ELSEVIER

Ocean Modelling 4 (2002) 363–383

Ocean Modelling

www.elsevier.com/locate/omodol

A neural network technique to improve computational efficiency of numerical oceanic models ☆

Vladimir M. Krasnopolsky *,[1], Dmitry V. Chalikov [2], Hendrik L. Tolman *,[1]

Ocean Modeling Branch, Environmental Modeling Center, National Centers for Environmental Prediction, NWS, NOAA, 5200 Auth Road, Camp Springs, MD 20746, USA

Abstract

A new generic approach to improve computational efficiency of certain processes in numerical environmental models is formulated. This approach is based on the application of neural network (NN) techniques. It can be used to accelerate the calculations and improve the accuracy of the parameterizations of several types of physical processes which generally require computations involving complex mathematical expressions, including differential and integral equations, rules, restrictions and highly nonlinear empirical relations based on physical or statistical models. It is shown that, from a mathematical point of view, such parameterizations can usually be considered as continuous mappings (continuous dependencies between two vectors). It is also shown that NNs are a generic tool for fast and accurate approximation of continuous mappings and, therefore, can be used to replace primary parameterization algorithms. In addition to fast and accurate approximation of the primary parameterization, NN also provides the entire Jacobian for very little computation cost.

Three successful particular applications of the NN approach are presented here: (1) a NN approximation of the UNESCO equation of state of the seawater (density of the seawater); (2) an inversion of this equation (salinity of the seawater); and (3) a NN approximation for the nonlinear wave–wave interaction. The first application has been implemented in the National Centers for Environmental Prediction multi-scale oceanic forecast system, and the second one is being developed for wind wave models.

The NN approach introduced in this paper can provide numerically efficient solutions to a wide range of problems in environmental numerical models where lengthy, complicated calculations, which describe physical processes, must be repeated frequently.

Keywords: Neural networks; Oceanic model; Wave model; Equation of state; Nonlinear interaction; Parameterization of physics; EOF

☆ OMB Contribution No. 215.

* Corresponding authors. Tel.: +1-301-763-7262; fax: +1-301-763-8545.

E-mail address: vladimir.krasnopolsky@noaa.gov (V.M. Krasnopolsky).

[1] SAIC/GSO.

[2] UCAR project scientist.

1463-5003/02/$ - see front matter
PII: S1463-5003(02)00010-0

1. Introduction

Any atmospheric or oceanic numerical model is based on a set of prognostic and diagnostic differential equations together with additional equations required to obtain a mathematically closed system. Such a system, in principle, can then be solved to predict the evolution of the environment in time if the initial conditions and any required external boundary conditions are prescribed. Even though the forecast problem may now be considered solvable in a theoretical sense, in the real world of running operational forecast models, it is necessary to deal with practical aspects of available computational resources and minimize the computer time taken to produce a forecast by introducing certain simplifications in the system for the following reasons.

The numerical model contains coefficients that appear in the dynamical equations, such as turbulence coefficients representing the unresolvable subgrid scale processes, which need to be parameterized in terms of the dependent variables. Also, implicitly contained in the system are processes that deal with model physics such as radiation, convection, etc., which need to be parameterized. Accurate treatments of such parameterizations generally require computations involving complex mathematical expressions, which may include differential and integral equations, rules, restrictions, highly nonlinear empirical expressions, etc. that are developed based on physical or statistical models. The complex mathematical formulations of these processes require considerable computational resource.

For example, a spectral atmospheric model with a well-developed description of physics and subgrid scale parameterizations may spend up to 70% of calculation time for simulating these processes (Estrade et al., 2001). The long-wave radiative code requires $>10\%$ of the computing time in the European Center for Medium-Range Weather Forecast general circulation model (Chevallier et al., 1998) and in the National Centers for Environmental Prediction (NCEP) global model. This percentage would be considerably larger if the long-wave radiative variables in both models were updated every time step (in NCEP model they are updated every 3 h only).

In modern high-resolution ocean models the estimation of the full UNESCO equation of state to compute the seawater density, represented by an empirically derived highly nonlinear equation relating density to pressure, salinity, and temperature, takes a very significant amount of the total computational effort. In addition, most forecast models include data assimilation procedures as an integral part of the forecast system to improve the initial conditions of the model. When dealing with ocean models, most often the data assimilation consists of assimilating surface and subsurface temperature observations to correct the model's thermal field. This temperature correction automatically makes it necessary to adjust the salinity field in the ocean model in order to avoid gravitational instabilities in the water column. This requires inverting the complicated oceanic equation of state, which makes the computational effort even more time consuming than the forward problem of computing the density itself. Another example where intensive computational is needed in a forecast model is the calculation of the land surface temperature using a set of equations describing the atmospheric boundary layer and physical processes in the soil. Yet another example of intensive computational problem in forecast models is the wind wave forecasting problem in which an exact calculation of the nonlinear wave–wave interactions using the formulation of Hasselmann (1963) takes a prohibitively long time.

In this paper, we use the term "parameterization" for convenience to represent in general all the computationally expensive and complex mathematical formulations involved in forecast systems, a few examples of which have been mentioned above.

In view of the constraints imposed on the available computer resources, the calculation time allowed for each parameterization is strictly limited in most operational forecast models. Hence, very often it is found necessary to use simplified forms of these complex representations in carrying out the time integrations in a forecast model, thereby sacrificing accuracy of forecasts to a certain extent. For example, the nonlinear wave interactions in a wave forecast model are replaced by a simplified discrete interaction approximation (DIA) (see Hasselmann et al., 1985). Similarly, simplified fast parameterizations of physics are used in many parts of atmospheric and oceanic models. In most of these cases, accurate physical models have been developed, but they cannot be used because they are too expensive computationally. Often simplified (even oversimplified due to computational efficiency requirements) parameterizations are obtained, for example, by neglecting higher order terms of perturbation theory, by using empirical approximations, or simply by neglecting the effects, which complicate the calculations. It is common in many parameterization schemes that the number of input and output variables is relatively small, whereas the volume of internal calculations is large. Hence, most often the specific parameterization is a result of a compromise between accuracy and computational efficiency with an (sometimes) unpredictable effect on the forecast.

Improvements in forecast modeling can be achieved not only by improving the representation of such parameterizations as our understanding of the underlying physical processes increases but also by improving our ability to compute these parameterizations accurately within the constraints imposed by the available computer resources.

In this paper we present some of the problems dealing with physical parameterizations and their computations from a different (formal mathematical) point of view, namely that of improving the computational efficiency of available algorithms. We propose a generic approach, which is based on developing fast and accurate parameterizations of physics by approximating solutions of exact physical models using neural networks (NNs). From this formal point of view an exact (best known) physical model representing a physical process performs a continuous (or almost continuous) conversion of an input vector of parameters, $X = \{x_1, x_2, \ldots, x_n\}$, $X \in \mathfrak{R}^n$ into an output vector of parameters, $Y = \{y_1, y_2, \ldots, y_n\}$, $Y \in \mathfrak{R}^m$. A worst scenario includes a continuous dependence accompanied by several discontinuities. Thus, each output parameter y_i is in general a continuous function of multiple inputs variables $x_1, x_2, \ldots, x_n$ (input vector X). Symbolically this input–output dependence can be written as

$$Y = F(X), \quad X \in \mathfrak{R}^n, \quad Y \in \mathfrak{R}^m. \tag{1a}$$

If X and Y are related through a cause and effect principle, the forward parameterization, Eq. (1a), can be derived from first principles. If the inverse dependence

$$X = f(Y), \quad X \in \mathfrak{R}^n, \quad Y \in \mathfrak{R}^m \tag{1b}$$

is required in a numerical model, the inverse problem should be solved, which implies that Eq. (1a) should be inverted. A solution of the inverse problem (1b) or an inverse parameterization provides each output parameter x_i as a continuous function of multiple input variables $y_1, y_2, \ldots, y_n$

