Chapter

Neutral Network Adaptive Filter with Application to Ocean Current Energy Estimation

Hong Son Hoang and Remy Baraille

Abstract

This chapter proposes a new approach for the design of an adaptive filter (AF), which is based on an artificial neural network (NN) structure for estimating the system state. The NNs are now widely used as a technology offering a way to solve complex and nonlinear problems such as time-series forecasting, process control, parameter state estimation, and fault diagnosis. The proposed NN-based adaptive filtering (NNAF) is designed by considering the filtering algorithm as an input–output system and two-stage optimization procedure. The first concerns a learning process where the weights of the NNAF are estimated to minimize the error between the filtered state and the state samples generated by a numerical model. The adaptation is carried out next to minimize the mean prediction error (MPE) of the system outputs (error between the observations and the system output forecast) subject to the coefficients associated with the estimated NN weights. Simulation results for different numerical models, especially for state estimation of the chaotic Lorenz system as well as for the ocean current at different deep layers which is important for renewable energy device placements, are presented to show the efficiency of the NNAF.

Keywords: neural network, Kalman filter, adaptive filter, machine learning, minimum prediction error approach, ocean current energy

1. Introduction

In this chapter, we develop an NNAF for solving filtering problems, especially for dynamic systems of high dimension (an order of 10^7–10^8).

Optimal filtering originated from the astronomical studies at the end of the eighteenth century. It is worth of mentioning the early works on the filtering [1, 2], where the theory of linear minimum-variance filters is developed independently. In Ref. [1], the filter is developed to estimate the desired value of a noisy signal, whereas the filter in Ref. [2] minimizes the mean squared error between the estimated random process and the desired process. These two works constitute a foundation for the celebrated Kalman filter (KF) [3], which processes the measurements streaming in continuously and updates the estimate of the signal recursively upon the arrival of each measurement.

Under the condition that the system state and observation processes satisfy linear equations driven by white Gaussian noises, the KF produces the unbiased estimate with minimum error variance. Since in most applications, the dynamic and observation processes are nonlinear, and a linearization technique is applied, which results in an extended Kalman filter (EKF). However, the EKF is sometimes far from optimal and even divergent.

The KF has received a great interest for many industrial and engineering fields since the 1970s, such as the system control, signal processing, robotics, seismology, communication, economics, and finance.

Direct application of the KF to the data assimilation (DA) in geophysical systems such as meteorological or oceanic systems is very difficult, even impossible due to high computational effort and memory storage required to deal with very high system dimensions, not to say on nonlinearities and uncertainties in statistics of the model and observation errors.

This situation forces researchers to search new, simple but effective algorithms to overcome these difficulties. We find here the approaches more appropriate for solving DA problems in high dimensional systems (HdSs). For example, the 4D-Var algorithm [4] adjusts a forecast, in space and time, to bring it into closer agreement with observations. Based on the KF version, the EnKF [5] uses the covariance estimated from samples instead of solving the Algebraic Riccati matrix equation. The EnKF is related to the particle filter (a particle is the same thing as ensemble member). Particle filtering [6] uses a set of particles (also called samples) to represent the posterior distribution of some stochastic process given noisy and/or partial observations. The state-space model can be nonlinear, and the initial state and noise distributions can be non-Gaussian. These approaches are proposed with strategies to better control the uncertainty and to make it less non-Gaussian: targeting of observations, specific treatment of highly nonlinear degrees of freedom, localization, and model reduction. A review of the application and usefulness of 4D-Var, ensemble KF, and particle filters in geosciences is given in Ref. [7]. Among the efficient approaches for solving DA in high dimensional setting, we note the adaptive filtering (AF, see Section 2.2). The idea of the AF will be exploited in this chapter to improve the neural network filtering (NNF) algorithms.

To make more clear on why the AF is capable of dealing with the difficulties to assimilate data into the HdSs, it is useful to list the main differences between the KF and the AF as follows:

1. The KF is aimed at seeking an optimal estimate in the space of measurable Borel functions of the system state. As for the AF, it seeks the estimate, optimal in the class of stable filters of a given structure.

2. Optimality: the KF optimizes the mean squared error between the estimate and true system state given observations up to and included the recent moment *k*. The AF minimizes the mean squared error between the observation and the forecast of system output.

3. The parameters to be estimated in the KF are all components of the system state, which are time varying. The tuning parameters in the AF are the parameters in the filter gain, which are assumed to be time invariant. It is mentioned that the tuning parameters in the AF have no physical sense.

4. Nonlinear matrix equations are to be solved in the KF.

In this chapter, a new NNAF is proposed and tested for solving DA problems with moderate and high dimensional systems. In particular, the problem of estimating the ocean underwater current by the NNAF will be presented in detail due to its importance in producing energy from the ocean. It is mentioned that there are some basic ways to tap the ocean for its energy such as ocean wave, tidal range, tidal current, underwater current, and temperature differences in water. All these variables can be estimated by the NNAF. Ocean currents are suitable locations for deploying energy extraction devices such as turbines.

At the present, the use of ocean energy remains mainly in the prototype stage. Last few years, the prototypes are no longer confined only to laboratories, and some companies have set up commercial scale like RITE Project. In 2006, Verdant Power installed its first grid-connected Free Flow turbine (see Ref. [8]).

The modern DA techniques represent a computational challenge, even with the use of parallel computing with thousands of processors. Today, many operational weather prediction centers use high resolution models, with the number of observations (provided by a new satellite generation) growing exponentially. There is an increasing need to address new DA algorithms for HdSs and to improve their performance. This makes the problem of developing new DA algorithms like NNAF a great computational challenge.

In this chapter, we propose a new version of an artificial NN-based AF as a new DA technique, with the objective to construct a high performance forecasting system. Recently, the NN has become a popular and helpful model for classification, clustering, pattern recognition, and prediction in many disciplines. For a survey of NN applications in the real-world scenario, see Ref. [9]. In Ref. [10], the focus is on explaining the concept and evolution of machine learning, and some of the popular machine learning algorithms are described and compared. The work of Ref. [11] describes the important problems that can arise when applying NN models to meteorology and oceanography, especially in large-scale, low-frequency studies, and the attempts to resolve these problems. For data assimilation in chaotic dynamics using NNs, see Ref. [12].

In this chapter, the NNF based on AF developed in Ref. [13] will be constructed and implemented to produce the estimation results, which will serve as the reference for performance comparison with other NNFs. The need in developing the AF is to construct a filter, which is suitable in a new environment. The AF is a powerful device for signal processing, control applications, identification, inverse modeling, and prediction in an uncertain environment. The AF is formally defined as a *self-designing device with time-varying parameters that are adjusted recursively* in order to minimize an error function—a distance measurement between the reference or desired signal and the output of the adaptive filter. For a review of the AF, see Ref. [14].

The idea on the AF for data assimilation in HdS has been first presented in Ref. [13]. The originality of this AF approach lies in considering the AF as a filter, which minimizes the MPE for the system output, with tuning variables chosen as some pertinent parameters of the filter gain. This idea allows to optimize the filter performance iteratively, in real time, but not on the basis of a batch of observations as done in the 4D-Var approach. Moreover, the *dimensionless* tuning parameters are chosen in a way to ensure a stability of the filter. This makes the filter always stable during an optimization process and easily controls the values of tuning parameters. For a detail presentation of the main characteristics and advantages of the AF in the context of data assimilation for HdSs, see Section 2.2.

Assimilation experiments will be carried out with synthetic conventional data. The application of NNF produces a significant reduction in the computational effort. The goal of using the NN approach is to obtain a similar quality for analyses with better computational performance for prediction process.

The chapter is organized as follows. In the next section, a brief description of the NN and the AF for filtering problems is given. Section 3 presents the scheme of the NNF algorithm. The adaptation scheme for the NNFs is developed in Section 4. In

Section 5, the simulation experiments on filtering problems for moderate models are presented. Here first the one-dimensional system is presented to see clearly the idea and structure of the proposed NNAF. The assimilation experiment with the widely used Lorenz system, being of chaotic character, is also considered. The experiment with the ocean model MICOM (Miami Isopycnic Coordinate Ocean Model) for estimation of the oceanic current in different layers by the NNF and NNAF is described in Section 6, which demonstrates a possibility of the use of NNAF for state estimation in very high dimensional systems. It will be seen that the NNAF can improve considerably the performance of the NNF, which is beneficial for decision on future planning installation of stations exploiting renewable energy resources such as the tides and currents, as well as the waves and the thermal energy that resides in the seas. The conclusions are given in Section 7.

2. Artificial neural network filtering

2.1 Artificial neural network

Artificial neural networks (ANNs or NNs) are computing systems that are inspired by biological neural networks. Such systems are designed to "learn" by considering samples to perform the specified tasks. NNs are based on a collection of connected units or nodes called artificial neurons, which model the neurons in a biological brain. A standard simple NN scheme is shown in **Figure 1**. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it.

Neural networks have been extensively investigated in the context of adaptive control and system identification [15], but at the present, not so much in state estimation algorithms.

