NEURAL NETWORKS AS TOOLS TO SOLVE PROBLEMS IN PHYSICS AND CHEMISTRY.

Włodzisław Duch
Department of Computer Methods,
Nicholas Copernicus University,
ul. Grudziądzka 5, 87–100 Toruń, Poland.
e-mail: duch @ phys.uni.torun.pl

Geerd H. F. Diercksen
Max–Planck–Institute für Astrophysik
Karl–Schwarzschild–Straße 1
85740 Garching bei München, Germany
e-mail: ghd @ mpa-garching.mpg.de

(May 18, 1994)

Application of the neural network methods to problems in physics and chemistry has rapidly gained popularity in recent years. We show here that for many applications the standard methods of data fitting and approximation techniques are much better than neural networks in the sense of giving more accurate results with a lower number of adjustable parameters. Learning in neural networks is identified with the reconstruction of hypersurfaces based on a knowledge of sample points and generalization with interpolation. Neural networks use sigmoidal functions for these reconstructions, giving for most physics and chemistry problems results far from optimal. An arbitrary data fitting problem may be solved using a single-layer network architecture provided that there is no restriction on the type of functions performed by the processing elements. A simple example illustrating unreliability of interpolation and extrapolation by the typical backpropagation neural network learning of a smooth function is presented. Some results from approximation theory are quoted giving a rigorous foundation to applications requiring correlation of numerical results with a set of parameters.

I. INTRODUCTION

Neural computing and the field of neural network modeling has become very fashionable in the last decade. Availability of general neural network simulators [1] has encouraged many scientists to try these new techniques for solving their physics and chemistry problems. Therefore it is of great importance to understand what neural networks can do and when their application may lead to new results, hard to obtain with standard methods. A number of good books and review articles on neural network models [2] have appeared in recent years, unfortunately rarely giving a good mathematical perspective of relevant theories, such as the theory of statistical decisions or approximation theory.

Artificial neural networks (ANNs) are networks of simple processing elements (called "neurons") operating on their local data and communicating with other elements. Thanks to this global communication the ANN has stable states consistent with the current input and output values. The design of ANNs was motivated by the structure of a real brain, but the processing elements and the architectures used in artificial neural networks frequently have nothing in common with their biological inspiration. The weights of connections between the elements are adjustable parameters. Their modification allows the network to realize a variety of functions. In the typical case of a supervised learning a set of input and output patterns is shown to the network and the weights are adjusted (this is called "learning" or "adaptation") until the outputs given by the network are identical to the desired ones.

In principle ANN has the power of a universal computer, i.e. it can realize an arbitrary mapping of one vector space to another vector space. Since physicists and chemists deal with such problems quite often one of the applications of ANNs in these fields is to correlate parameters with some numerical results in hope that the network, given a set of examples, or a statistical sample of data points, will somehow acquire an idea of what the global mapping looks like. One of the goals of this paper is to investigate whether such a hope is justified.

ANNs are especially suitable for problems where a high error rate is acceptable, the conditions are ill-defined and the problem mathematically ill-posed. The brain has evolved to process the data from the senses and works much better at solving problems requiring perception and pattern recognition than problems involving logical steps and data manipulation. Most ANN architectures share this quality with real brains. They are not suited for the tasks that the sequential computer programs can do well, such as the manipulation of symbols, logical analysis or solving numerical problems.

The most common architecture of ANNs is of the multilayered feedforward type. The signals are propagated from the input to the output layer, each processing element being responsible for integration of the signals coming from the lower layer and influencing all processing elements of the next layer to which it is connected. If all possible connections between the consecutive layers are allowed the network is called "fully connected". In some cases it is better to use ANN that is not fully connected: reduction of the number of adjustable weights may improve not only the timing of computations for training the network but also the accuracy of learning.

Backpropagation of errors (BP) is the most commonly used algorithm to train such an ANNs [3] (for classification problems feedforward learning vector quantization and counterpropagation networks are most commonly used) [2]. Although the BP learning rule is rather universal and can be applied to a number of different architectures of neural nets a term "backpropagation net" is commonly used to designate those nets that are trained using the BP algorithm. This learning rule compares the desired output with the achieved output and error signals (differences between desired and achieved outputs) are propagated layer by layer from the output back to the input layer. The weights are changed using a gradient descent or some other minimization method in such a way, that the error should be reduced after the next presentation of the same input. Although the BP algorithm is rather slow and requires many iterations it enables the learning of arbitrary mappings and therefore it is widely used. Over 40 other learning rules for different network architectures exist and new rules are still being discovered [2].

ANNs are interesting to the physicists as an example of complex systems that are more general than the Ising or the spin glass models. From this point of view, as interesting dynamical systems, their evolution is investigated and the methods of statistical physics applied to such problems as network capacity, efficiency of various learning rules or chaotic behavior [4]. ANNs are also interesting as models of various sensory subsystems and as simplified models of the nervous system.