(vector Y is an input vector now). Forward, Eq. (1a), and inverse, Eq. (1b), parameterizations represent the same mathematical object—a continuous mapping which is a continuous relationship between two vectors. Usually these input/output relationships are highly complex and nonlinear, but continuous (or have a finite number of discontinuities), for physical processes taken into account in atmospheric, oceanic, and wave models. Different components y_i of the output vector Y may demonstrate different types of nonlinear dependencies on the input vector X. Moreover, for the same component y_i, the type of nonlinear behavior may be different in different parts of the X domain. When we approximate (1a) or (1b), we also usually do not know in advance what kind of nonlinearity to expect; therefore, we need a flexible, self-adjustable approach that can accommodate various types of nonlinear behavior and represent a broad class of nonlinear mappings. The NN technique is such a generic mathematical tool (see Appendix A). Hence, if exact solutions to these complex relationships are calculated, however expensive the computational efforts may be, these solutions can be used by the NNs to produce fast and accurate approximations for continuous mappings. In this approach the costly exact calculation of the physics needs to be performed only once and "off line" (outside the model) to enable the development of the fast and accurate approximation. After that only this fast and accurate approximation will be used to calculate the physics (coefficients of differential equations) "on line" in a numerical model during the integration process.

There are many different types of the NNs (e.g., Haykin, 1994). The multi-layer perceptron (MLP) is one, which is well suited for approximation of continuous mappings (Funahashi, 1989). In our applications we used only this type of the NNs, and when we use the term NN in this paper we mean the MLP. Basic ideas of the NN technique (MLP) are introduced in Appendix A. Here, several main properties of NNs are listed which make them a very suitable generic tool for nonlinear mapping (and, therefore, for fast parameterization of physics). Some of these properties are introduced in Appendix A; others are described in the literature (e.g., Bishop, 1995; Haykin, 1994; Ripley, 1996).

- NNs are able to accurately approximate any continuous nonlinear mapping (even with finite number of discontinuities).
- While training the NN is often time consuming, its application is not. After the training is finished (it is usually performed only once), each application of the trained NN is an estimation of (A.3) with known weights and biases, which is practically instantaneous (several tens of floating point additions and multiplications).
- NNs are analytically differentiable, so the calculation of entire Jacobian matrix is cheap.
- NNs are well suited for parallel processing (Chen, 1996) (all neurons in the same layer are completely independent and can be evaluated simultaneously).

In their pioneering works Chevallier et al. (1998, 2000) considered a particular case of the generic problem formulated above—the long-wave atmospheric radiation—and applied a NN technique to solve this specific problem. In this paper we generalize their approach and apply it to solve several problems in oceanic and wave numerical models. In Section 2 we present two (forward and inverse) parameterizations for oceanic models and in Section 3 a parameterization for wind wave models developed using NNs, and, in Section 4, formulate conclusions. In Appendix A we demonstrate that NNs are a generic, fast, and accurate tool for approximating any

continuous mapping, which allows using NNs for efficient calculation of the parameterizations of physics used in numerical models.

2. Oceanic applications: neural networks for efficient calculation of sea water density or salinity from the UNESCO equation of state

Here we apply a NN technique to two related problems in the fast calculation of physics in oceanic modeling and data assimilation. (i) In most ocean models, the UNESCO International Equation of State (UES) for seawater (e.g., UNESCO, 1981) is used for the calculation of the density at each point of a three-dimensional (3D) grid using a relatively small time step. The frequency of updating the density depends on specifics of the model. For example, if the model explicitly describes external waves, the time step for recalculating the density should not be more than several times larger than the global time step. Hence, for high-resolution models, the solution of this equation consumes a significant part of the overall computation time. (ii) In the data assimilation process, assimilation of temperature alone, without making corresponding adjustments to salinity, in ocean models, which employ the full equation of state, can lead to problems of gravitational instabilities (Woodgate, 1998; Chalikov et al., 1998). To adjust the salinity, we need to calculate the salinity from UES as a function of temperature, density and depth (or pressure), i.e. solve an inverse problem in many points. Numerical inversion of the UES is an iterative procedure, which can consume several orders of magnitude more time than solving of the UES itself.

The UES for seawater gives the following expression for the density anomaly δ_ρ ($\mathrm{kg\,m^{-3}}$) as described by Fofonoff and Millard (1983),

$$\begin{aligned} \delta_\rho(T,S,P) &= \rho(T,S,P) - 1000, \\ \rho(T,S,P) &= \frac{\rho(T,S,0)}{1 - \frac{P}{K(T,S,P)}}, \end{aligned} \tag{2}$$

where ρ is the density of seawater in $\mathrm{kg\,m^{-3}}$, T is the temperature in °C, S is the salinity in practical salinity units (psu), P is the pressure, and $K(T,S,P)$ is a bulk modulus.

The UES (2) is empirically based and given over a 3D domain $D = \{-2 < T < 40\ ^\circ\mathrm{C}, 0 < S < 40$ psu, and $0 < P < 10{,}000$ decibars$\}$. This domain represents all possible combinations of T, S, and P, which are globally encountered. Mathematically, the functions $\rho(T,S,0)$ and $K(T,S,P)$ are represented by multi-dimensional polynomials and, as a result, the density (2) is a ratio of two 3D polynomials which contain more than 40 parameters.

The UES has two major drawbacks when it is applied in the context of ocean modelling. First is its cumbersome form. In ocean models the UES is used for calculating the density at each point of a 3D grid for each time step. For modern high-resolution short term forecast models, like the new NCEP multi-scale ocean forecast system (MSOFS), which uses short time integration steps to avoid instabilities due to internal waves, the solution of the UES consumes a significant part (up to 40%) of the overall computation time. Second, it is not a simple matter using the UES to obtain solutions for salinity, since this solution represents an inverse dependence.

The UES determines the density field from observed temperature, salinity, and pressure to within a standard error of approximately 0.009 $\mathrm{kg\,m^{-3}}$; however, due to variations in the composition of dissolved salts, the uncertainty in the density of natural seawater is of the order of 0.05 $\mathrm{kg\,m^{-3}}$ (Apel, 1987). The UES is usually applied in numerical models in combination with an approximate hydrostatic pressure-depth relationship of the following form:

$$P = g \int_{-H}^{0} \rho(T, S, Z)\, \mathrm{d}Z, \tag{3}$$

where g is the acceleration of gravity. This linear approximation neglects the dependence of g on latitude and the nonlinear terms in the dependence of Z on P (Fofonoff and Millard, 1983) which introduce additional uncertainties in the calculation of $\rho(T, S, Z)$ of about 0.05–0.1 $\mathrm{kg\,m^{-3}}$.

Taking these uncertainties into account, it does not make sense to use parameterization with higher accuracy in numerical ocean models if accuracy and computing time grow up together. This is why the accuracy of about 0.1 $\mathrm{kg\,m^{-3}}$ was selected as the expected accuracy for the NN parameterization. The accuracy of the NN parameterization for salinity expressed in terms of density was also expected to be of about 0.1 $\mathrm{kg\,m^{-3}}$.