There are following principal types of NN: (i) feedforward neural network; (ii) radial basis function neural network; (iii) recurrent neural network (RNN); (iv) convolutional neural network (CNN); and (v) modular neural network. For more details, see Ref. [16]. For example, the RNN and CNN are designed to perform different tasks: the RNN is useful for *learning* the structure and syntax of the *language*, whereas the CNN is complex feedforward NN used for image classification and recognition because of its high accuracy, and it works by extracting features of image (identifying faces, individuals, street signs, and many other aspects of visual data).

NNs offer the useful properties such as linearity, nonlinearity, learning with teacher (modification of the NN weights), and adaptivity (capability to adapt their weights to changes in the surrounding environment). A neuron model scheme is given in **Figure 2**. One has

$$v_k = \sum_{i=1}^{m} W_{k,i} x_i + b_k, y_k = \varphi(v_k) \tag{1}$$

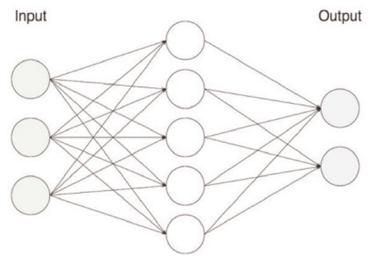

Figure 1.
Standard simple NN scheme: input, hidden, and output.

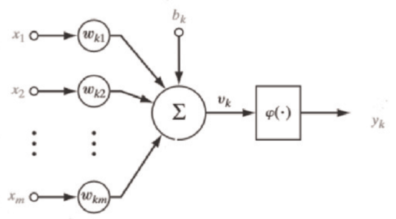

Figure 2.
Model of a neuron.

where $W_{k,i}$ is the synaptic weight between neuron i (at some l^{th} layer) and neuron k at the $(l+1)^{th}$, $x_1, \ldots, x_n$ are the inputs, $\varphi(.)$ is the activation function, b_k is the bias, and y_k is the output signal of the neuron k at the $(l+1)^{th}$ layer.

There are different activation functions (AcF) such as Step, Linear, and Sigmoid functions. In **Figure 3**, three AcFs, Step, Linear, and Sigmoid, are displayed.

One of the most important results concerning a capacity of NN as a powerful algorithm for function approximation is expressed in terms of the **universal approximation theorem,** which states that a feedforward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of R^n under mild assumptions on the AcF. The theorem thus states that simple neural networks can *represent* a wide variety of functions when given appropriate parameters. One of the first versions of the theorem was proved by Cybenko in 1989 for sigmoid activation functions [17].

In this chapter, the NN is designed in order to solve assimilation problems. The design procedure consists in the following steps:

Step 1. The choice of NN structure: What is the structure for the NN? This concerns the questions on (i) How many layers to be chosen? (ii) What are the connections between different nodes in different layers? (iii) What are the values to be initialized for the weights?

Step 2. The choice of variables for the input and output signals.

Step 3. Formulate an optimization problem for estimating the NN weights.

Step 4. Solve the formulated optimization problem: What is the optimization algorithm to be used?

All these questions will be addressed in details in Sections 3–6.

The objective of this chapter is to show that there exists a simple NNF structure, which can work efficiently and produce a good estimation results for the system states. As will be seen, all the NNFs employed in the experiments will have only one hidden layer. As to the AcFs, three types of AcF will be used: Linear, Sigmoid, and Step AcFs.

2.2 Adaptive filter

The AF for state estimation problem is defined as a filter, minimizing the MPE of the system output. One of the main features of the AF is that it is optimal among

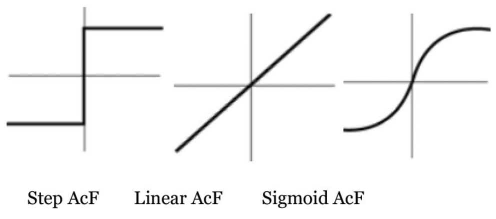

Figure 3.
Typical AcFs: step, linear, and sigmoid functions.

members of the set of parameterized state-space innovation representations, with the vector of pertinent parameters of the gain as a control vector. The assimilation problem considered in this chapter is formulated as a standard filtering problem

$$x(k+1) = \Phi x(k) + w(k) \tag{2}$$

$$z(k+1) = Hx(k+1) + v(k+1), k = 0, 1, \ldots \tag{3}$$

where $x(k)$ is the n-dimensional system state at the moment k, and $z(k)$ is the p-dimensional observation vector. Under standard conditions, related to the noise sequences $v(k), w(k)$, the optimal in mean squared estimate for $x(k)$ can be obtained from the system of recursive equations known as the KF.

$$\hat{x}(k+1) = \Phi\hat{x}(k) + K\varsigma(k+1), \varsigma(k+1) := z(k+1) - \Phi\hat{x}(k) \tag{4}$$

where $K = K(M)$ is the gain matrix, which is a function of $M := M(k+1)$—the error covariance matrix (ECM) for the state prediction error (PE).

$$e(k+1) = \hat{x}(k+1/k) - x(k+1), \hat{x}(k+1/k) = \Phi\hat{x}(k) \tag{5}$$

It is mentioned that in the KF, there exists a nonlinear matrix Riccati equation for computing $M(k+1)$. For a system with dimension of order10^7–10^8, under most favorable conditions (perfect knowledge of all system parameters and noise statistics), it is impossible to implement the KF (4) in the most powerful computers due to computational cost and insufficient memory.

In order to exploit the optimal structure of the filter (4), an AF is proposed in Ref. [13]. The filter is of the form (4) with the gain $K = K(\theta)$, which is a function of some vector θ of pertinent parameters. The choice of gain structure and its parameterization (i.e., θ) from the point of view of filter stability is studied in Ref. [18]. In Ref. [19], a simple and efficient algorithm is proposed for estimating the prediction error covariance matrix (ECM) based on the hypothesis on the separability of vertical and horizontal structure of the ECM and its parameterization. This allows to estimate optimally the unknown parameters of the structured parameterized ECM as well as to approach numerically the solution of the traditional Nearest Kronecker Problem in a simple and efficient way.

The optimality of the AF is understood in the Minimum Mean Squared (MMS) sense, which minimizes the objective function:

$$J(\theta) = E[\Psi(\varsigma)], \Psi(\varsigma) = \|\varsigma(k;\theta)\|^2$$
$$J = J(\theta) \to \min \arg\theta \tag{6}$$

with $E(.)$ is the mathematical expectation, and $\|.\|$ is the Euclidean norm. In Eq. (6), $\varsigma(k;\theta)$ is the PE for the system outputs, which depends on the vector of parameters θ. It is seen that if the KF minimizes the MMS *filtered* error for the system state (in state space), the AF minimizes the MMS *prediction* error for the system outputs in the observation space by tuning some pertinent parameters of the filter gain.

2.3 Simultaneous perturbation stochastic approximation (SPSA)

The choice of Eqs. (4) and (6) is important in many aspects in order to obtain a simple and efficient data assimilation algorithm. The idea lying behind Eqs. (4) and (6) is to select the vector $\theta \in \Theta$ consisting of some pertinent parameters as control variables for minimizing the cost function (6). Here Θ is the set of admissible values of θ, which ensure a filter stability.

The solution to the problem (6) can be found iteratively using a stochastic optimization (SA) algorithm.

$$\theta(k+1) = \theta(k) - \gamma(k+1)\nabla_\theta \Psi[\varsigma(k+1;\theta(k))] \tag{7}$$

where $\gamma(k)$ is a sequence of positive scalars satisfying some conditions to guarantee a convergence of the estimation procedure. The standard conditions are

$$\gamma(k)>0, \sum_{k=1}^{\infty}\gamma(k) = \infty, \sum_{k=1}^{\infty}\gamma^2(k) < \infty \tag{8}$$

The algorithm of Eqs. (4), (8), (11) is very simple to implement: at the k^{th} assimilation instant, we need to integrate the numerical model Φ by $\hat{x}(k)$ to produce the forecast $\hat{x}(k+1/k)$ and to update $\theta(k)$ according to Eq. (7). The algorithm of Eq. (7) is completely defined if the gradient of the sample cost function $\nabla_\theta \Psi[\varsigma(k+1;\theta(k))]$ is available. This gradient can be computed using the adjoint equation (AE) approach [4]. However, writing the AE code is very complicated and hard task for complex physical systems. Moreover, it requires a linearization of the system (2) and (4).

A much less computational burden can be achieved by measuring the sample objective function (but not based on a gradient formula): one approximates the gradient using the values of the cost function [on the basis of finite difference scheme (FDSA)]. This can be done by applying a so-called component-wise derivative approximation algorithm. However, for high-dimensional systems, this algorithm is inapplicable: for approximation of each partial derivative, we need to make two integrations of the model (2).