In this paper we are concerned only with applications of neural networks as tools that can help to solve real physical problems. The number of papers in the section "neural networks" in *Physics Abstracts* has approximately doubled comparing the 1992 and 1991 entries. This is a reflection of the enthusiasm with which ANNs are received by the scientific community. Is this enthusiasm well founded? Since the field is not well known among physicists and chemists, in this paper we are going to set neural networks applications in the perspective of a better established mathematical theories.

In the next section we critically analyze a few recent applications of ANNs to the problems in chemical physics in which various parameters are correlated with output data. We are going to summarize the idea behind such applications and in the third section elucidate what ANNs are really doing. In the fourth section we describe some alternative approaches and give an illustrative example of learning a simple functional dependence by a backpropagation ANN. In the last section we present a general discussion on the use and misuse of neural networks in physics and chemistry.

II. ASSOCIATIONS USING NEURAL NETWORKS

The typical architecture of a neural network is presented in Fig. 1. The input signals x_i are received by the first layer of processing elements (called "neurons"), the input layer. The results are obtained from the output layer, in Fig. 1 consisting of only one neuron. Between the input and the output layers there are a number of "hidden" layers; in case of Fig.1 only one such layer is present. These hidden layers of neurons are not directly accessible to the user who gives the inputs and sees the outputs from the network. Connections are allowed only between the layers, not within the layers. The input signals are propagated in one direction, from the input layer to the output layer, hence such an architecture is called "a feedforward network", in contrast to the "recurrent networks" with feedback connections between the layers and within the layers. If all possible connections between the layers are allowed the network is called "fully connected". For a large number of neurons, to avoid an excessive number of connections, partial connectivity is assumed (if each neuron in the brain was connected with all others the brain would have to be about 10 km in diameter).

The strength of the connections between the neuron number i and number j is a variable parameter W_{ij}, corresponding to the strength of the synaptic connections in real nervous tissue, called "the weight" of the connection. Adjustment of these weights allows the network to perform a variety of mappings of input to output signals. Each neuron performs a weighted sum of the incoming signals

$$X_i = \sum_j W_{ij} \sigma_j \tag{1}$$

and processes the result via a function Φ. Because of the biological motivations most of the feedforward networks assume for an output function of a neuron a sigmoidal function

$$\Phi(X) = \sigma(X) = \frac{1}{1 + e^{-(X-\theta)/T}} \tag{2}$$

where T is a global parameter, usually fixed for all processing elements (neurons) of the network, determining the degree of the non-linearity of neurons, and θ is called the threshold and is usually also fixed for all processing elements. Thus the number of parameters adjusted during the adaptation of a network to a given set of data is equal to the number of network weights.

This short introduction should be sufficient to understand the applications described below. One of the pioneers in the field of neural modeling, T. Kohonen, wrote in his 1984 book on associative memory [5]:

"... arithmetic problems are seldom solved in biological tasks... Any attempts to build neural models for the adder, subtractor, multiplier, and other computing circuits, or even analog–to–digital converters are therefore based on immature reasoning."

Unfortunately many applications in physics and chemistry are of this type. Here we shall write only about applications that use neural networks in the "most proper" way, to form associations between the input and the output values. The papers of Darsy *et.al* [6] and Androsiuk *et.al* [7] are rather typical in this regard. The neural network is taught solutions of the Schrödinger equation i.e. correlation between the parameters of some Hamiltonian and the energy. In case of the two papers quoted above a two-dimensional harmonic oscillator Hamiltonian was used. The potential is quadratic and the lowest eigenvalue of this Hamiltonian is linear in both frequency parameters:

$$V(x, y; \omega_x, \omega_y) = \frac{1}{2} m(\omega_x^2 x^2 + \omega_y^2 y^2)$$
$$E(\omega_x, \omega_y) = A(\omega_x + \omega_y) \tag{3}$$
$$\tag{4}$$

where $A = \hbar/2$ is constant. A set of training data for different (ω_x, ω_y), consisting of values of $V(x, y; \omega_x, \omega_y)$ at a rectangular (x, y) mesh taken as inputs and the corresponding energy values E taken as outputs are given to the network. The backpropagation algorithm is used to change the weights of the ANN bringing the responses of the net, identified with the E values, for the given input $V(x_i, y_i, \omega_x, \omega_y)$, as close as possible to the desired E.

Perfect agreement is usually not possible or even not desirable from the point of view of "a generalization" or prediction of the unknown E values corresponding to the new (ω_x, ω_y) parameters. When fitting the data one rarely requires that the approximation function should pass exactly through the given data points, because this may lead to the "overfitting" or oscillatory behavior. The accuracy and the speed of learning depends on the number of hidden neurons and the architecture of the ANN. After the training phase is finished a number of new parameters (ω_x, ω_y) are selected and values of the $V(x, y; \omega_x, \omega_y)$ taken as a test data to check the ability of the trained network to guess the new values of E. The accuracy in learning and testing phases does not exceed a few percent.