The UES defines two relationships (second relationship for salinity through inversion),

$$\rho = \rho(T, S, Z), \tag{4a}$$

$$S = S(T, \rho, Z), \tag{4b}$$

which are continuous mappings (degenerated mappings because one-dimensional (1D) vectors are on the left). The NN technique can be applied to approximate these mappings (see Appendix A and also Krasnopolsky et al., 2000). To create a training set for these NN parameterizations in the 3D domain D, 4000 points (T_i, S_i, Z_i) were generated on a grid. The UES was used to estimate the density of seawater, ρ_i, for each point. This simulated data set $\{\rho_i, T_i, S_i, Z_i\}$ was used in order to train the NNs to extract density and salinity. NNs with three nonlinear neurons in one hidden layer and one linear neuron in the output layers were selected for training. The Broyden–Fletcher–Goldfarb–Shanno algorithm (Dennis and Schnabel, 1983) with the Bayesian regularization (MacKay, 1992) was used for training. No pruning was performed because of the minimal size of the NNs. Two NN parameterizations were obtained (see Fig. 1):

$$\rho = \rho_{\mathrm{NN}}(T, S, Z), \tag{5a}$$

$$S = S_{\mathrm{NN}}(T, \rho, Z), \tag{5b}$$

where both ρ_{NN} and S_{NN} are expressed by Eq. (A.3).

Derivatives (Jacobian matrix) shown in Fig. 1 as additional NN outputs are not actual outputs, which are trained during the NN training; they are calculated analytically through direct differentiating Eq. (A.4). The NN parameterization (5a) for the density is about two times faster then the UES. The calculation of the Jacobian matrix with the NN parameterization requires an additional time, which is about 70% of time required for the calculation of density. Of course, these estimates are approximate; they depend on many parameters like programming language, compiler, optimization, computer, etc. The NN parameterization (5b) for the salinity is several hundred times faster than an iterative numerical inversion of the UES; the time required for the

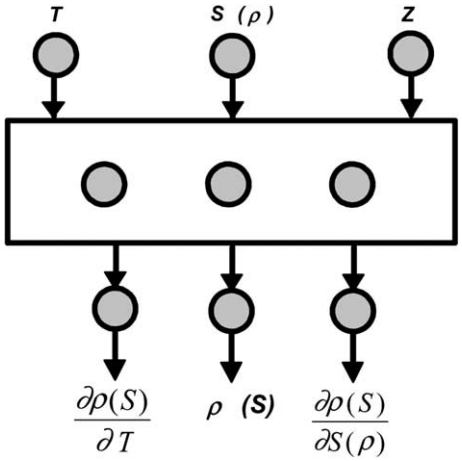

Fig. 1. Schematic representation of NN parameterizations for the density and salinity of seawater.

numerical inversion of the UES (rate of conversion of the iterations) varies significantly depending on inversion algorithm and on the choice of the initial approximation for the salinity.

To evaluate the accuracy of the NN approximation (5a) and (5b), 16,000 points were generated on a grid, which does not overlap with the training grid, within the domain D. The density of sea water calculated from the UES (2) was compared to that calculated from the NN, $\rho_{\mathrm{NN}}(T,S,Z)$, using (5a). These comparisons are presented in Tables 1 and 2. Table 1 shows statistics for the density anomaly, $\delta = \rho(T,S,Z) - 1000$, for seawater, generated by the UES (2) and by the NN (5a). Table 2 shows several statistical measures of the differences between the UES and the NN estimates for density. In terms of the bias and the RMS differences, the NN results for density clearly satisfy the criterion mentioned above; both the bias and the RMS values do not exceed the uncertainties indicated there and are less than 0.1 $\mathrm{kg\,m^{-3}}$.

Fig. 2 shows the difference, $\rho_{\mathrm{UES}} - \rho_{\mathrm{NN}}$, as a function of salinity and temperature at the surface layer of the ocean. This figure demonstrates that maximum errors occur at the combination of these parameters (low temperature and low salinity), which are never encountered in the ocean.

Table 1
Statistics for the density anomaly $\delta = \rho(T,S,Z) - 1000$

Expression	Min δ	Max δ	Mean δ	σ_δ
UES (2)	−6.02	74.32	31.04	16.33
NN (5a)	−5.98	74.38	31.04	16.33

Minimum, maximum, and mean values together with the standard deviations σ_δ are shown ($\mathrm{kg\,m^{-3}}$).

Table 2
Minimum, maximum, and mean differences (i.e., the biases), $\varepsilon = \rho_{\mathrm{NN}} - \rho_{\mathrm{UES}}$, and the RMS differences, all expressed in $\mathrm{kg\,m^{-3}}$

Min ε	Max ε	Bias	RMS
−0.25	0.12	0.00	0.04

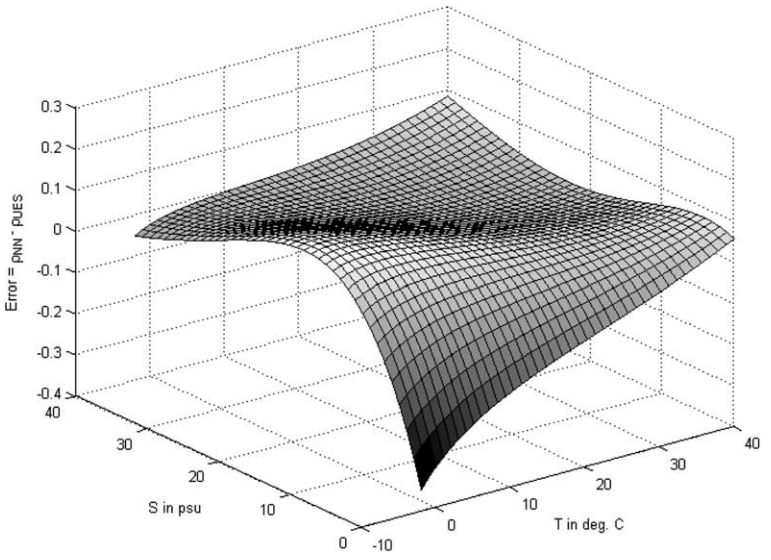

Fig. 2. Error of the NN-derived density ($\rho_{NN} - \rho_{UES}$ in kg m^{-3}) as a function of the temperature T and salinity S at the ocean surface ($z = 0$ m).

To evaluate the errors in using the NN approach to estimate the salinity, we used the same 16,000 points (ρ_i, T_i, S_i, Z_i) which were used for estimating the density. Initially, the NN for S_{NN} (5b) was applied to calculate a new salinity, s_i, using the corresponding values (T_i, ρ_i, Z_i). Then the differences $(S_i - s_i)$ were utilized to estimate the accuracy of the NN-derived salinities (first line in Table 3). To further evaluate the quality of the NN-derived salinities, the UES was applied again, this time to the triad (T_i, s_i, Z_i) to recalculate the density of seawater, ρ'_i. If the NN-obtained values for salinity were perfect, then the density, ρ'_i, would be equal to ρ_i. The differences between these two values, $(\rho_i - \rho'_i)$, were then used to further estimate the accuracy of the salinity-trained NN in terms of the density (second line in Table 3).

Table 3 shows that the NN estimates of salinity (5b) have an RMS error of 0.1 psu. In terms of the related error in density, this accuracy corresponds to an RMS error of 0.08 kg m^{-3}, which again does not exceed the uncertainties discussed above.