Let us look at the class of SPSA algorithms [20, 21]. The algorithm SPSA is of the same structure as that of FDSA, with the difference that it perturbs not each component of θ separably, but perturbs stochastically and simultaneously all its components. Concretely, let $\Delta_k \coloneqq (\Delta_{k,1}, \dots, \Delta_{k,n})^T$ be a random vector, $\Delta_{k,i}, i = 1, .., n$ are Bernoulli independent identically distributed (iid). The gradient of the sample objective function is estimated as

$$g = \nabla_\theta \Psi = (g_1, \dots, g_n)^T, \tag{9}$$

$$g_i = [\Psi(\theta(k) + c_k\Delta_k) - \Psi(\theta(k) - c_k\Delta_k)]/(2c_k\Delta_{k,i}), i = 1, \dots, n. \tag{10}$$

One sees from Eqs. (9) and (10) that all the directions in the parameter space are perturbed at the same time (the numerator is identical in all components). Thus, SPSA uses only two (or three) times the direct integration of the model, independently on the dimension of θ, which makes it possible to apply to high dimensional optimization problems. No development of the AE code is needed. Generally, SPSA converges in the same number of iterations as the FDSA, and it follows approximately the steepest descent direction, behaving like the gradient method [20, 21]. In fact, SPSA, with a random search direction, does not follow exactly the gradient path. In average, though, it tracks the gradient nearly because the gradient approximation is almost an unbiased estimator of the gradient, as shown in Refs. [20, 21]. For SPSA algorithm, the conditions for the sequence of positive c_k are

$$c_k>0, \sum_{k=1}^{\infty} c_k = \infty, \sum_{k=1}^{\infty} (\gamma(k)/c_k)^2 < \infty \tag{11}$$

where $\gamma(k)$ satisfies (8).

3. NN filtering (NNF)

Let us return to the input–output system (2)–(3). Instead of the AF, we will construct a NNF for estimating the system state $x(k)$ based on the set of observations from beginning up to the moment k. According to the NN scheme, we have to define the set of inputs and the set of output signals. These two sets are introduced as

$$S_I := \{\hat{x}_{nn}(k/k-1), z(k)\}, S_O := \{\hat{x}_{nn}(k)\} \tag{12}$$

where S_I is the set of inputs, and S_O is the set of output signals, and the estimate $\hat{x}_{nn}(k/k-1) = \Phi\hat{x}_{nn}(k-1)$ is the forecast obtained from the integration of $\hat{x}_{nn}(k-1)$ by the numerical model Φ. It is mentioned that $\hat{x}_{nn}(k-1)$ is the filtered estimate produced by the NNF, and symbolically, it is obtained by application of the NNF on the basis of the inputs S_I subject to the weights $W(k)$,

$$\hat{x}_{nn}(k) = f_{nn}[\hat{x}_{nn}(k/k-2), z(k); W(k)] \tag{13}$$

The difference between $\hat{x}_{nn}(k)$ in Eq. (13) and $\hat{x}(k)$ in Eq. (4) is that $\hat{x}_{nn}(k)$ is the estimate produced by the NNF, which is of the NN structure. Let us introduce a new model

$$x_m(k+1) = \Phi x_m(k) + w_m(k) \tag{14}$$

with $w_m(k)$ being a stochastic process, which may be different from $w(k)$. The reason is that in practice we are given not exactly the information on the process $w(k)$ (in the experiments for simplicity, we assume $w_m(k) = 0$). The vector $x_m(k)$ will serve as samples for learning process to estimate the NN parameters W. For illustration, one scheme of NNF structure for the system with two-dimensional state and two-dimensional observation vector is given in **Figure 4**.

In **Figure 4**, $\hat{W}_l(i,j)$ denotes the weight between the i node at the l^{th} layer and the j node at the next layer. The algorithm for updating the weights in the NNF looks as follows:

Algorithm 3.1 (Learning algorithm for estimating NNF weights).

Suppose the set of observations $z(k), k = 1, \ldots, N$ is available. At the moment $k = 1$, let the a priori values $\hat{W}_l(k-1;i,j)$, $\hat{x}_{nn}(k/k-1)$ be given. Suppose we are given the set of observations $z(k), k = 1, \ldots, N$.

Step 1. Pass the inputs $\hat{x}_{nn}(k/k-1), z(k)$ through the NNF. The NNF produces the output (estimate) $\hat{x}_{nn}(k)$.

Step 2. Compute the state error $e_{nn}(k) = \hat{x}_m(k) - \hat{x}_{nn}(k)$.

Step 3. Using the SPSA algorithm to update the NNF weights according to

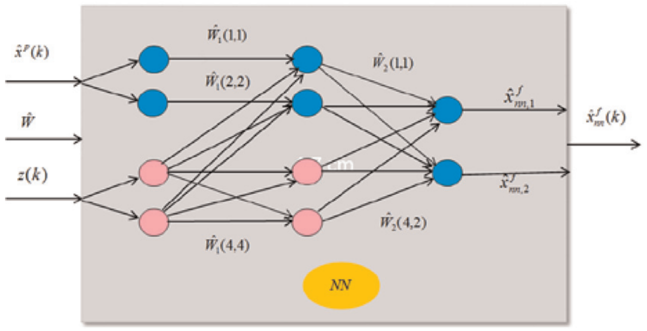

Figure 4.
Scheme of NNF structure.

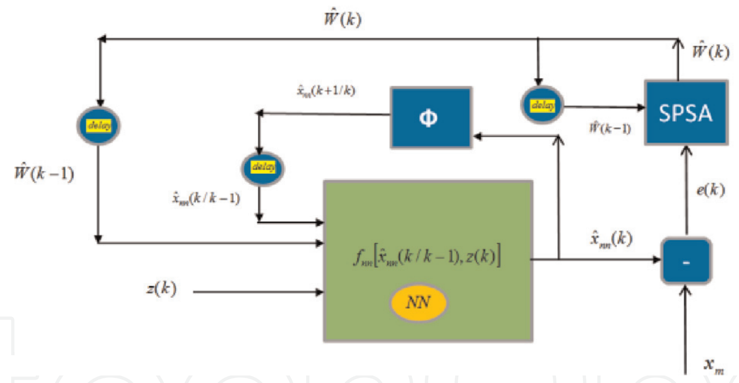

Figure 5.
Scheme for Algorithm 3.1.

$$\hat{W}(k) = \hat{W}(k-1) - \gamma(k)\nabla_W\Psi\left[\hat{W}(k-1)\right] \tag{15}$$

by minimizing the cost function

$$J_{nnf}(W) := E[\Psi(e_{nn}(W))] = E\left[\|e_{nn}(k; W)\|^2\right] \rightarrow \min\arg(W) \tag{16}$$

Step 4. If $k < N$, go to Step 1. Otherwise, go to Step 5.
Step 5. Stop.
The scheme for *Algorithm 3.1* is shown in **Figure 5**.

Comment 3.1. The sample gradient $\nabla_W\Psi\left[\hat{W}(k-1)\right]$ is evaluated as shown in Section 2.3 with $\gamma(k)>0$ (jointly with $c_k>0$) satisfying two conditions (8) and (11) ensuring a convergence of the algorithm (15) [20]. The sequence $\{c_k\}$ of positive scalar values participates in generating simultaneous perturbations to approximate $\nabla_W\Psi\left[\hat{W}(k-1)\right]$.

Comment 3.2. The estimates for the system state in (2) are $\hat{x}_{nn}(k)$, which are produced during running of *Algorithm 3.1*. To obtain the better estimates on the basis of nonadaptive NNF, the NNF should be rerun subject to the weights resulting from applying *Algorithm 3.1.*

Comment 3.3. If the number of observations is insufficient for well estimating the NNF coefficients, it is possible to simulate a longer sequence of pseudo-observations

$$z_m(k) = Hx_m(k) + v_m(k)$$

where $v_m(k)$ may be different from $v(k)$ (but chosen as closely as possible to $v(k)$ in the statistical sense). The similar experiment will be carried out in this framework for the ocean model MICOM in Section 6.

4. Adaptive NNF (NNAF)

To improve the performance of the NNF, the NNAF is developed in this section. The NNAF is of the same structure as that of the NNF, but it is designed to minimize the MPE of the system output. The NNAF is based on the second optimization procedure with the initial weights resulting from application of *Algorithm 3.1*.

For the design of the NNAF, the following objective function is introduced

$$J_{nnaf}(\theta) := E\left[\Psi(\varsigma_{nn}(W(\theta))\right] = E\left[\|\varsigma_{nn}(k; W(\theta))\|^2\right] \rightarrow \min\arg(\theta) \tag{17}$$

The difference between two objective functions (16) and (17) is that if (16) is expressed in terms of the (filtered) state estimation error, the error function (17) measures the difference between the observation and the prediction of the system output. As to the parameters to be adjusted, the following parameterizations are possible to be used:

$$W(\theta) = \hat{W}_o + \theta \tag{18}$$

$$W(\theta) = \theta\hat{W}_o \tag{19}$$

where $\hat{W}_o$ is the estimate obtained from the learning procedure (*Algorithm 3.1*). The initial value for θ is $\theta(0) = 0$ for the parameterization (18) and $\theta(0) = 1$ for (19). It is mentioned that when the structure of the NNAF is closed to the AF in Ref. [19], the parameterization (19) is more appropriate. We have now

$$\begin{aligned} \varsigma_{nn}(k; W(\theta)) &= z(k) - H\Phi\hat{x}_{nn}(k-1) \\ \hat{x}_{nn}(k-1) &= f_{nn}[\hat{x}_{nn}(k-1/k-2), z(k-1); W(\theta)] \end{aligned} \tag{20}$$

More concretely, the NNAF is a result of the following.