In the paper of Darsy *et.al* a fully connected network with 40 hidden neurons was used, 49 values of $V(x, y)$ on 7×7 mesh were given, the number of data sets for training was 50 and the maximum error for the training data was around 5%. Slightly better results were obtained by Androsiuk *et.al* [7] with a backpropagation network that was not fully connected: only 3 hidden neurons, 1 output and 49 input neurons (one for each point on potential surface) were used, the number of the data sets for training was also 50, and the maximum error for the training data was about 2.5%. The extrapolation of the results to other values of (ω_x, ω_y) gives errors gradually increasing as the parameter values move outside the test range.

Interpolation and extrapolation is discussed in these papers in terms of "generalization of acquired knowledge". One may be easily misled by the use of such concepts. For example, the authors of the papers quoted above conclude [6] that: "neural networks can be used to investigate the more perplexing questions related to basic issues of physics and chemistry" and [7] claim: "... we presented studies of a neural network capable of performing the transformations generated by the Schrödinger equation required to find eigenenergies of a two-dimensional harmonic oscillator". In fact in both papers the authors have trained a network to recognize points on a plane in 3 dimensions (ω_x, ω_y, E), and tested whether the network can interpolate the data. Certainly such ability has nothing to do with "the basic issues of physics and chemistry" or with "transformations generated by the Schrödinger equation", but rather with the data interpolation and extrapolation techniques.

A conceptually very similar, although computationally more ambitious, example of using backpropagation ANNs for correlation of data is found in a series of papers of Sumpter *et.al* [8]. Internal energy flow in molecular systems was studied using the data from molecular dynamics calculations. The ANN was taught the relationship between phase-space points along a classical trajectory and energies for stretch, bend and torsion vibrations. The network used after some experimentation had 84 nodes in 4 layers, with a total of 1648 connections. The input vectors were 24-dimensional (coordinates and momenta for 4 atoms) and the output was 4-dimensional (energies of 4 modes). The accuracy of energy prediction after training on 2000 examples of data points was between 5-20%. The authors conclude that "a trained neural network is able to carry out qualitative mode-energy calculation for a variety of tetratomic systems".

Many other examples of this sort may be found in the literature. Although superficially they are similar to the papers quoted below there is a crucial difference that we will point out in the summary.

Peterson has used a BP network for classification of atomic levels in Cm I, Cm II and Pu I ions [9] according to their electronic configuration. Each level was described by 4 numbers: energy, angular momentum, g factor and isotope shift, that should be correlated with a small number (4 to 8) of electronic configurations. The network used had less than 100 neurons and the accuracy of classification was between 64 and 100%.

Many papers have been published on applications of neural networks to various problems in protein chemistry [10]. The ANNs are used here for classification purposes to find the correlation between the 3D structure and the sequence of aminoacids. Since an average protein has a few hundred aminoacids ANNs used in this case have tens of thousands processing elements and hundreds of thousands of weights and their simulation requires a large amount of supercomputer resources.

The time to train a fully connected feedforward network is very long (thousands or hundreds of thousands iterations may be necessary) and selection of the architecture for networks that are not fully connected is a long trial-and-error procedure. However, once the feedforward network has been trained it gives the answers very quickly, in one iteration, since computation is reduced to optimized function evaluation.

III. WHAT ARTIFICIAL NEURAL NETWORKS REALLY DO.

Comments in this section are relevant mainly for the most commonly used feedforward ANNs of the backpropagation type. Applications of ANNs to various problems in physics and chemistry are frequently not based on solid mathematical foundations, but rather on the availability of the software to simulate neural networks. In the applications mentioned in the previous section the ability of neural networks for learning from examples, associating a set of parameters with the output values (cf. [6]- [10]) and subsequently generalizing to new values, was used. Accuracy of interpolation did not exceed a few percent and the number of adjustable parameters (weights of the network) was in most cases quite high, ranging from 10 to almost 10^6.

The quotations from the papers reviewed in the previous section may mislead some readers into believing that neural networks really solve physical problems or carry out transformations of the Schrödinger equation, i.e. do something intelligent. Therefore we should stress that the problem of learning associations between parameters and output values is equivalent to the data fitting, i.e. to the problem of approximation of an unknown input-output mapping. Vice versa, all data fitting problems may be presented in the ANN form. Therefore the question one should ask is: are neural networks efficient in fitting the data and what is the functional form they are using?