A substantial additional acceleration of calculations may be achieved by use of differential increments of density, temperature, and salinity to substantially reduce the computational burden via replacement of the calculations of density per se by calculations of its total differential

$$\Delta\rho = \frac{\partial\rho}{\partial T}\Delta T + \frac{\partial\rho}{\partial S}\Delta S, \tag{6}$$

Table 3
Accuracies of the salinities estimated by the NN in terms of salinity and density

Units	Min error	Max error	Mean error	RMS error
psu	−0.33	0.85	0.00	0.10
kg m^{-3}	−0.27	0.71	0.00	0.08

Minimum, maximum, and mean errors together with the RMS errors are presented.

where ΔT and ΔS are increments of T and S, $\partial\rho/\partial T$, and $\partial\rho/\partial S$ are functions of T, S, and z. Hence, we extend our approach to estimate these quantities also. In this approach, after the density and its derivatives are calculated, the Eq. (6) is used during several (usually several tens) steps of integration to estimate the new density. Then the densities and its derivatives are recalculated, using the UES or NN approximation of the UES, to update the estimated values obtained using (6). Because of these periodical updates and due to the fact that the increments ΔT and ΔS are usually small, accuracy requirements for derivatives in this case are not stringent. As a result, the derivatives in (6) can be represented at fixed depths by low-order NN approximation. The derivatives $\partial\rho/\partial T$ and $\partial\rho/\partial S$ (and $\partial\rho/\partial z$) can be accurately calculated from the NN ρ_{NN}, Eq. (5a), using Eq. (A.4) (see also Fig. 1). The relative errors in $\partial\rho/\partial T$ do not exceed 5% and the errors in $\partial\rho/\partial S$ do not exceed 1%. Eq. (6) can be reduced to

$$\Delta\rho = \frac{\partial\rho_{\mathrm{NN}}}{\partial T}\Delta T + \frac{\partial\rho_{\mathrm{NN}}}{\partial S}\Delta S. \tag{7}$$

The estimation of the density using Eq. (6) requires five calculations of the UES (in that case the Jacobians are computed be centered finite differences). The estimation of the density using Eq. (7) requires one estimate of the NN (4a) and its derivatives, which is about five times faster than (6). If $\partial\rho_{\mathrm{NN}}/\partial z$ is also used in (7) for vertical integration, the gain in the speed of calculations due to the use of the NN reaches about seven times.

This use of NN and its derivatives has been shown to accelerate model performance significantly. Table 4 estimates an absolute accuracy in calculating $\Delta\rho$ from Eq. (7), as compared with $\Delta\rho$ calculated using UES with the same ΔT and ΔS. In this case bias is negligible and the RMSE $< 0.5\%$.

Based on results discussed above, we accepted the following scheme for calculating the density in our MSOFS at NCEP. At the beginning of each run of the model, the density and its derivatives are calculated using the NN approximation. Then for each time step assigned for updating the density, the density increment values are calculated using Eq. (7), with the initial density and its derivatives kept unchanged. The frequency of updating (recalculating) the density and its derivatives depends on specific properties of the model and region to which the model is applied. For the global version of the NCEP ocean model with the horizontal resolution of about 80 km the density and its derivatives are updated once in every 90 time integration steps of the external mode (i.e., every 180 min). Fig. 3 shows that using Eq. (7) the calculations may be accelerated about 10 times with the error in the density calculations not exceeding the natural uncertainty of about 0.1 $\mathrm{kg\,m^{-3}}$. The figure presents results of calculations performed with the MSOFS at NCEP when the number of integration steps between updated of the density and its derivatives in (7) varied. The computational expense of calculating density has decreased by an order of magnitude as a result of using the NN approximation in combination with Eq. (7).

Table 4
Mean, standard deviation (σ), and mean differences (i.e., biases) and the standard deviations (SD) for the differences for $\Delta\rho$ calculated using Eqs. (7) and (6) with $\Delta T = 0.1$ and $\Delta S = 0.1$

	Mean	σ	Bias	SD
$\Delta\rho$	0.052	0.012	−0.00001	0.0002

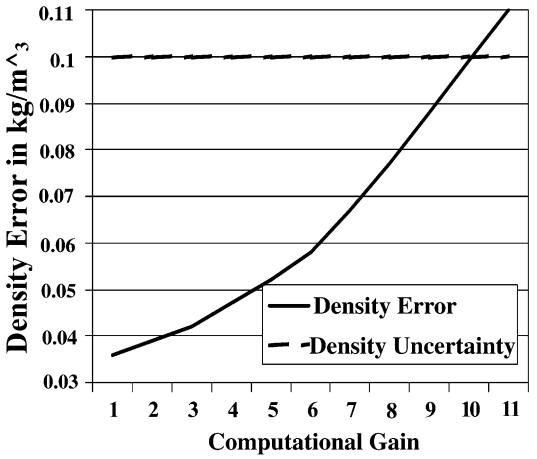

Fig. 3. Error in the density as a function of computational gain when Eq. (7) is used for density calculations in the MSOFS at NCEP.

3. Wave application: a neural network approximation for nonlinear interactions in wind wave models

Ocean wind wave modeling for hindcast and forecast purposes has been at the center of interest for many decades. Numerical prediction models are generally based on a form of the spectral energy or action balance equation

$$\frac{\mathrm{D}F}{\mathrm{D}t} = S_{\mathrm{in}} + S_{\mathrm{nl}} + S_{\mathrm{ds}} + S_{\mathrm{sw}}, \tag{8}$$

where F is the spectrum, S_{in} is the input source term, S_{nl} is the nonlinear interaction source term, S_{ds} is the dissipation or 'whitecapping' source term, and S_{sw} represents additional shallow water source terms. The JONSWAP study (Hasselmann et al., 1973) identified the active role of the nonlinear interactions in wave growth. The SWAMP study (SWAMP Group, 1985) then identified the need for explicit modeling of S_{nl} in wave models. State-of-the-art or so-called third-generation wave models therefore explicitly model this source term.

In its full form (e.g., Hasselmann and Hasselmann, 1985), the calculation of the interactions S_{nl} requires the integration of a six-dimensional Bolzmann integral:

$$\begin{aligned} S_{\mathrm{nl}}(\vec{k}_4) = T \otimes F(\vec{k}) = \omega_4 \int G(\vec{k}_1,\vec{k}_2,\vec{k}_3,\vec{k}_4)\cdot\delta(\vec{k}_1+\vec{k}_2-\vec{k}_3-\vec{k}_4)\cdot\delta(\omega_1+\omega_2-\omega_3-\omega_4) \\ \times [n_1\cdot n_3\cdot(n_4-n_2)+n_2\cdot n_4\cdot(n_3-n_1)]\,\mathrm{d}\vec{k}_1\,\mathrm{d}\vec{k}_2\,\mathrm{d}\vec{k}_3, \\ n(\vec{k}) = \frac{F(\vec{k})}{\omega}, \quad \omega^2 = g\cdot k\cdot\tanh(kh), \end{aligned} \tag{9}$$

where the complicated coupling coefficient G contains moving singularities (Hasselmann, 1963). This integration requires roughly 10^3–10^4 times more computational effort than all other aspects of the wave model combined. Present operational constraints require that the computational effort for the estimation of S_{nl} should be of the same order of magnitude as the remainder of the wave model. This requirement was met with the development of the DIA (Hasselmann et al.,

1985). The development of the DIA allowed for the successful development of the first third-generation wave model WAM (WAMDI Group, 1988; Komen et al., 1994). More than a decade of experience with the WAM model and its derivatives has identified shortcomings of the DIA. The DIA tends to unrealistically increase the directional width of spectra, has a systematic spurious impact on the shape of the spectrum near the spectral peak frequency, and has a much too strong signature at high frequencies. In present third-generation wave models, these deficiencies can be countered at least in part by the dissipation source term S_{ds}, which is generally used for tuning the energy balance in the Eq. (8). Although this approach gives good results, it is counterproductive, because it prohibits development of dissipation source terms based on solid physical considerations. With our increased understanding in the physics of wave generation and dissipation, this becomes an even bigger obstacle impeding further development of third-generation wave models.