Algorithm 4.1 (Adaptive procedure for improving NNF performance).

At the initial moment $k = 1$ suppose we are given $\hat{x}_{nn}(k/k-1)$ and $\hat{W}_l(i,j)$ ($\hat{W}_l(i,j)$ are obtained at the end of applying *Algorithm 3.1*). Suppose the set of observations $z(k), k = 1, \dots, N$ is available.

Step 1. Pass the inputs $\hat{x}_{nn}(k/k-1), z(k)$ through the NNF initialized by the weights $\hat{W}_l(i,j)$. The NNF produces in the output the estimate $\hat{x}_{nn}(k)$.

Step 2. Compute the error $\varsigma(k) = z(k) - H\hat{x}_{nn}(k/k-1)$ where.

$$\hat{x}_{nn}(k/k-1) = \Phi\hat{x}_{nn}(k-1), \hat{x}_{nn}(k-1) = f_{nn}\left[\hat{x}_{nn}(k-1/k-2), z(k-1); \hat{W}_l(i,j)\right].$$

Step 3. Using the SPSA algorithm to update the weights

$$\theta(k) = \theta(k-1) - \gamma(k)\nabla_{\theta(k-1)}\Psi[\theta(k-1)] \tag{21}$$

for seeking the optimal parameters θ to minimize the cost function

$$J_{nnf}(\theta) := E\left[\Psi\left(\varsigma_{nn}(k; \hat{W}(\theta))\right)\right] = E\left[\left\|\varsigma_{nn}\left(k; \hat{W}(\theta)\right)\right\|^2\right] \to \min \arg\theta \tag{22}$$

Step 4. If $k < N$, go to Step 1. Otherwise, go to Step 5.

Step 5. Stop.

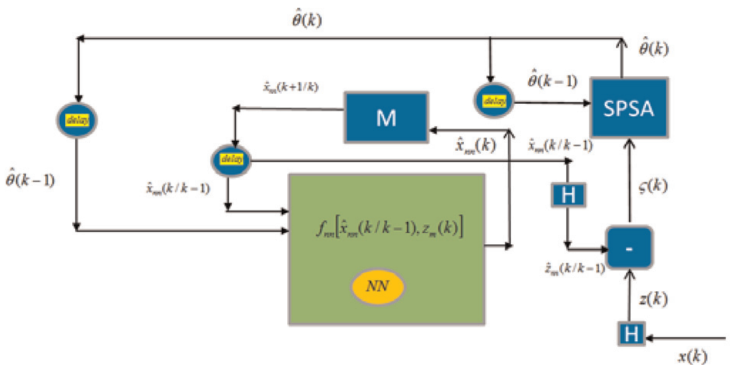

Figure 6.
Scheme for Algorithm 4.1.

Figure 6 displays the scheme of *Algorithm 4.1*. Compared to **Figure 3**, here the weights are assumed given (as resulting from application of *Algorithm 3.1*). States of the numerical model are no longer participating in the SPSA algorithm; instead, the innovation vector is involved in the optimization process.

5. Numerical experiments for systems of moderate dimensions

5.1 Experiment 1: One-dimensional system

We present in this section a very simple numerical experiment with an one-dimensional system to illustrate clearly the idea on the NNAF.

Let us consider the system of Eqs. (2)–(3) subject to $\Phi = 0.99, H = 1, Q = 0.09, R = 0.01$. The experiment is carried out on the basis of a NNF, and the performance of the NNF is compared with that of the KF and NNAF. The structure of the NNF is shown in **Figure 7**.

Thus, the NNF has one hidden layer structure. The Linear and Sigmoid AcFs are used in this NNF. The weights $W_l(i,j)$ are estimated by applying *Algorithm 3.1*. Here the error $e(k) = x(k) - x_m(k)$ is the difference the "wrong" model state governed by $x_m(k+1) = \Phi x_m(k), k = 0, 1, \ldots, N$ and the true system state governed by $x(k+1) = \Phi x(k) + w(k), k = 0, 1, \ldots, N$. The performance of the NNF is expressed in the terms of root mean square (RMS) of the state prediction error.

Figure 8 shows the performances of three filters—NNF, NNF-FIX, and NNAF. The red curve "*NNF*" is the RMS of state prediction error produced by the NNF during the optimization process (*Algorithm 3.1*). The green curve "*NFF-FIX*" is produced by the NNF with the fix weights obtained at the end of *Algorithm 3.1*. As to the blue curve "*NNAF,*" it displays the performance of the NNAF by applying *Algorithm 4.1*. The reason for which the NNAF has produced the best performance, compared to the NNF and NNF-FIX, is that the NNAF minimizes the MPE for the system outputs. It means that learning process does not extract all useful information on the system state contained in the observations.

Time evolution of the weights, associated with connecting links between the nodes in the hidden layer and the node of the output layer, is displayed in **Figure 9**.

Figure 10 shows the performances of four filters implemented for estimating the system state of the 1d model: NAF (nonadaptive filter), AF developed in Ref. [18], NNF (*Algorithm 3.1*), and NNAF (*Algorithm 4.1*). The NAF is the filter with the constant gain computed on the basis of the stationary solution of the Riccati equation associated with the KF. One sees that the NNF behaves better than the NAF at the beginning of the assimilation process. Application of adaptation allows to

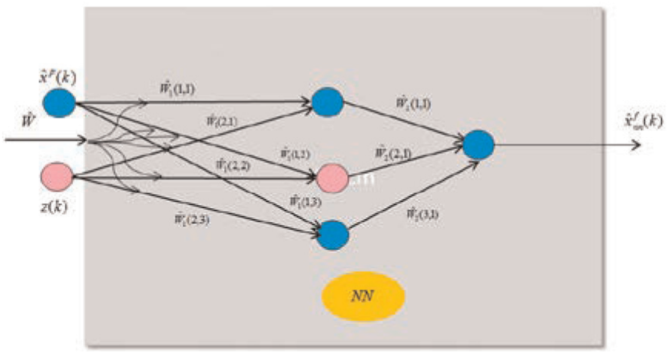

Figure 7.
Structure of the NNF for the one-dimensional system.

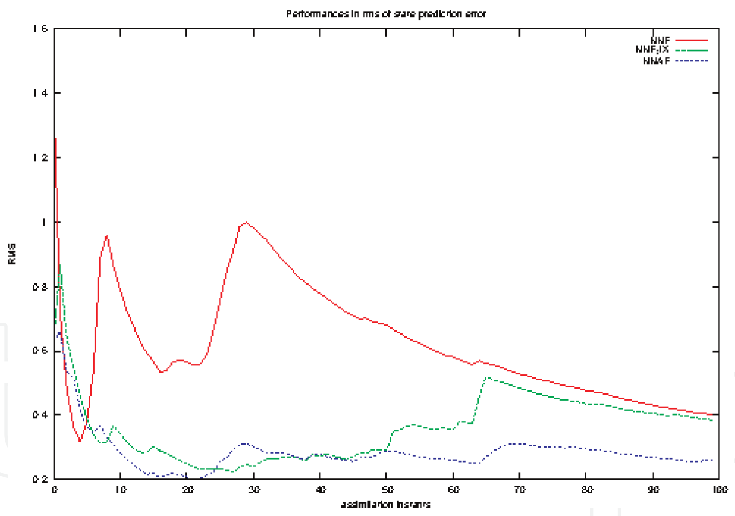

Figure 8.
*RMSs of state prediction error resulting from three filters: NNF (*Algorithm 3.1*, red line), NNF-FIX (green line), and NNAF (*Algorithm 4.1*, blue line). The error produced by the NNF during the optimization process is too high compared to that of the NNF-FIX. It (NNF) reaches the same level (of NNF-FIX) at the end of the assimilation period. As to the NNAF, the error remains low during all assimilation periods (after about the first 10 iterations).*

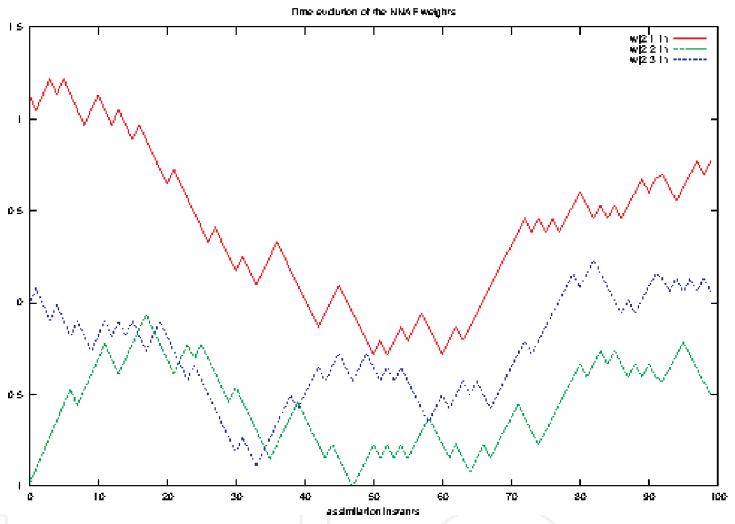

Figure 9.
Time evolution of several weights (between the nodes in the second (hidden) layer and the nodes in the output layer) in the NNAF during the adaptation process: the weight of the connection link between the first node (hidden layer) and the output node (red curve); the weight of the connection link between the second node (hidden layer) and the output node (green curve); and the weight of the connection link between the third node (hidden layer) and the output node (red curve).

improve considerably the performances of the two nonadaptive NAF and NNF. In general, two adaptive filters AF and NNAF are of the same performance.

