + b(i,j) \Big]$$

$$q^{Analysis}(i,j) = \tanh[v_1(i,j)] + \tanh[v_2(i,j)]$$

```
        end
    end
end
```

Algorithm 2: Artificial Neural Network Data Assimilation (ANN_DA) algorithm, where $w_{1l}(i,j)$ are the connection weights and $b(i,j)$ is a threshold parameter.

5 Results

The shallow-water system was defined by considering the ocean circulation, and the numerical values for the parameters are shown in Table 1, 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, i.e., $N_x^{(1)} = N_y^{(1)} = 40$. According to runtimes in Table 2, the KF method is much more computing expensive than the ANN method, nearly six orders of magnitude

```
!$OMP PARALLEL             &
!$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
```

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

Table 1. Parameters used in the integration for the SW-model.

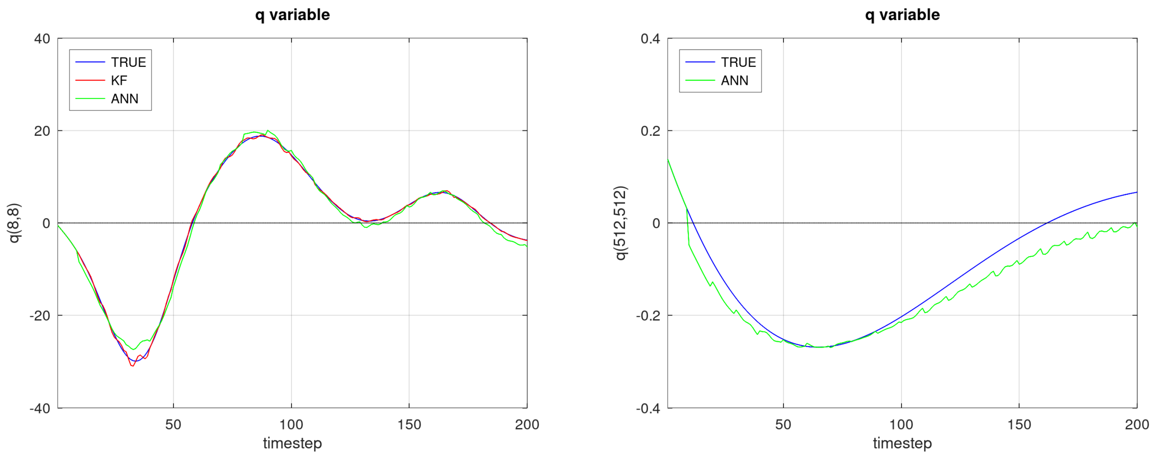
Parameter	Value	Parameter	Value
$\Delta t\ (h)$	180	$r_u\ (s^{-1})$	1.8×10^4
N_t	200	$r_v\ (s^{-1})$	1.8×10^4
$t_{\max}\ (h)$	3.6×10^4	$r_q\ (s^{-1})$	1.8×10^4
$\Delta x\ (km)$	10^5	$\rho_a\ (kg/m^{-3})$	1.275
$\Delta y\ (km)$	10^5	$\rho_w\ (kg/m^{-3})$	1.0×10^3
$N_x^{(1)}$	40	C_d	1.6×10^{-3}
$N_y^{(1)}$	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}

higher for this test case. 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 Figure 3a. Particularly, at this grid position, the normalized root-mean-square error (NRMSE) related to reference value is NRMSE$_{KF}$=0.0065 for KF and NRMSE$_{ANN}$=0.0222 for ANN. And for the total 25 observations points in the 5×5 grid we have NRMSE$_{KF}$=0.0034 for KF and NRMSE$_{ANN}$=0.0105 for ANN. Similar comparisons between KF and ANN methods has already previously been done – see references [7, 9, 12].

Table 2. Serial performance comparison between KF and ANN methods for the 40-point grid size.

Algorithm	Time (s)
KF_DA	$3.33 \times 10^{+3}$
ANN_DA	8.13×10^{-3}

The contribution of this work refers to the study of the parallel performance of the ANN assimilation method for a computational problem with a high number of grid-points. The number of grid-points used for this purpose was one with 2560 points in the horizontal and vertical directions, 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 3 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,



(a) (b)

Figure 3. 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 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 Figure 3b. At this grid point, the normalized root-mean-square error (NRMSE) related to reference value is NRMSE$_{ANN}$=0.1008 for ANN. And for the total 25 observations points in the 5×5 grid we have NRMSE$_{ANN}$=0.0196 for ANN.