Considering the above, it is of crucial importance for the development of third-generation wave models to develop *an economical yet accurate approximation* for S_{nl}. Here, we explore a Neural Network Interaction Approximation (NNIA) to achieve this goal (see also Krasnopolsky et al., 2001). NNs can be applied here because the nonlinear interaction (9) is essentially a nonlinear mapping (symbolically represented in Eq. (9) by T) which relates two vectors (two-dimensional (2D) fields in this case). Thus, the nonlinear interaction source term can be considered as a nonlinear mapping between a spectrum F and a source term S_{nl},

$$S_{nl} = T(F), \tag{10}$$

where T is the exact nonlinear operator given by the full Bolzmann interaction integral (9) (Hasselmann and Hasselmann, 1985; Resio and Perrie, 1991). Discretization of S and F (as is necessary in any numerical approach) reduces (10) to continuous mapping of two vectors of finite dimensions. Modern high-resolution wind wave models use discretization on a 2D grid which leads to dimensions of S and F vectors of order of $N > 600$ (Tolman, 1999). It seems unreasonable to develop a NN approximation of such a high dimensionality (more than 600 inputs and outputs). Moreover, such a NN will be grid dependent.

In order to reduce the dimensionality of the NN and convert the mapping (10) to a continuous mapping of two finite vectors independent on the actual spectral discretization, the spectrum F and source function S_{nl} are expanded using systems of 2D functions each of which (Φ_i and Ψ_q) creates a complete and orthogonal 2D basis

$$F \approx \sum_{i=1}^{n} x_i \Phi_i, \qquad S_{nl} \approx \sum_{q=1}^{m} y_q \Psi_q, \tag{11}$$

where for x_i and y_q we have

$$x_i = \int\int F \Phi_i, \qquad y_q = \int\int S_{nl} \Psi_q, \tag{12}$$

where the double integral identifies integration over the spectral space. Because both sets of basis functions $\{\Phi_i\}_{i=1,\ldots,n}$ and $\{\Psi_q\}_{q=1,\ldots,m}$ are complete, increasing n and m in (11) improves the accuracy of approximation, and any spectrum F and source function S_{nl} can be approximated by (11) with a required accuracy. Substituting (11) into Eq. (10) we can get

$$Y = T(X) \tag{13}$$

which represents a continuous mapping of the finite vectors $X \in \Re^n$ and $Y \in \Re^m$, and where T still represents the full nonlinear interaction operator. As described in Appendix A, this operator can be approximated with a NN with n inputs and m outputs and k neurons in the hidden layer

$$Y = T_{\mathrm{NN}}(X). \tag{14}$$

The accuracy of this approximation (T_{NN}) is determined by k, and can generally be improved by increasing k (see Appendix A).

To train the NN approximation T_{NN} of T, a training set has to be created that consists of pairs of vectors X and Y. To create this training set, a representative set of spectra F_p has to be generated with corresponding (exact) interactions $S_{\mathrm{nl},p}$ using Eq. (9). For each pair $(F, S_{\mathrm{nl}})_p$, the corresponding vectors $(X, Y)_p$ are determined using Eq. (12). These pairs of vectors are then used to train the NN to obtain T_{NN}. After T_{NN} has been obtained by training, the resulting NNIA algorithm consists of three steps:

- Decompose the input spectrum, F, by applying Eq. (12) to calculate X.
- Estimate Y from X using Eq. (14).
- Compose the output source function, S_{nl}, from Y using Eq. (11).

A graphical representation of the NNIA algorithm is shown in Fig. 4.

The above describes the general procedure for developing an NNIA. Development of an actual NNIA requires the following steps:

- Select basis functions Φ_i and Ψ_q and the number of each (n, m).
- Design a NN topology (number of neurons k).
- Construct a representative training set.
- Select training strategies.

The first three points all have a significant impact on both accuracy and economy of a NNIA. Unfortunately, there is no pre-defined way to tackle these issues. It is therefore unavoidable that

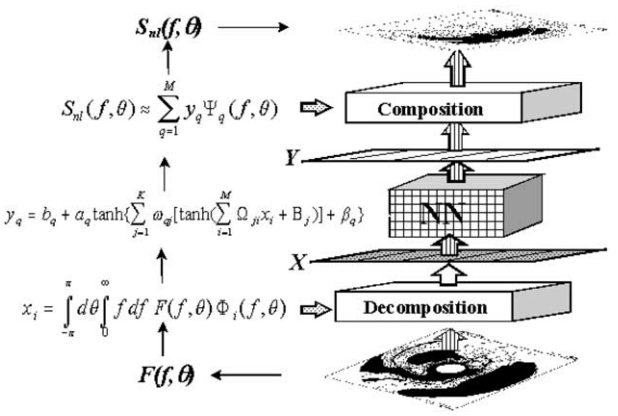

Fig. 4. Graphical representation of the NNIA algorithm.

the development of a NNIA involves much iteration. The first requirement for a NNIA to be potentially useful in operational wave modeling is that the exact interactions S_{nl} are closely reproduced for computational costs comparable to that of the DIA. The following shows the potential of this approach with the design of a simple ad-hoc NNIA.

To address the basic feasibility of a NNIA, we have considered a NNIA to estimate the nonlinear interactions $S_{\mathrm{nl}}(f,\theta)$, as a function of frequency f and direction from the corresponding spectrum $F(f,\theta)$. We also consider deep water only here. To train and test this NNIA, we used a set of about 20,000 simulated realistic spectra for $F(f,\theta)$, and the corresponding exact estimates of $S_{\mathrm{nl}}(f,\theta)$ (Van Veldder et al., 2000). Simulation has been performed using a generator that calculated a spectral function as a composition of several Pierson–Moskowitz spectra for different peak frequencies, where each peak is oriented randomly in $[0,2\pi]$ interval. Comparison of simulated spectra with spectra simulated by the WAVEWATCH model (Tolman, 1999; Tolman and Chalikov, 1996) shows that this approach allowed us to simulate reasonably realistic and complicated spectra describing a broad range of wave systems. Spectra with four peaks were used in calculations below. Separate data sets have been generated for training and validation.

As is common in parametric spectral descriptions, we choose separable basis functions where frequency and angular dependence are separated. For Φ_i this implies

$$\Phi_i(f,\theta) \Rightarrow \Phi_{ij} = \phi_{f,i}(f)\phi_{\theta,j}(\theta). \tag{15}$$

A similar separation is used for Ψ_q. Considering the strongly suppressed behavior of F and S_{nl} for $f \to 0$, and the exponentially decreasing asymptotic behavior for $f \to \infty$, generalized Laguerre's polynomials (Abramowitz and Stegun, 1964) are used to define φ_f and ψ_f. Considering that no directional preferences exist in F and S_{nl}, Fourier decomposition is used for φ_θ and ψ_θ. The number of base functions is chosen to be $n = 51$ and $m = 64$ to keep the accuracy of approximation for F on average better than 2% and for S_{nl}—better than 5–6%. A NN with $n = 51$ inputs, $k = 30$ nonlinear neurons in one hidden layer and $m = 64$ nonlinear neurons in the output layer was selected, which allows a satisfactory NN approximation of the mapping (13) using (14).

Table 5 compares three important statistics for the source function RMS errors (with respect to exact solution) calculated using DIA and NNIA for 10,000 spectra (independent validation set). The NNIA is nearly twice as accurate as DIA.