5.2 Experiment 2: Chaotic Lorenz system

5.2.1 Lorenz equations

The Lorenz attractor is a chaotic map, noted for its butterfly shape. The map shows how the state of a dynamical system evolves over time in a complex, non-repeating pattern.

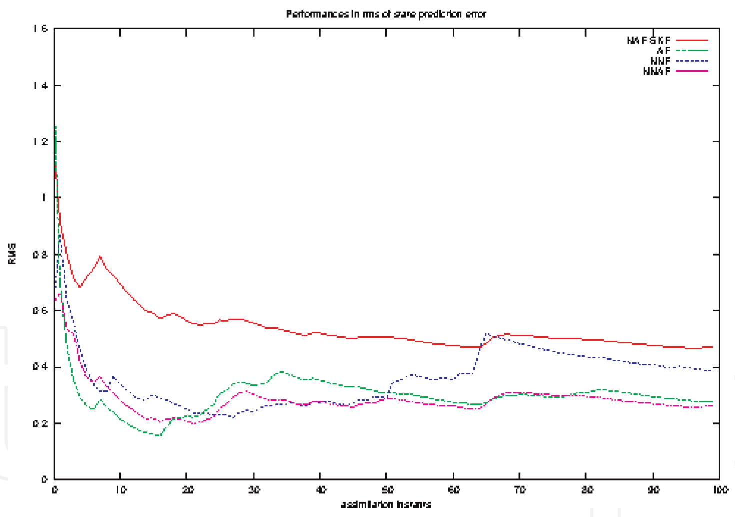

Figure 10.
*RMSs of state prediction error resulting from four filters: NAF (nonadaptive filter, red line), AF (green line), NNF (blue line), and NNAF (pink line). The NNF, thanks to the learning process, behaves better than NAF. The AF developed in Ref. [18] and NNAF (*Algorithm 4.1*) are of the same performance.*

The attractor itself and the equations from which it is derived were introduced by Lorenz [22], who derived it from the simplified equations of convection rolls arising in the equations of the atmosphere.

The equations that govern the Lorenz attractor are:

$$dy_1/dt = -\sigma(y_1 - y_2),$$
$$dy_2/dt = \rho y_1 - y_2 - y_1 y_3, dy_3/dt = y_1 y_2 - \beta y_3 \tag{23}$$

where σ is called the Prandtl number, and ρ is called the Rayleigh number. All $\sigma, \beta, \rho > 0$, and usually $\sigma = 10, \beta = 8/3, \rho$ are varied. The system exhibits chaotic behavior for $\rho = 28$ but displays knotted periodic orbits for other values of ρ.

5.2.2 Numerical model

In the experiments, the parameters σ, β, ρ are chosen to have the values 10, 28, and 8/3, respectively, for which the "butterfly" attractor exists.

The numerical model (NM) is obtained by applying the Euler method (first-order accurate method) to approximate Eq. (23). Symbolically, we have the NM

$$y(t_{k+1}) = F[y(t_k)], y(t_{k+1}) := [y_1(t_{k+1}), y_2(t_{k+1}), y_3(t_{k+1})]^T \tag{24}$$

where $\delta t := t_{k+1} - t_k$ is the *model* time step. The observations arrive at the assimilation instants T_k and $\Delta T_k := T_{k+1} - T_k$.

5.2.3 Numerical model and observation

The numerical model is derived subject to $\delta t := 0.005$. The true states $x(k)$, produced by the NM corrupted by additive noise at the moment T_k subject to $\Delta T_k = 25\delta t$, are taken as observation $z(k), k = 1, \ldots, N_o$ contaminated with the noise v_k having zero mean and variance σ_k^2, with $\sigma_k = 0.1$. The true dynamical system, corresponding to the transition of the states between two assimilation instants T_k and T_{k+1}, is expressed as

$$x(k+1) = F[x(k)] + w(k) \tag{25}$$

In Eq. (25), $w(k)$ simulates the model error. The sequence $\{w(k)\}$ is assumed to be a white noise having diagonal covariance matrix with variances 2, 12.13, and 12.13, respectively. The observation system is given by

$$z(k+1) = Hx(k+1) + v(k) \tag{26}$$

where the observation operator $H = \left[h_1^T, h_2^T\right]^T, h_1 = (1, 0, 0), h_2 = (0, 0, 1)$, that is, the first and third components $x_1(k), x_3(k)$ are observed at each time instant k. The noise sequence $\{v_k\}$ is white with zero mean and variance $R = 2I_2$, where I_n is the unit matrix of dimension n. The initial estimate in all the filters is given by $\hat{x}(0) = (5, 10, 27)^T$.

The true model (TM) state $x^*(0)$ is modeled as the solution of Eq. (25) subject to the initial state $x^*(0) = (1.509, 1.531, 25.461)^\mathrm{T}$.

The problem in this experiment is to apply the NNF and its adaptive version to estimate the true system state using the observations $z(k)$, $k = 1, \dots, N_o$, and to compare their performances with those produced by the AF.

5.2.4 NNF structure and numerical results

The NNF used in this section has the following structure.

The NNF has only one hidden layer, and its AcFs are chosen as Linear and Step AcFs (**Figure 3**). The structure in **Figure 11** is chosen to be as simple as possible. It is interesting to ask whether there are more complicated NNFs, which can yield a better performance? However, in this chapter, we restrict our self to considering only the NNF with the structure in **Figure 11**.

The weights in the NNF are first estimated (learning process) using the sequence of the erroneous model (EM) states [which is a noise-free version of the TM (true model) (25)]. The initial state in the EM is the same as that in the TM. Typical evolution of weights during the learning process is shown in **Figure 12**. It is seen that the weights' estimates are relatively stable, though we cannot say they are already stabilized due to an insufficient number of iterations. All the weights are initialized with the values 0.5. We have also tested the NNF for the situation with

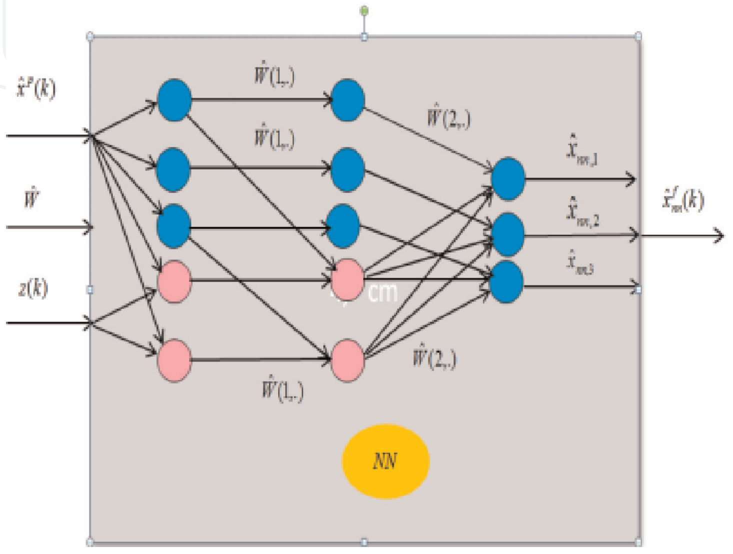

Figure 11.
Structure of the NNF for the Lorenz data assimilation system.

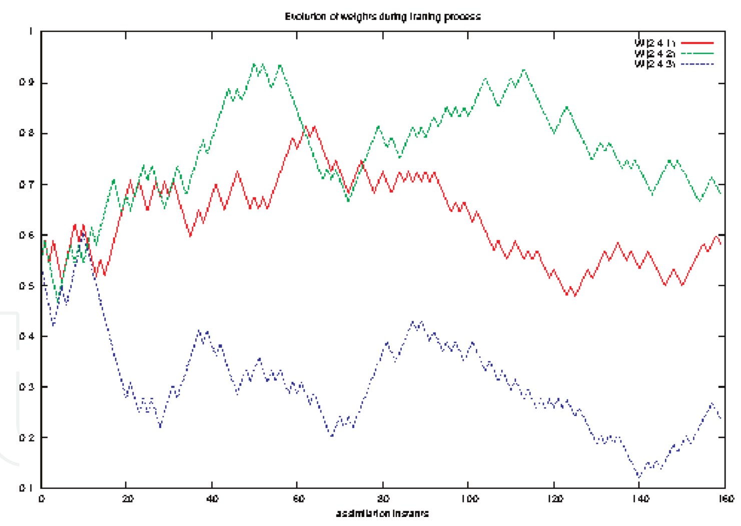

Figure 12.
Typical time evolution of NN weights (between the nodes in the second (hidden) layer and the node in the output layer) in the NNAF during the learning process: the weight of the connection link between the fourth node (hidden layer) and the first node of the output layer (red curve); the weight of the connection link between the fourth node (hidden layer) and the second node of the output layer (green curve); and the weight of the connection link between the fourth node (hidden layer) and the third node of the output layer (red curve). One notes that the weights do not yet attend a stationary regime.

the initial weight values equal to 1. The results show that there is no significant difference in filter performance between these two situations.