The general approximation problem is stated as follows [11]: if $f(X)$ is continuous function defined for $X = (x_1, ..x_n) \in \mathcal{X}$, $F_W(X)$ is continuous approximating function of X and some parameters $W \in \mathcal{W}$, then finding the best approximation amounts to finding $\bar{W}$ such that the distance $||F_{\bar{W}}(X) - f(X)||$ is the smallest for all W in the parameter space $\mathcal{W}$.

Consider the linear expansion $F_W(X)$ in a set of basis functions $\Phi_i(X)$, approximating an unknown function $f(X)$ of many variables X:

$$F_W(X) = \sum_i^m W_i \Phi_i(X) \tag{5}$$

Given a set of sample points $(X_k, f_k = f(X_k))$ our task is to find the best possible expansion coefficients W_i. We may imagine a network realization of the approximation problem in several ways, for example, a single-layer network that solves the problem is shown in Fig. 2, with m hidden nodes ($m = 5$ in this example), a single input and single output node. Each input node is connected with the weight equal to 1 to the hidden node and with the weight equal to W_i to the output node. The input node sends unmodified signals to the hidden nodes that have output functions equal to $\Phi_i(X)$ and the output node performs the summation of all weighted contributions. Any linear or iterative method of data fitting may be used as a learning algorithm to find the best "weights" or expansion coefficients W_i. This scheme covers many approximation methods, including Fourier transforms, spline interpolations and polynomial fits. Generalization for vector functions (many-components) requires many output neurons and realizes a general mapping between two vector spaces.

Most of the feedforward networks assume for an output function of a neuron a sigmoidal function Eq. (2). Since this function has as an argument x_i, a weighted sum of all signals σ_j received by a given neuron i from neurons j that are connected to its inputs

$$x_i = \sum_j W_{ij}\sigma_j \tag{6}$$

the number of parameters adjusted during learning is equal to the number of non-zero connections of the network. In the examples presented in the previous section the ANNs had hundreds of adjustable parameters. The task required from these networks during training was to fit the available (training) data to some functional form by adjusting these parameters. Although one may think that in using ANNs no functional form is *a priori* assumed this is obviously not true. How does an explicit function realized by the backpropagation network look? It is usually presented in the recursive way

$$F_W(X) = \sigma(\sum_{i_1} W^{(1)}_{i_1}\sigma(\sum_{i_2} W^{(2)}_{i_2}\sigma(...(\sum_{i_k} W^{(k)}_{i_k} x_{i_k})...))) \tag{7}$$

where σ is the sigmoidal output function of the node and the upper indices $1..k$ refer to the network layer. For a network with 2 active layers (Fig. 1) the explicit output function is not difficult to write:

$$\begin{aligned} F_W(X) &= \sigma(\sum_i W^{(1)}_i \sigma(\sum_j W^{(2)}_{ij} x_j)) \qquad (8)\\ &= \sigma\left(\sum_i W^{(1)}_i \frac{1}{1+\exp(-\sum_j W^{(2)}_{ij} x_j/T)}\right) \\ &= \frac{1}{1+\exp(-\sum_i W^{(1)}_i \frac{1}{1+\exp(-\sum_j W^{(2)}_{ij} x_j/T)}/T)} \end{aligned}$$

The formulas for the backpropagation of errors training algorithm [3] may easily be obtained by computing the gradient of this function with respect to the weights (approximation errors are proportional to this gradient). If the signals x_i are small and T is large we can expand the exponential function and the geometrical series leaving the linear approximation:

$$F_W(x_1, .., x_n) \approx \sum_i W_i \sigma(x_i) \tag{9}$$

which has the same form as Eq. 5 with $\sigma(x_i)$ as the basis functions for the expansion. However, in the usual case the dependence on the parameters W is non-linear.

In general sigmoidal functions can approximate any continuous multivariate function [12], although high quality of this approximation requires in most cases a rather large number of parameters W. Conditions for convergence of such expansions have been found only recently [13]. Neural networks may therefore serve as the universal approximators. Many other neuron output functions lead to networks that may serve as the universal approximators. Radial basis functions [14], and even more general kernel basis functions [15] can be used for uniform approximation by neural networks. In fact many other types of function may be used, for example rational functions [16] for some approximation problems lead to networks of lower complexity (smaller number of parameters, i.e. faster convergence) than the networks based on sigmoidal or radial functions.

For many problems linear approximation is the most appropriate method and an attempt to use neural networks or any other nonlinear approximators will lead to low accuracy and lengthy computations. Periodic functions (covering the prediction of time series in physics, chemistry, economics and other fields) are much better represented via Fourier, wavelet [17] or similar expansions rather than by sigmoidal functions. Many other functions useful in physics are of gaussian type. Why should we hope that the ANN based on the sigmoidal functions will give us better results than the fitting procedures?