Fig. 5 shows mean RMSE as functions of the frequency f (left) and the angle θ (right). It also illustrates the improvement of the NNIA accuracy by increasing the number of neurons, k, in the hidden layer from 20 to 30. Numbers in Table 5 correspond to a NNIA with 30 neurons in the hidden layer (51:30:64). Fig. 6 shows 1D integrated spectra.

Table 5
RMSE statistics for 10,000 S_{nl}

	Mean RMSE	σ_{RMSE}	Max RMSE
DIA	0.0133	0.0111	0.104
NNIA	0.0068	0.0063	0.065

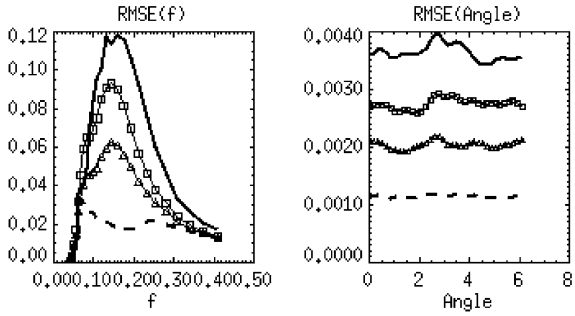

Fig. 5. RMSE as functions of frequency f and angle (averaged over entire test set). Dashed line—error of approximation (lower bound for all other errors), solid line—DIA, line with squares—NNIA ($n = 51$:$k = 20$:$m = 64$), and line with triangles—NNIA (51:30:64).

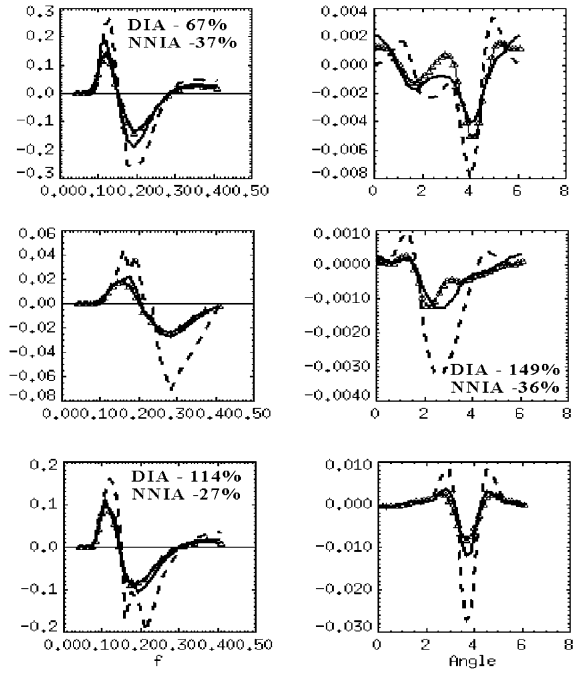

Fig. 6. Three pairs (one row in the figure corresponds to a pair). Each pair shows the source function $S_{\mathrm{nl}}(f)$ integrated over θ as function of frequency (left column), and the source functions $S_{\mathrm{nl}}(\theta)$ integrated over f as function of angle (right column) from the validation data set. Thick solid curves correspond to the exact S_{nl}. Dashed curves correspond to DIA of S_{nl}. Curves with triangles correspond to the NNIA estimate of S_{nl}. Numbers inside the panels show relative errors of DIA and NNIA (in %) with respect to the exact solution.

The results in Fig. 6 are fairly representative for the validation data set. In general, the NNIA reproduces the exact S_{nl} accurately. Even when spurious oscillations are present in the DIA

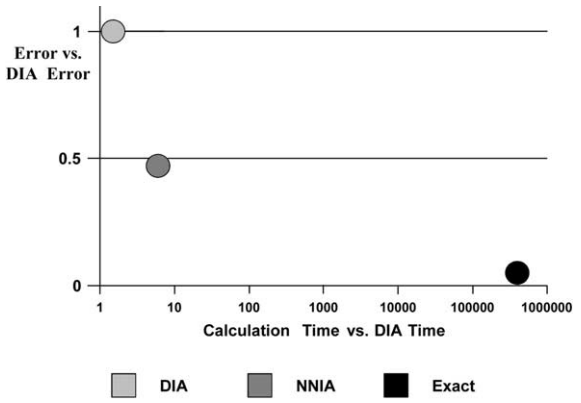

Fig. 7. Comparison of the accuracy and computational efficiency of the DIA, NNIA, and exact algorithms. The horizontal time scale is logarithmic.

spectrum (e.g., dashed lines in middle and lower panel on the left), the NNIA shows no such behavior and gives reasonable results. In general, many DIA source functions exhibit complicated behavior and spurious oscillations. Major peaks in these functions coexist with more or less random small-scale fluctuations. These fluctuations are probably an artifact produced by the simplified nature of DIA. Exact interactions are the result of averaging over a much larger number of resonant sets of wave numbers, and are therefore much smoother than the results of the DIA.

And finally, Fig. 7 compares the DIA, NNIA, and exact algorithms in terms of the accuracy and computational efficiency. Computational time (in s) corresponds to a control calculation performed on the same computer. The current preliminary version of the NNIA algorithm is twice as accurate and only about five times slower than the DIA algorithm. In the current version of the wind wave models, an algorithm that is up to 20 times slower than DIA can be accommodated; therefore, we still have enough room for further improving the accuracy of the NNIA. Considering that no optimization has yet been applied in the development of the NNIA composition and decomposition procedures, it appears reasonable to expect a final NNIA algorithm with computation requirements similar to DIA but with significantly better accuracy.

4. Conclusions

In this paper we formulated a new approach for simplifying and accelerating time-consuming calculations in environmental numerical models using NN techniques. Parameterizations of physical, chemical, and biological processes, which occur at different scales, constitute an important class of such calculations. It is shown that, from a mathematical point of view, descriptions of such processes can usually be considered as continuous mappings (continuous dependencies between two vectors). It is also shown that NNs are a generic tool for approximation

of such mappings and, therefore, can be used for fast and accurate approximation of parameterizations of such processes. They also provide the entire Jacobian for very little computational cost. Because the NN Jacobian is computationally cheap, this approach is expected to be very beneficial when used in 3D and especially in 4D variational data assimilation systems. We applied this approach to three specific problems associated with oceanic and wave modeling; however, the method can be applied to efficiently calculate some columnar physical processes in atmospheric models as well. For instance, NNs have been used for fast calculation of atmospheric long-wave radiation by Chevallier et al. (1998, 2000).

The first application considered in the paper deals with the oceanic equation of state, which is used for estimating the density and salinity of seawater in ocean circulation models. Separate NNs for density and salinity were developed using the UES as a basis. Although the estimation of density represents a forward problem, estimating salinity from the UES represents a complicated inverse problem, which has been very efficiently solved using the NN approach. The accuracy of the NN-generated densities and salinities were of the same order as those obtained directly from the UES itself. However, the time required to perform the calculations of density using the NN is several times less than that for UES. The time required for calculating salinity using the NN is several hundred times less than that required for the numerical inversion of the USE. Consequently, this approach has direct application to numerical ocean models where the equation of state must be estimated repeatedly. At NCEP, a NN equation for seawater density is currently used in the NCEP MSOFS.

The second application deals with the nonlinear wave–wave interactions in wind wave models. A prototype of the NN approximation for this interaction is presented in this work. The NNIA calculations of S_{nl} are more accurate than that from DIA by a factor of two and require roughly 4–5 times more computational effort than the DIA calculations with less than 5% of this time spent in the actual NN part of the algorithm [i.e., Eq. (A.5)]. Decomposition of the input spectra F and composing the source function S_{nl} from the NN output accounts for the rest. Having established that a NNIA has the potential of being both accurate and efficient, we intend to take the following steps towards developing a NNIA for application in operational wave models: (i) use more realistic spectra and S_{nl} calculated from them for training; (ii) optimize the NNIA by successive integration of physical properties in the basis functions, normalizing F and S_{nl}, optimizing the number of basis functions and network topology, and optimizing numerical aspects of the decomposition/compositions algorithms; (iii) expand the NNIA to arbitrary water depths, either by expanding the underlying NN or by scaling as in the DIA.