It is mentioned that the Lorenz system is very sensitive to the initial state condition. Different initial states will lead to a variety of solution behavior: some solutions are steady; others oscillate between two or more states; and still others vary in an irregular manner.

We generate observations at 160 assimilation instants (4000 model time steps). The different filters will be implemented to assimilate the observations.

Figure 13 shows the performance of three filters—NFF1, NNF2, and NNF3, which are initialized with the same initial condition $\hat{x}(0) = (5, 10, 27)^T$. It is seen

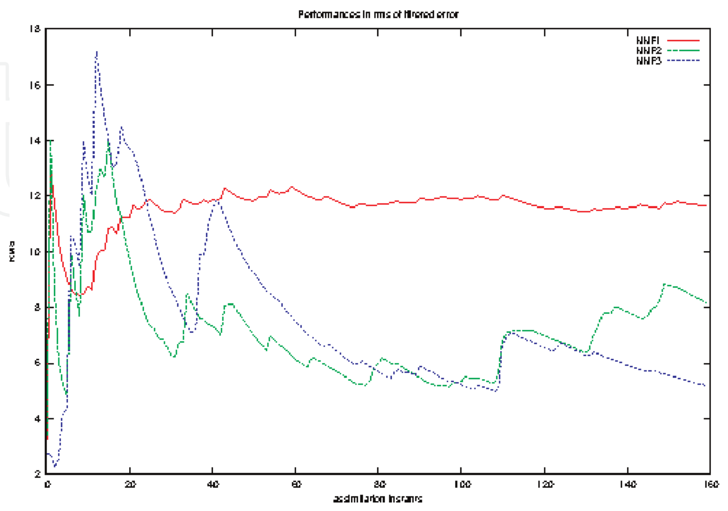

Figure 13.
RMSs of state filtered error resulting from three filters: NNF1 (red line)—the Lorenz model alone (without data assimilation), which is running subject to the another initial state (5,10,27); NNF2 (green line)—the NNF running with fix initial (connection link) weights (i.e., without learning process); and NNF3 (blue line)—the NNF (applying Algorithm 3.1*) with the initial weights in NNF2. One sees that the estimation error is too high if no observations are assimilated into the model. If the observations are assimilated, with the learning process, the error is decreasing as the learning process progresses and becomes lower than that in NNF2 at the end of the learning process.*

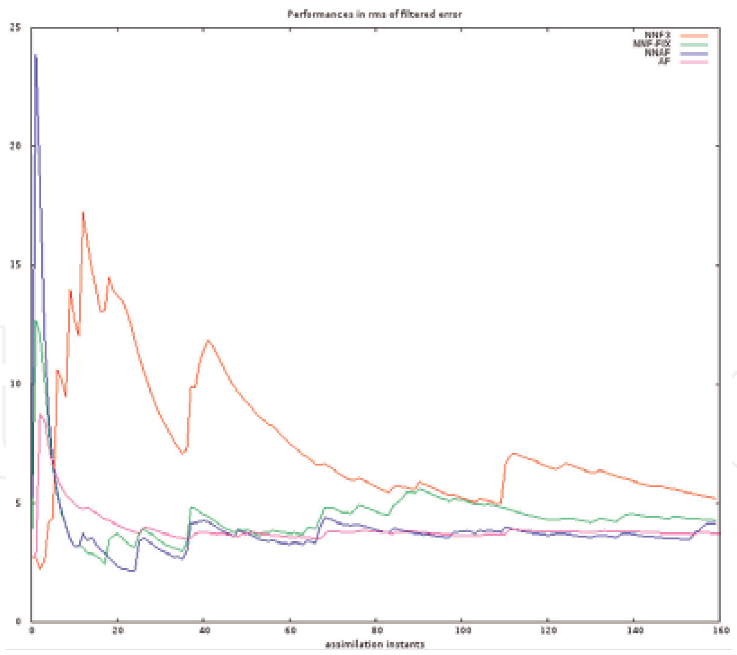

Figure 14.
RMSs of state filtered error resulting from three filters: NNF3 (red line)—the same NNF3 in ***Figure 13*** *(*Algorithm 3.1*); NNF-FIX (green line)—the NNF subject to the fix coefficients obtained at the end of the optimization process in NNF3; and NNAF (blue line)—the NNAF, which is obtained by tuning the weights in NNF (initialized by the values obtained at the end of NNF3) to minimize the prediction error for the system outputs (*Algorithm 4.1*) AF (pink line)—the AF (4) and (6) developed in Ref. [19].*

that without data assimilation, the erroneous initial condition leads to big errors between the true states and the states of the erroneous model (red curve, NNF1). By simply initializing the weights in the NNF, it is possible to reduce an important portion of the errors (green curve, NNF2). At the beginning of the learning process (NNF3), the error is relatively high (even higher than that in NNF2), but it quickly decreases after about 20 assimilation instants.

The SPSA algorithm is really capable of well searching optimal weights as seen from comparison of performances of the two filters NNF3 and NNF4 (**Figure 14**). Finally, tuning the weights in the NNF to minimize the prediction error (NNAF) of the system output yields the best results (among all the filters of the neural network structure) as seen from the curve "*NNAF*" (*Algorithm 4.1*). To verify the optimality of the NNAF, we have also implemented the AF (4) and (5) based on tuning the parameters of the filter gain [19]. The performance of the AF is shown by the pink curve "*AF*" in **Figure 14**, which is almost the same as that of the NNAF. As before, all the filters are starting from the same initial condition as used in the NNF1. It is interesting to note that the two adaptive filters NNAF and AF are of different structures, but they can yield the same performance. If for the NNAF, the choice of the NN structure (sets of inputs, outputs; number of layers, initial values of the weights to be selected, and learning process) is most important, in the AF the question about the choice of stabilizing gain with appropriate parameterization is primordial for the success of the AF.

6. Numerical experiment for high dimensional ocean model MICOM

The oceans represent a vast source of renewable energy. In general, ocean energy can be divided into six types of different origin and characteristics: ocean wave, tidal range, tidal current, ocean current, thermal energy, and salinity gradient. Knowledge of these quantities is important for decision making on sizing, placement, and optimization of installation of system energy.

These oceanic quantities are now can be well modeled through an ocean numerical model, which is constructed according to the ocean configuration of interest. The difficulties arising here are that the model solution gives us only the relative values of quantities of interest because the model is always far from the reality. Therefore, in order to more accurately assess the true values of oceanic quantities, the measurement data are required to be inserted into the model to correct the model solution. The role of DA algorithms is just to properly (in some sense, in the best way) insert data into a numerical model so that to ensure a stability of the algorithm and to minimize the estimation error.

In this section, the experiment with the ocean model MICOM is carried out, which is aimed at estimating the ocean current in different depths (vertical layers).

6.1 Numerical model

The ocean model used for the DA experiment with the NNF and NNAF is the MICOM (Miami Isopycnic Coordinate Ocean Model), which is identical to that described in Ref. [18]. The model configuration is a domain situated in the North Atlantic from 30°N to 60°N and 80°W to 44°; for the exact model domain and some main features of the ocean current produced by the model, see Ref. [19].

The MICOM used here has the state $x = (h, u, v)$, where $h = h(i, j, k)$ is a layer thickness, and $u = u(i, j, k)$ and $v = v(i, j, k)$ are two velocity components. In the model, $i = 1, \ldots, 140$; $j = 1, \ldots, 180$; and $N_z = 4$ vertical layers, resulting in the state vector of dimension 302,400. The model is integrated from the state of rest during 20 years (ys). During the period of 2 years 19 and 20, every 10 days, the sea surface height (SSH) is computed as a linear function of the layer thickness h, which is considered as observation in assimilation experiments (totally 72 observations are available). To be closer to realistic situations with satellite observations, available only at along-track grid points, we assume that the observations are noise-free sea surface height (SSH), available only at the set $S_o$of horizontal grid points $S_o = \{(i, j), i = 1, 11, \ldots, 131, j = 1, 11, \ldots, 171\}$(but not at all horizontal grid points).

6.2 Structure of NNF

We first construct a NNF block for estimating the layer thickness variable h using the SSH observations. The Linear and Step AcFs are used in the NNF. The two velocity components $(u, v)$are computed using the geostrophy hypothesis. The scheme of the NNF is shown in **Figure 15**.