The training algorithm modifies the weights (or logical functions of the nodes in case of logical networks [18]), slowly changing the landscape of the $F_W(X)$ function realized by the net until it will approximately give in the N sample points the same values as the $f(X)$ function. Since the initial form of the mapping depends on the random set of weights W at the start of the training period, the final form of the function $F_{\bar{W}}$ after training is to a large degree arbitrary, except for the close neighborhood of the N training points. Hoping that the ANN will find an approximation far from the sample points $(X_i, f(X_i))$ is unreasonable. At most the accuracy that one may obtain is equal to that of a fit with W_{ij} parameters using sigmoidal functions.

Fitting a set of data points to a linear combination of basis functions by the least-squares procedure, in case when the function is smooth, is not difficult. Formulated as a neural network learning problem with just one output neuron such an approach is known as the functional link neural network [19]. Nonlinear fitting problems are difficult and neural network approach is not an exception, in fact even quite simple networks lead to NP-complete problems [20]. A very interesting solution to this problem is based on the idea of growing and shrinking networks, allocating resources to new data and constructing the approximating function incrementally [21].

The problem of fitting non-smooth functions in many dimensions, as in the clustering analysis, is also difficult. Fitting in a multidimensional space is much harder than in one or two dimensions and it remains to be seen if neural networks can compete with least squares fits to expansions in some basis set. Comparison of various nonlinear approaches to classification for real world data shows that the accuracy of neural networks is similar to nonlinear regression and tree-induction statistical methods [22].

IV. EXAMPLE OF ANN BEHAVIOR

From the point of view of the approximation theory physical and chemical problems treated in the literature by ANNs are in some cases trivial. Consider for example the results of [6] and [7]: we do not need 50 sample data points to conclude that the dependence of E on (ω_x, ω_y) is linear. If there is no way of guessing the functional dependence or if there is no global fit that will give small errors we may try to use various spline functions to get the correct local behavior and glue them together to get a global description of the data. However, functional dependencies for most physical problems are known (in contrast to problems in pattern recognition to which ANNs are applied with some success, for example in high-energy physics [23]). They result from the underlying simplified physical models; parametrization of functions and data fitting is a well-established and highly accurate method for creating mathematical models.

Perhaps an explicit example will clarify the need for rigorous methods in the reconstruction of functional dependencies from sample data. In Fig. 3 we have presented a function $f(x) = \sin(\sqrt{2}x)\sin(2x)$ and various fits and approximations based on the 11 sample points taken every $2\pi/10$ in the $(0, 2\pi)$ region. The original function $f(x)$ is left for comparison on all drawings (thin line). Polynomial fits of at least 8–th order have to be used to give a sufficient number of minima and maxima and such polynomials lead to large oscillations between the sample points. Fig. 3a) shows the original function and its approximation by a Fourier series based on an 11-parameter expansion in the functions $1, \sin(x), \cos(x), ... \sin(5x), \cos(5x)$. It is a global fit, good for interpolation in the whole $(0, 2\pi)$ region but completely failing outside of it. In the second drawing we see the fit based on the highly non-linear local sigmoidal functions with T=0.01. It has the following form:

$$F(x) = \sum_{i=1}^{11} W_i \sigma(x - x_i) \qquad (10)$$

Results that one can obtain with this type of fitting function represent the limit that an ANN of any architecture with 11 linearly adjustable parameters and sigmoidal output functions of neurons may achieve (numerical experiments with other, multi-layer networks with non-linear parameters, always gave worse results). Variable thresholds for individual neurons, equivalent to the centering of sigmoidal functions around the data points, are a necessary prerequisite for a good local approximation in this case. For $T = 0.01$ the non-linearity of the sigmoidal functions in the range of $\pm\pi/5$ is rather strong and the overall fit is poor. In Fig. 3b) one can see how the approximating function is combined from steep sigmoids – the shape of the sigmoid functions is also shown in this figure. Networks with high non-linearities are simply not capable of a smooth modeling of the data in this example.

The behavior of the approximating function for parameters outside the training range strongly depends on the value of T. This is illustrated by the next two drawings, Fig. 3c) and d), obtained with local sigmoidal fits for $T = 0.5$ and $T = 1$. The accuracy of these fits is even better than that of the Fourier series fit! In this case the sigmoidal functions, shown in these figures, could be replaced by the semi-linear functions, and the approximating function by the semi-linear spline function. In our experience it is hard to find a function $f(x)$ for which this would not be the case.

The last drawing shows the results obtained by a back-propagation network with 11 non-zero weights trained on the 11 input points and tested on 100 points in the $(-\pi, 3\pi)$ range. The curve gives an overall idea of the quality of the ANNs approximation – the results can vary, depending on the randomly set starting weights and the parameter T. Fig. 3e) shows the best results we could find after many experiments with different network architectures, hundreds of thousands of iterations and essentially forcing the network to learn some points (giving the values of these points more frequently than the others). The network learned the values of the 11 training points to a very high precision (0.001).