Three applications presented in the paper illustrate the strengths and limits of the NNs for the application to the fast simulation of environmental processes. The NNs are likely to be faster than original parameterizations. The simulation of environmental processes may involve a large number of inputs (i.e., several hundreds), which make the NN too complex and complicates the training. For this complexity problem, two possible solutions were developed: the input and output vectors may be expanded into a basis (e.g., the NNIA application in this paper) or a battery of smaller NNs may be used (e.g., infrared radiation application by Chevallier et al. (1998)). Finally, a cheap computation of Jacobian is one of the advantages of the NN approach. Using this Jacobian in a combination with the tangent-linear approximation like Eq. (7) can additionally accelerate calculations (e.g., the seawater density application). However, since Jacobian is not trained, it is simply calculated through direct differentiation of a trained NN. For

complex NNs the accuracy of the NN Jacobian may not be sufficient for using with the tangent-linear approximation and the approach may require some refinements.

Acknowledgements

We thank D.B. Rao for his thorough and critical review of this paper and useful comments. We thank Gerbrant Ph. Van Vledder for providing us with a code for calculating the exact nonlinear wave–wave interaction. We also thank ONR for providing partial support for the NNIA research project.

Appendix A. Neural network—a generic tool for continuous mappings

In Section 2 we showed that both forward and inverse parameterizations (Eqs. (1a) and (1b)) can be considered as continuous mappings which map a vector of input parameters, $X \in \Re^n$, to a vector of output parameters, $Y \in \Re^m$ or vice versa (for the inverse parameterization). These mappings are defined on finite discrete sets of pairs of input/output vectors X and Y, $\{X_i, Y_i\}_{i=1,\ldots,N}$.

NNs are well suited for a very broad class of such nonlinear approximations and mappings (Funahashi, 1989). Any continuous mapping with a finite number of discontinuities can be approximated. A NN is a complicated combination of uniform processing elements, nodes, units, or neurons (Bishop, 1995; Haykin, 1994; Ripley, 1996). A typical processing element is shown in Fig. 8. Each processing element usually has several inputs (components of vector X) and one output, z_j. The neuron usually consists of two parts, a linear part and a nonlinear part. The linear part calculates the inner product of the input vector X and a weight vector Ω_j (which is a column of the weight matrix Ω_{ji}), and adds a bias, B_j. The result of this linear transformation of the input vector X goes into the nonlinear part of the neuron as the argument of an activation function φ. The neuron output, z_j, can be written as

$$z_j = \phi\left(\sum_{i=1}^{n} \Omega_{ji}^T x_i + B_j\right), \quad \Omega \in \Re^{n\times m}, \ B \in \Re^m. \tag{A.1}$$

For the activation function φ, it is sufficient to be a Tauber–Wiener (nonpolynomial, continuous, bounded) function (Chen and Chen, 1995a,b). The three mostly popular activation functions are

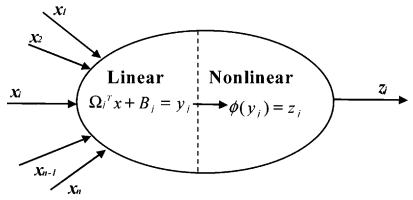

Fig. 8. Typical processing element (neuron). Linear and nonlinear parts of the neuron are shown.

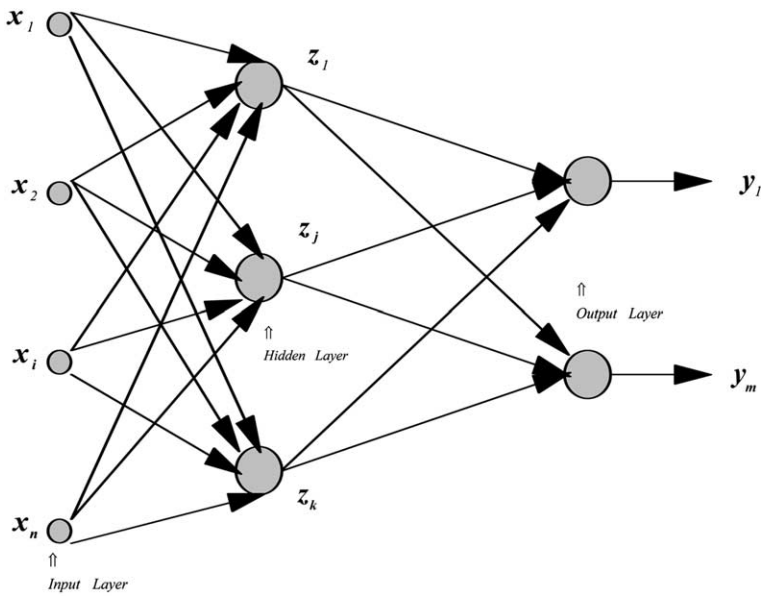

Fig. 9. MLP—feed forward, fully connected topology.

sigmoid, which is $\phi(x) = \{1 + \exp(-x)\}^{-1}$, $x \in (-\infty, \infty)$, $\phi \in (0, 1)$, hyperbolic tangent, and step function. The neuron is a nonlinear element because its output z_j is a nonlinear function of its inputs X. Neurons can be connected in many different ways into networks with complicated architectures (or topologies). The most common topology for approximating continuous mappings is the MLP, which is shown in Fig. 9. Only this type of the NNs is described here because it is sufficient for solving any continuous mapping problems. In a MLP, neurons are situated into layers. The neurons in the input layer are linear; they are simple distributors of inputs. The number of input neurons in the input layer is equal to the number of inputs (dimension of input vector X). The neurons in the output layer may be linear and/or nonlinear, depending on the problem to be solved. The number of output neurons in the output layer is equal to the number of outputs (dimension of output vector Y). The neurons in the hidden layer(s) are usually nonlinear. The number of hidden layers, the number of neurons in each hidden layer, and the type of connections between neurons and layers depend on the complexity of the problem to be solved.

The topology of the MLP shown in Fig. 9 is called feed-forward (there are no feedbacks; the data flow moves only forward) and fully connected (each neuron in a previous layer is connected to each neuron in the following one). Symbolically this mapping can be written as

$$Y = F_{\mathrm{NN}}(X), \tag{A.2}$$

where F_{NN} denotes this NN mapping. For the NN topology shown in Fig. 9, using (A.1) and assuming k neurons in one hidden layer, a linear output layer, and activation function, $\varphi(x) = \tanh(x)$, the symbolic expression (A.2) can be written down explicitly as

$$y_q = \sum_{j=1}^{k} \omega_{qj} z_j + \beta_q = \sum_{j=1}^{k} \omega_{qj} \left[\phi \left(\sum_{i=1}^{n} \Omega_{ji} x_i + B_j \right) \right] + \beta_q$$
$$= \sum_{j=1}^{k} \omega_{qj} \left[\tanh \left(\sum_{i=1}^{n} \Omega_{ji} x_i + B_j \right) \right] + \beta_q, \quad \text{(A.3)}$$

where the matrix Ω_{ji} and the vector B_j represent weights and biases in the neurons of the hidden layer; ω_{qj} and the β_q represent weights and biases in the neurons of the output layer. For some applications (e.g., see Section 2) we need to know the Jacobian matrix, whose elements are partial derivatives $\partial y_i / \partial x_j$. From (A.3) these derivatives can be calculated analytically,

$$\frac{\partial y_q}{\partial x_p} = \sum_{j=1}^{k} (1 - z_j^2) \Omega_{pj} \omega_{jq}. \quad \text{(A.4)}$$

It has been shown by many authors (e.g., Chen and Chen, 1995a,b; Hornik, 1991; Funahashi, 1989; Cybenko, 1989) that a NN with one hidden layer, like NN (A.3), can approximate any continuous mapping defined on compact sets in $\Re^n$. It means that any problem, which can be mathematically reduced to a nonlinear mapping like (1a) or (1b), can be solved using the NN represented by (A.3). NN solutions (A.3) for different problems can have different number of inputs, n, and outputs, m. They can have different numbers of neurons, k, in the hidden layer. They will also have different weights and biases in the hidden and output layers. The next and crucial problem is how to determine all these parameters.