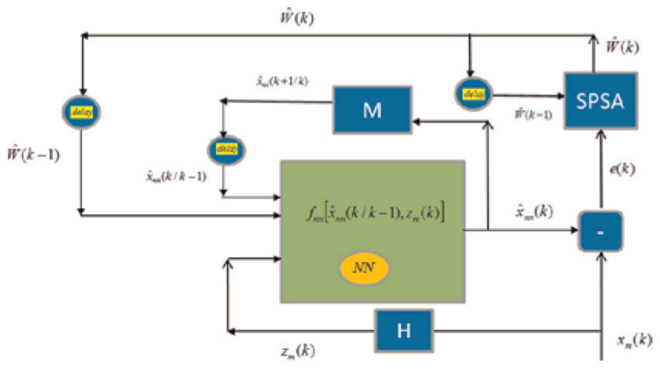

Figure 15.
Scheme for NNF for data assimilation in MICOM.

It is mentioned that as the observations are available only at the set S_o, one sub-NN block (with sigmoid AcF), named "optimal interpolation," is designed to interpolate the observations (i.e., to estimate the values of observations), available at S_o, on all the horizontal grid points of the surface. This operation is very important to ensure smooth corrections to all layer thicknesses and to estimate all layer velocities.

The NNF has one hidden layer as that used in the Lorenz experiment. One note from **Figure 15** that in the MICOM experiment, the *Algorithm 3.1* has been implemented not on the basis of the true observations $z(k)$ in Eq. (3) but using the other set of "observations" (calculated from $x_m(k)$, see **Figure 15**) for estimating the weights in the NNF (see *Comment 3.3*). To test the influence of "true samples" and "erroneous samples" on the NNF performance, we have implemented two learning procedures, one is based on the set of erroneous samples $x_m \propto x_{err}$, and another on the basis of true samples $x_m \propto x^*$.

6.3 Numerical results

In **Figure 16**, we show the evolution of some NNF weights during the learning process on the basis of the set of erroneous samples.

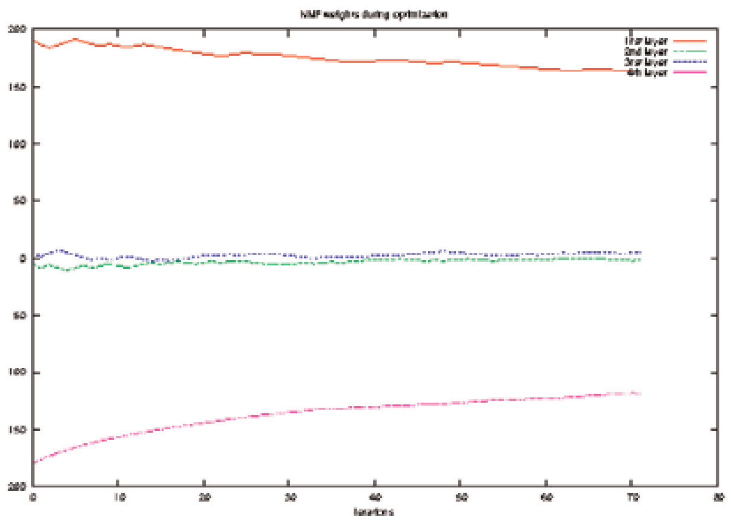

Figure 16.
Typical evolution of estimated weights in different layers during learning process. Initial weights are taken from application of the Cooper and Haines's water column lifting–lowering method [19].

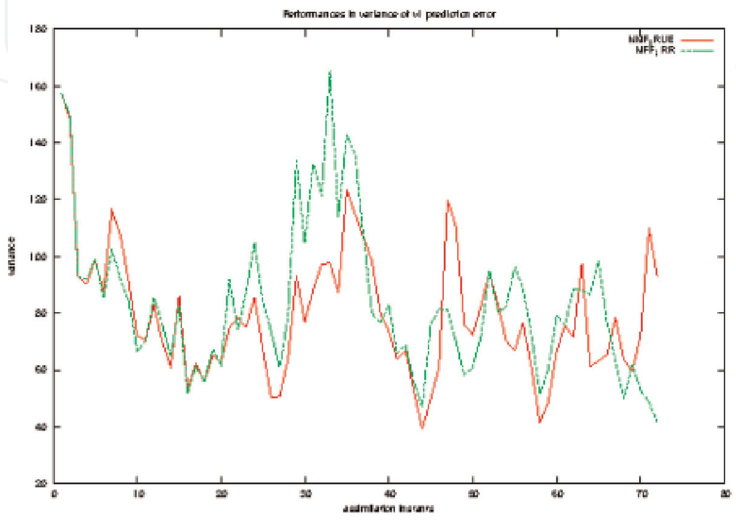

Figure 17.
Variances of prediction error for the velocity (v component) at the first layer by NNFT (red curve) and NNFE (green curve).

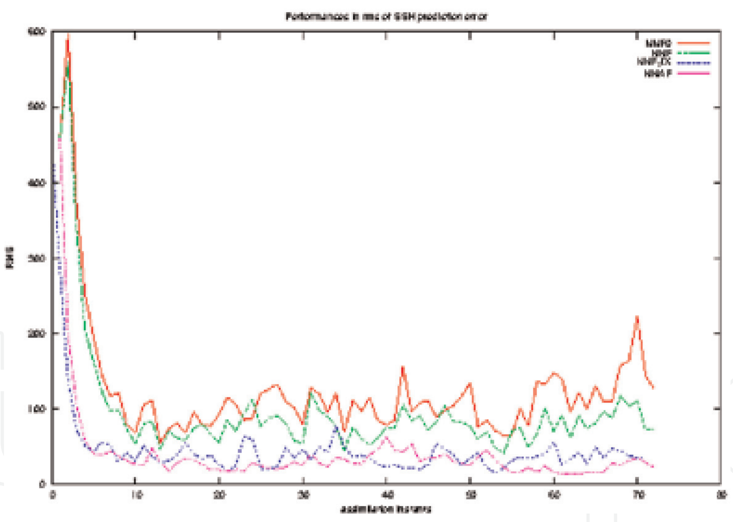

Figure 18.
Performance of different filters (in variance) of SSH prediction error (objective function).

The weights, obtained at the end of learning process, are inserted into two NNFs—NNFT and NNFE, where the weights in the NNFT are obtained by minimizing the difference between the NNF state and the true system state, and NNFE—with the weights resulting from minimizing the difference between the NNF state and the state of the erroneous model. **Figure 17** displays the variances of the prediction error for velocity (v component) at the first layer produced by NNFT (red curve) and NNFE (green curve). One sees that learning on the basis of samples of the true state yields more optimal weights, especially in avoiding the error pick, but generally speaking, only a slight improvement is observed.

We have tested different filters to see the effect of optimization in finding the weights in the NNF and those in the NNAF. The performances of these filters are shown in **Figure 18**. From **Figure 18**, the error in the $NNF0$ on the basis of the initialized weights (red curve) is too high compared to those of the other filters. The improvement is observed for the NNF (*Algorithm 3.1*) during the optimization process (green curve). If we run the $NNF - FIX$ with the fix weights, resulting at the end of the optimization process in theNNF, a much better performance has been produced (blue curve). Finally, the best results are obtained by running the NNAF (*Algorithm 4.1*) (pink curve).

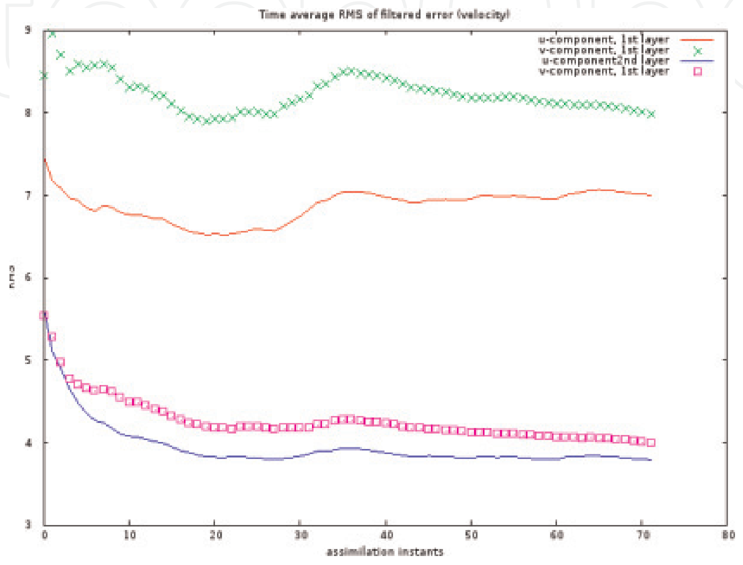

Figure 19.
Spatial temporal average RMS of velocity at first two layers for u and v component (by NNF).