Although the trained network is useful for interpolation of the data the extrapolation properties of this network are completely unreliable. A simple way to improve the extrapolation is to use auto-regression, i.e. instead of using pairs of points (x_i, y_i) for fitting or network training one may use a set of $y_i, y_{i-1}, ...$ points. The use of autoregression does not change the overall conclusion concering the fitting procedures and the network behavior.

The strength of ANNs for some problems where non-linear relations make it hard to find a global approximation lies in the local description of the approximated function around the training data vectors. The accuracy of such a description is rather low. Looking at the function realized by the untrained net, starting with random weights, with the net outputting one real number for n-dimensional input vector, we get a hypersurface in $n+1$ dimensions, rather smooth but irregular. Learning input-output associations changes this hypersurface to reproduce the output values around the training input values, but most of the random structure of the initial hypersurface, including local minima and regions far from the input data, remain unchanged. It is instructive to look at this function during training, and to notice how the model of the data that the network has is changing.

V. ALTERNATIVES TO ANNS

Approximation theory leads to regularization of functions, a very important concept [24] especially for applications with noisy data (as usually obtained from experiments). In essence regularization takes into account additional information in form of constraints that should be fulfilled by the approximating function, defining a functional:

$$H[f] = \sum_{i=1}^{N} (y_i - f(X_i))^2 + \lambda ||\hat{P}f||^2 \tag{11}$$

where (X_i, y_i) are know data points, $\hat{P}$ is the constraint operator and λ is a real parameter determining how important the constraints should be. This functional is minimized over all functions $f(X)$ belonging to some class of trial functions. The approximation in the least-square sense follows for $\lambda = 0$ (no regularization). Elegant solutions are know for many constraint operators, allowing the avoidance of "data overfitting" effects in case of noisy data inputs. For example, to smooth the approximating function by minimizing the rapid variation of its curvature $\hat{P}$ should include second derivatives:

$$||\hat{P}f||^2 = \sum_{i=1}^{N} \left\{ \sum_{j=1}^{n} \left(\frac{\partial^2 f(X_i)}{\partial x_j^2} \right)^2 \right\} \tag{12}$$

Regularization may also be applied to the backpropagation ANNs [25] and can be implemented by ANNs with one hidden layer [26]. Poggio and Girosi [26] show how regularization theory may be extended to what they call "theory of Hyper Basis Functions", containing the Radial Basis Function (RBF) method [14] as a special case. The RBF approach allows for multivariate interpolation in a way that is better for most chemistry and physics problems than the ANN interpolation with sigmoidal functions. The RBF method assumes the following functional form to the approximate function $f(X)$ given the value at N points $X_i = (x_1^{(i)}, x_2^{(i)}, ..x_n^{(i)})$:

$$F(X) = \sum_{i=1}^{N} C_i h(||X - X_i||) + \sum_{i=1}^{m} D_i p_i(X) \tag{13}$$

where h is a continuous function centered at X_i and p_i is a polynomial of some low order (in particular a constant). Some of the h functions used with very good results for problems in physics [27] include linear and Gaussian functions:

$$\begin{aligned} r = ||X - X_i||;\ h(r) &= r \qquad (14) \\ h(r) &= \frac{1}{(c^2+r^2)^\alpha}, \quad \alpha > 0 \\ h(r) &= (c^2+r^2)^\beta, \quad 1 > \beta > 0 \\ h(r) &= e^{(-r/c)^2} \end{aligned}$$

Another branch of mathematics in which the approximation of multidimensional functions may be based is the statistical decision theory (cf. [5] and references therein), probabilistic Bayesian classifiers [28] and regression theory [29]. These approaches should be preferred for classification problems [9], [10] and if the data clustering is rather strong. The best known programs based on this theory are variants of the Learning Vector Quantization (LVQ) algorithm of Kohonen [30]. Other relevant methods were developed by the high energy and plasma physics communities and are known as function parametrization methods [31].

A recent comparison of various non-linear approaches to classification and data modeling, such as neural networks, statistical pattern recognition, MARS and BRUTO [22] shows that all these methods have their weak and strong points, depending on particular applications. Fuzzy sets theory [32] and local coordinate transformation methods based on differential geometry are also strong competitors of ANN algorithms [33] offering a well-defined mathematical background for local data approximation. ANNs are but one family of systems among many types of adaptive systems reconstructing hypersurfaces from the sample data by adjusting internal parameters.