The serial execution profiling of the Fortran implementation of Sallow-Water 2D Data Assimilation algorithm (SW2D_DA, in **Algorithm** 1) is shown in Table 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	240.3	75.4
ANN_DA	35.5	11.1
OTHERS	43.0	13.5
Total time	318.8	100.0

The parallel performance of the functions SW2D_DA and ANN_DA, obtained using up to 32 OpenMP threads, is presented in Table 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 240.3 seconds to 23.9 seconds. However, we believe the speed-up obtained with 32 threads could be even better. Further investigation is needed to improve the parallel efficiency with this number of threads.

The parallel performance of the ANN_DA function was not the same as the 2D shallow-water function, mainly due to the unusual behavior with two threads, where the processing time is higher than that with only one thread. Using 32 OpenMP threads, the runtime is reduced to 10.8 seconds, obtaining a speed-up of about 3 times in relation to the serial execution. We do not know how to explain this result, which will require further investigation.

After the parallelization performed in the two main hotspots, the remaining code functions (OTHERS in Table 3), not listed here, spent now the most of the processing time. Therefore, it is also important to improve the performance of these functions in future developments.

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

SW2D_MODEL				ANN_DA			
#threads	Time (s)	Speed-up	Eff	#threads	Time(s)	Speed-up	Eff
1	240.3	1.00	1.00	1	35.5	1.00	1.00
2	125.7	1.91	0.96	2	53.6	0.66	0.33
4	67.2	3.58	0.89	4	39.8	0.89	0.22
8	38.7	6.21	0.78	8	26.8	1.32	0.17
16	25.9	9.28	0.58	16	18.5	1.92	0.12
32	23.9	10.05	0.31	32	10.8	3.29	0.10

6 Conclusions and final remarks

The parallel processing techniques was applied to reduce the processing time of data assimilation with neural networks for domains containing a high number of grid points, presenting a greater than 10 speeding-up. However, a deeper study must be carried out in order to obtain a better parallel efficiency of the functions implemented to the shallow-water 2D algorithms and data assimilation by neural networks. An initial strategy was implemented using OpenMP. Thus, one way to try 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 larger instances of the shallow-water problem, using a grid containing an even larger 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).

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] R. Daley. *Atmospheric Data Analysis*. Cambridge University Press, 1993.
[2] E. Kalnay. *Atmospheric Modeling, Data Assimilation, and Predictability*. Cambridge University Press, 2003.
[3] S. Reich and C. Cotter. *Probabilistic Forecasting and Baysian Data Assimilation*. Cambridge University Press, 2015.
[4] R. S. C. Cintra, H. F. Campos Velho, and S. Cocke. Tracking the model: data assimilation by artificial neural network. In *IEEE International Joint Conference on Neural Networks – IJCNN*, volume 4, pp. 403–410, Vancouver, Canada, 2016.
[5] R. S. C. Cintra and H. F. Campos Velho. Data assimilation by artificial neural networks for an atmospheric general circulation model. In A. El-Shahat, ed, *Advanced Applications for Artificial Neural Networks*, chapter 14, pp. 265–285. Intech, 2018.
[6] M. Boucher, J. Quilty, and J. Adamowski. Data assimilation for streamflow forecasting using extreme learning machines and multilayer perceptrons. *Pater Resources Research*, vol. 56, n. 1, pp. 1–23, 2020.
[7] H. F. Campos Velho, F. P. Härter, E. L. Rempel, and A. Chian. Neural networks in auroral data assimilation. *Journal of Atmospheric and Solar-Terrestrial Physics*, vol. 70, n. 10, pp. 1243–1250, 2008.
[8] A. F. Bennett. *Inverse Modeling of The Ocean and Atmosphere*. Cambridge University Press, 2002.
[9] H. C. Furtado, R. S. C. Cintra, H. F. Campos Velho, L. D. Chiwiacowsky, and E. E. N. Macau. Neural network for data assimilation method applied to shallow water equation. In *2nd International Symposium on Uncertainty*

Quantification and Stochastic Modeling, pp. 299–311, Rouen, France, 2014.

[10] F. Mesinger and A. Arakawa. *Numerical methods used in atmospheric models*. Global Atmospheric Research Program – World Meteorological Organization, 1976.

[11] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall Inc., 1994.

[12] S. B. M. Sambatti, H. F. Campos Velho, H. C. M. Furtado, V. C. Gomes, and A. S. Charão. 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.

[13] J. A. Anochi, H. F. Campos Velho, and R. Hernandez Torres. Two geoscience applications by optimal neural network architecture. *Pure and Applied Geophysics*, vol. 1776, n. 1, pp. 1–21, 2019.

[14] E. F. P. Luz, J. C. Becceneri, and H. F. Campos Velho. Two geoscience applications by optimal neural network architecture. *A new multi-particle collision algorithm for optimization in a high performance environment*, vol. 1, n. 1, pp. 3–10, 2008.