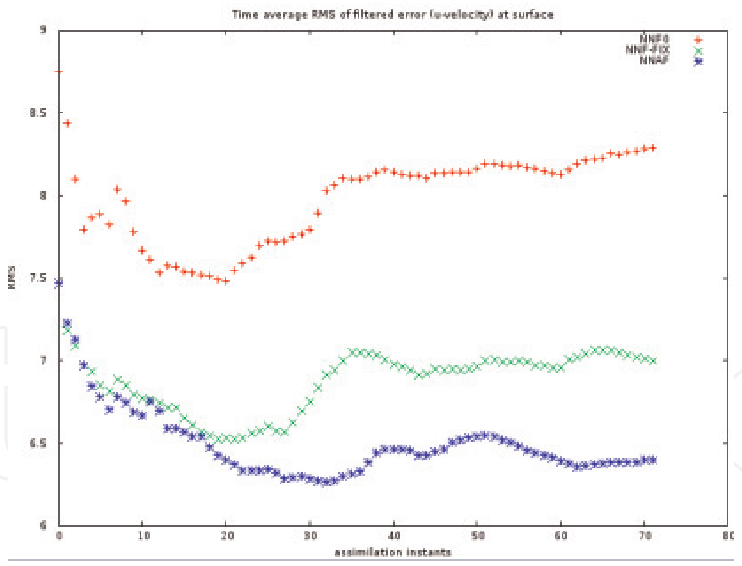

Figure 20.
Spatial temporal average RMS of u *velocity at surface produced by NNF, NNF-FIX, and NNAF.*

To have a more complete idea on how work the *NNF* and *NNAF* in high dimensional setting, in **Figure 19**, we show the spatial temporal average RMS of velocity *(u,v)* at the first two layers. As expected, estimating the velocity at the surface is more difficult than in deeper layers. One remark is that the u-velocity component is easier to estimate (i.e. with less error) than the v component. **Figure 20** gives a picture on performance of different filters. It displays the error (for u velocity at the surface) produced by three filters: $NNF0$, $NNF - FIX$, and $NNAF$. If in **Figure 18**, the difference (in *SSH* forecast error) between $NNF - FIX$ and *NNAF* is less visible for the velocity variable, and the superiority of the*NNAF* over the $NNF - FIX$ is remarkable. This tendency is observed in the comparison of velocity estimation errors produced by the three mentioned filters, at different layers as well as for the v-velocity component.

7. Conclusions

In this chapter, a constructive algorithm offering an attractive approach for the design of a NNAF is proposed. In particular, a simple NN structure with one hidden layer has proved to be efficient: it can yield a high performance of all the NNAFs implemented in the experiments—either with moderate or with high dimensional systems. The superiority (in performance) of the NNAF is confirmed even for the Lorenz system with chaotic character, which is very popular due to its high sensitivity to the initial condition. It would emphasize here that in the NNF and NNAF, no initial condition is used as a control variable to formulate the optimization problems. The experiment with the HdS ocean model MICOM demonstrates a possibility to exploit the NNF for state estimation problems for realistic HdSs.

It is found from the experiments that an appropriate choice of initial values for the weights is important for yielding a high performance of the NNF, with a moderate number of hidden layers and iteration steps. In the MICOM experiment, we have initialized the weights with the values corresponding to the lifting-lowering water column method. A more efficient way to initialize the weights (to achieve a better performance) is to follow the method described in Ref. [23] based on computing the dominant Schur vectors of the system dynamics. It is highly probable that the NNF with a larger number of hidden layers and nodes could produce a better performance. The question on how many numbers of layers and nodes to be chosen for the NNAF is open and is left for the future study.

The experimental results confirm that including adaptation as a mechanism for minimizing the MPE of the system outputs is an efficient way to improve the NNF performance.

We want to stress that the success of the NNAF is possible partly due to application of SPSA optimization algorithms. These algorithms are relatively ease of use for solving difficult multivariate optimization problems and are simple to implement and possess a good convergence rate. They require only two objective function measurements per iteration regardless of the dimension of the optimization problem. These features make SPSA as a powerful tool for solving optimization problems with nonlinearities and high dimensionality of complex dynamical systems.

Finally, as the developed NNAF in this chapter is very effective and easy to implement, which makes it quite applicable for solving different real-world tasks, in particular for solving DA problems in more sophisticated ocean models like the HYCOM (HYdrodynamic Coordinates Ocean Modeling). With the HYCOM model, it is possible to estimate, for example, thermal energy and salinity gradients—the quantities, serving important sources of renewable energy to generate electricity for global electrical demand in future. It is mentioned that the model HYCOM is used by the French Navy Hydrographic and Oceanographic Service (SHOM) for the prediction of ocean currents, to predict variations on the climatic scale, with a model of waves and a metric resolution, with very short-lived physics at the littoral scale. Implementation of the NNAF represents real challenges for the future operational forecasting systems [24].

Author details

Hong Son Hoang* and Remy Baraille
REC/HOM/SHOM, Toulouse, France

*Address all correspondence to: hhoang@shom.fr

References

[1] Kolmogorov AN. Stationary sequences in Hilbert space. Bulletin of Moscow University. 1941;**2**(6):1-40. In Russian

[2] Wiener N. Extrapolation, Interpolation, and Smoothing of Stationary Time Series With Engineering Applications. Cambridge: MIT Press; 1949

[3] Kalman RE. A new approach to linear filtering and prediction problems. Transaction of the ASME—Journal of Basic Engineering. 1960;**82**(Series D): 35-45

[4] Talagrand O, Courtier P. Variational assimilation of meteorological observations with the adjoint vorticity equation. I: Theory. Quarterly Journal of the Royal Meteorological Society. 1987; **113**:1311-1328

[5] Evensen G. Data Assimilation: The Ensemble Kalman Filter. Berlin: Springer; 2007

[6] Gordon NJ, Salmond DJ, Smith AFM. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. IEE Proceedings of Radar and Signal Processing. 1993;**140**(2):107-113. ISSN 0956-375X

[7] Carrassi A, Bocquet M, Bertino L, Evensen G. Data assimilation in the geosciences: An overview on methods, issues and perspectives. Wiley Interdisciplinary Reviews: Climate Change. 2017. DOI: 10.1002/wcc.535

[8] Roosevelt Island Tidal Energy (RITE) Project Demonstration. Available from: https://tethys.pnnl.gov/annex-iv-sites/roosevelt-island-tidal-energy-rite-project-demonstration

[9] Uihlein A, Magagna D. Wave and tidal current energy–A review of the current state of research beyond technology. Renewable and Sustainable Energy Reviews. 2016;**58**:1070-1081

[10] Das K, Behera RN. A survey on machine learning: Concept, algorithms and applications. International Journal of Innovative Research in Computer and Communication Engineering. 2017;**5**(2): 1301-1309

[11] Hsieh W, Tang B. Applying neural network models top prediction and data analysis in meteorology and oceanography. Bulletin of the American Meteorological Society. 1998;**79**(9): 1855-1870

[12] Nowosad A, Neto A.R. and Campos Velho H. Data assimilation in chaotic dynamics using neural networks. In: International Conference on Nonlinear Dynamics, Chaos, Control and Their Applications in Engineering. 2000; 212–221

[13] Hoang HS, De Mey P, Talagrand O, Baraille R. A new reduced-order adaptive filter for state estimation in high dimensional systems. Automatica. 1997;**33**:1475-1498

[14] Haykin S. Adaptive Filter Theory. Upper Saddle River, NJ: Prentice Hall. ISBN 978-0-13-048434-5; 2002

[15] Patrikar A, Provence J. Nonlinear system identification and adaptive control using polynomial networks. Mathematical and Computer Modelling. 1996;**23**(112):159-173

[16] Haykin S. Neural Networks and Learning Machines. 3rd ed. Upper Saddle River, NJ: Pearson Education; 2009

[17] Cybenko G. Approximations by superpositions of sigmoidal functions. Mathematics of Control, Signals, and Systems. 1989;**2**(4):303-314. DOI: 10.1007/BF02551274

[18] Hoang HS, Talagrandand O, Baraille R. On the design of a stable filter for state estimation in high dimensional systems. Automatica. 2001;**37**:341-359

[19] Hoang HS, Baraille R. On the efficient low cost procedure for estimation of high-dimensional prediction error covariance matrices. Automatica. 2017;**83**:317-330

[20] Spall JC. Introduction to Stochastic Search and Optimization. New York: Wiley; 2003

[21] Hoang HS, Baraille R. Stochastic simultaneous perturbation as powerful method for state and parameter estimation in high dimensional systems. In: Baswell AR, editor. Advances in Mathematics Research. Vol. 20. NY: Nova Science Publishers; 2015. pp. 117-148

[22] Lorenz EN. Deterministic non-periodic flow. Journal of the Atmospheric Sciences. 1963;**20**:130-141

[23] Hoang HS, Baraille R. Prediction error sampling procedure based on dominant Schur decomposition. Application to state estimation in high dimensional oceanic model. Journal of Applied Mathematics and Computing. 2011;**12**:3689-3709

[24] Data.shom.fr. Oceanographic Costal Forecasts. Available from: http://www.shom.fr/en/activities/projects/oceanographic-forecasts/datashom-coastal-oceanographic-forecasts-ibi/