VI. SUMMARY

From the preceding sections it is clear that there are many alternatives to the use of neural networks for complex approximation problems. There are obvious cases when the use of neural networks is quite inappropriate: whenever linear methods are sufficient or whenever least-squares fits in some basis functions works well. Given a sufficient number of network parameters (weights and processing functions) and a sufficient number of data points an approximation to an arbitrary mapping may be obtained [12]. For some problems approximation via sigmoidal functions, especially with strong non-linearity, is slowly converging – a reflection of the fact that no physical insight is used in the construction of the approximating mapping of parameters on the results. The number of adjustable parameters (weights) in an ANN is usually quite large. Time for training the ANN, tedious selection of network architecture, neuron output function and global learning parameters plus the dependence of results on the initial state of the network make the use of neural networks for solving physical problems a very unreliable method.

Recently Bishop [25] proposed a fast curve fitting procedure based on neural networks. A trained ANN is used to give quickly an approximation to the non-linear parameters of the iterative fitting procedure. It has already been suggested in the context of Adaptive Logical Networks [18] that after training the ALN net the function that it has learned should be extracted: isn't it better to construct such approximating functions directly? Alternative approaches, based on the approximation theory and theory of statistical decisions have more rigorous mathematical foundation and properly applied should lead to better results with a smaller number of parameters.

Consider for example theoretical molecular physics and quantum chemistry. We are trying to associate, using some complicated computational machinery, certain parameters, such as geometric or one-electron basis set information, with the values of energy and other properties. Does a global function of these parameters giving energy $E(p_1,..p_n)$ and other properties exist? Can we create an approximating function containing a large number of adjustable parameters from computational results plus the empirical data that will allow us to guess new values? If good empirical or *ab initio* data is available for construction of such a mapping it is to some degree possible and this approach has been used for many years to obtain reliable molecular (and more recently also solid state) potential energy surfaces [34]. The results of such mapping based on physical models for underlying expansions are much more reliable than the results one may obtain using ANNs [35].

Will the neural networks have significant impact on the methods of solving the physics and chemistry problems? Our conclusion is that using *feedforward neural networks* to solve some of these problems is a rather inefficient way of fitting the data to a specific functional form. However, if there is no approximate theory but the data is not completely chaotic – as for example in the case of proteins [10], QSAR and other problems in physics and chemistry [38] or time series forecasting in economics or physics [36], [37] – any data modeling tools are worth using, including neural networks. There is no need to insist on sigmoidal processing functions, usually more accurate results are obtained with a smaller number of parameters using approximating functions based on gaussian or other localized functions [14]. Moreover, various methods such as resource allocating neural networks and other constructive algorithms [21] automatically adding more network nodes to describe the data with higher accuracy may be more convenient for data modeling. Another case when neural network methods may have strong advantages is when a large amount of data coming from experiment or computations should be processed. Neural networks learning algorithms lead to small changes of the network parameters for each input data item presented, while global fitting methods require access to all data.

Approximation theory and statistical decision theory, especially if approximate functional dependencies are known, give a firm mathematical background to the treatment of many problems to which neural network techniques are applied in an *ad hoc* manner. Instead of backpropagation neural networks, applications based on explicit, well-controlled construction of approximate mappings should be preferred. Papers on applications of ANNs to problems of physics and chemistry should at least compare the results with the results obtained using statistical methods and data fitting procedures.

ACKNOWLEDGMENTS

W.D. gratefully acknowledges grant of the Research Council of the Polish Ministry of National Education and support of the Max-Planck Institute during his visits in Garching. We also thank prof. M. Zerner and prof. B. Wybourne for reading and correcting the manuscript and dr W. Kraemer for pointing out some references.

[1] For a short description of available hardware and software packages please see the Frequently Asked Questions on the Usenet comp.ai.neural-nets discussion group (available by anonymous FTP as file "neural-nets-faq" from many Internet nodes).

[2] J.A. Anderson, and E. Rosenfeld, (Eds). Neurocomputing: Foundations of Research. (The MIT Press: Cambridge, MA.(1988)); J.A. Anderson, A. Pellionisz, and E. Rosenfeld, (Eds). Neurocomputing 2: Directions for Research. (The MIT Press: Cambridge, MA.(1990)); R. Hecht-Nielsen, Neurocomputing. (Addison Wesley, 1990); J. Hertz, A. Krogh and R. Palmer, Introduction to the Theory of Neural Computation. (Addison-Wesley: Redwood City, California 1991); J.L. McClelland and D.E Rumelhart, Explorations in Parallel Distributed Processing: Computational Models of Cognition and Perception (software manual). (The MIT Press, Cambridge, MA 1988); P.D. Wasserman, Advanced Methods in Neural Computing. (Van Nostrand Reinhold: New York, 1993); T. Kohonen, An Introduction to Neural Computing. Neural Networks 1 (1988) 3-16.

[3] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, Learning representations by backpropagating errors. Nature, 323 (1986) 533–536; D.E. Rumelhart and J.L. McClelland, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, (The MIT Press, Cambridge, MA 1986), Vol.1, pp 318–362; P.J. Werbos, Proc. of IEEE 78 (1990) 1550

[4] T.L.H. Watkin, A. Rau and M. Biehl, Rev.Mod.Phys.65 (1993) 499

[5] T. Kohonen, Self-organization and Associative Memory. (Springer-Verlag, New York, 1984, 2nd Edition: 1988; 3rd edition: 1989).