For each particular problem, n and m are determined by the dimensions of the input and output vectors X and Y. The number of hidden neurons, k, in each particular case should be determined taking into account the complexity of the problem. The more complicated the mapping, the more hidden neurons are required (Attali and Pagès, 1997). Unfortunately, there is no universal recommendation to be given here. Usually k is determined by experience and experiment. After these topological parameters are defined, the weights and biases can be found, using a procedure, which is called NN training, which is essentially a complicated nonlinear optimization procedure. Various methods developed for nonlinear optimization (Dennis and Schnabel, 1983) can be applied for the NN training.

NN techniques can be also considered as an advanced statistical approach (Ripley, 1996). The MLP technique is very close to some advanced nonlinear regression techniques.

References

Abramowitz, M., Stegun, I.A. (Eds.), 1964. Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables. National Bureau of Standards.

Apel, J.R., 1987. In: Principles of Ocean Physics. Academic Press, New York, p. 145.

Attali, J.-G., Pagès, G., 1997. Approximations of functions by a multilayer perceptron: a new approach. Neural Networks 6, 1069–1081.

Bishop, C.M., 1995. Neural Networks for Pattern Recognition. Calderon Press, Oxford, 482 p.

Chalikov, D. et al., 1998. Revisiting the question of assimilating temperature alone into a full equation of state ocean model. Ocean Modeling 116, 13–14.

Chen, C.H. (Editor in chief), 1996. Fuzzy Logic and Neural Network Handbook. McGraw-Hill, New York.

Chen, T., Chen, H., 1995a. Approximation capability to functions of several variables, nonlinear functionals and operators by radial basis function neural networks. Neural Networks 6, 904–910.

Chen, T., Chen, H., 1995b. Universal approximation to nonlinear operators by neural networks with arbitrary activation function and its application to dynamical systems. Neural Networks 6, 911–917.

Chevallier, F. et al., 1998. A neural network approach for a fast and accurate computation of longwave radiative budget. Journal of Applied Meteorology 37, 1385–1397.

Chevallier, F. et al., 2000. Use of a neural-network-based longwave radiative transfer scheme in the EMCWF atmospheric model. Quarterly Journal of Royal Meteorological Society 126, 761–776.

Cybenko, G., 1989. Approximation by superposition of sigmoidal functions. Mathematics of Control, Signals and Systems 2, 303–314.

Dennis, J.E., Schnabel, R.B., 1983. Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Prentice-Hall, Englewood Cliffs, NJ.

Estrade, J.-F., Trémolet, Y., Sela, J., 2001. Experiments with NCEP's spectral model. In: Zwiefholer, W., Kreitz, N. (Eds.), Developments in Teracomputing—Proceedings of the Ninth ECMWF Workshop on the Use of High Performance Computing in Meteorology, Reading, UK, 13–17 November, 2000, World Scientific Publishing Co., Singapore, pp. 92–99.

Fofonoff, N.P., Millard, R.C. Jr., 1983. Algorithms for computation of fundamental properties of seawater. UNESCO technical paper in marine science 44, UNESCO.

Funahashi, K., 1989. On the approximate realization of continuous mappings by neural networks. Neural Networks 2, 183–192.

Hasselmann, K., 1963. On the non-linear energy transfer in a gravity-wave spectrum. Part 3: Evaluation of the energy flux and swell-sea interaction for a Neumann spectrum. Journal of Fluid Mechanics 15, 385–399.

Hasselmann, S., Hasselmann, K., 1985. Computations and parameterizations of the nonlinear energy transfer in a gravity wave spectrum. Part I: A new method for efficient computations of the exact nonlinear transfer integral. Journal of Physical Oceanography, 1369–1377.

Hasselmann, K. et al., 1973. Measurements of wind–wave growth and swell decay during the Joint North Sea Wave Project (JONSWAP). Ergänzungshelf zur Deutschen Hydrographischen Zeitschrift, Reihe A (8) Nr. 12, 95 pp.

Hasselmann, S. et al., 1985. Computations and parameterizations of the nonlinear energy transfer in a gravity wave spectrum. Part II: Parameterization of the nonlinear transfer for application in wave models. Journal of Physical Oceanography 15, 1378–1391.

Haykin, S., 1994. Neural Networks. A Comprehensive Foundation. Macmillan College Publishing Company, New York, 696 p.

Hornik, K., 1991. Approximation capabilities of multilayer feedforward network. Neural Networks 4, 251–257.

Komen, G.J. et al., 1994. Dynamics and Modelling of Ocean Waves. Cambridge University press, Cambridge, 532 pp.

Krasnopolsky, V.M. et al., 2000. Application of neural networks for efficient calculation of sea water density or salinity from the UNESCO equation of state. In: Proceedings, Second Conference on Artificial Intelligence, AMS, Long Beach, CA, 9–14 January, 2000, pp. 27–30.

Krasnopolsky, V.M., Chalikov, D.V., Tolman, H.L., 2001. Using neural network for parameterization of nonlinear interactions in wind wave models. In: International Joint Conference on Neural Networks, 15–19 July, 2001, Washington, DC, pp. 1421–1425.

MacKay, D.J.C., 1992. Bayesian interpolation. Neural Computation 4, 415–447.

Resio, D.T., Perrie, W., 1991. A numerical study of nonlinear energy fluxes due to wave–wave interactions, 1, methodology and basic results. Journal of Fluid Mechanics 223, 603–629.

Ripley, B.D., 1996. Pattern Recognition and Neural Networks. Cambridge University Press, Cambridge, 403 p.

SWAMP Group, 1985. Ocean wave Modeling. Plenum Press, 256 pp.

Tolman, H.L., 1999. User manual and system documentation of WAVEWATCH-III version 1.18. NOAA/NWS/NCEP/OMB technical note 166, 110 pp.

Tolman, H.L., Chalikov, D.V., 1996. Source terms in a third-generation wind wave model. Journal of Physical Oceanography 26, 2497–2518.

UNESCO, 1981. The practical salinity scale 1978 and the International Equation of State for seawater 1980. Tenth Report of the Joint Panel on Oceanographic Tables and Standards, UNESCO Technical Papers in Marine Science No. 36, UNESCO, Paris, France.

Van Veldder et al., 2000. Modelling of nonlinear quadruplet wave–wave interactions in operational coastal wave models. Abstract, accepted for presentation at ICCE 2000, Sydney.

WAMDI Group, 1988. The WAM model—a third generation ocean wave prediction model. Journal of Physical Oceanography 18, 1775–1810.

Woodgate, R.A., 1998. Can we assimilate temperature data alone into a full equation of state model? Ocean Modelling 114, 4–5.