[6] J.A. Darsey, D.W. Noid and B.R. Upadhyaya, Chem.Phys.Lett.177 (1991) 189; 181 (1991) 386

[7] J. Androsiuk, L. Kułak and K. Sienicki, Chem.Phys.173 (1993) 377

[8] Sumpter B.G., C. Getino, D.W. Noid, J.Chem.Phys.97 (1992) 293; J.Phys.Chem.96 (1992) 2761; B.G. Sumpter and D.W. Noid, Chem.Phys.Lett.192 (192) 455

[9] K.L. Peterson, Phys.Rev A 41 (1990) 2457; 44 (1991) 126

[10] Poliac M.O., G.L. Wilcox, Y. Xin, T. Carmeli, M. Liebman, IEEE International Joint Conference on Neural Networks (New York, NY, IEEE 1991); Wilcox G.L., M. Poliac, M. Liebman, Proceedings of the INNC 90, Paris (Kluwer, Dordrecht, Netherlands 1990), p. 365; E.A. Ferran, P. Ferrara, Physica A 185 (1992) 395; M. Blazek, P. Pancoska, Neurocomputing 3 (1991) 247; M. Compiani, P. Fariselli, R. Casadio, in: Proceedings of the Fourth Italian Workshop on Parallel Architectures and Neural Networks, Vietri sur Mare, Italy (World Scientific, Singapore 1991), p. 227

[11] J.R. Rice *The Approximation of Functions* (Addison Wesley, Reading, MA 1964); G.G. Lorentz, *Approximation of Functions* (Chelsea Publishing Co, New York, 1986).

[12] G. Cybenko, Math. Control Systems Signals 2 (1989) 303; K. Funahashi, Neural Networks 2 (1989) 183; K. Hornik, M. Stinchcombe, H. White, Neural Networks 2 (1989) 359; Jones L.K, Proc. of the IEEE 78 (1990) 1586

[13] H. White, Neural Networks 3 (1990) 535; A.R. Barron, IEEE Transactions on Information Theory 39 (1993) 930

[14] M.J.D. Powell, "Radial basis functions for mulitvariable interpolation: a review", in: J.C. Mason and M.G. Cox, eds, *Algorithms for Approximation* (Clarendon Press, Oxford 1987)

[15] V. Kůrková, K. Hlaváčková, Proceedings of the Neuronet'93, Prague (1993)

[16] H. Leung, S. Haykin, Neural Computation 5 (1993) 928

[17] C.K. Chui, Ed: *Wavelet Analysis and Its Applications* (Academic Press, Boston, 1992)

[18] W.W. Armstrong, A. Dwelly, R. Manderscheid and T. Monroe, An Implementation of Adaptive Logic Networks, Univ. of Alberta, Comp. Science Dept, November 11, 1990 technical report.

[19] Y-H. Pao, *Adaptive Pattern recognition and neural networks.* (Addison-Wesley, Readnig, MA 1989)

[20] A. Blum, R. Rivest, Training a 3-node neural network is NP-complete. In: COLT '88, Proc. of 1988 Workshop on computational learning theory. MIT 1988 ((M. Kaufmann, San Mateo, CA); S. Judd, in: Proc. IEEE intern. conference on neural networks, San Diego, CA, 1987, vol II, p. 685

[21] S.I. Gallant, Neural network learning and expert systems (Bradfor Book, MIT Press 1993); J. Platt, Neural Comput. 3 (1991) 213; V. Kadirkamanathan, M. Niranjan, Neural Comput. 5 (1993) 954; B. Fritzke, Vector quantization with growing and splitting elastic net, in: ICANN '93: Porceedings of the International Conference on Artificial Neural Networks, Amsterdam 1993; S.E. Fahlman, C. Lebiere, Tech. Rep. CMU-CS-90-100, Carnegi-Mellon School of Comp. Sci. (1990)

[22] B.D. Ripley, Neural networks and flexible regression and discrimination. In: *Statistics and Images*, ed. K.V. Mardia (Abingdon, Carfax, 1993); B.D. Ripley, Flexible non-linear approaches to classification. In: *From Statistics to neural networks. Theory and Pattern Recognition Applications*, eds. V. Cherkassky, J.H. Friedman, W. Wechsler (Springer Verlag 1994); B.D. Ripley, J. Roy. Stat. Soc. B 56 (1994) (in press)

[23] P. Treleaven and M. Vellasco, Comp.Phys.Comm.57 (1989) 543; B. Humpert, Comp.Phys.Comm.58 (1990) 223

[24] A.N. Tikhonov, Soviet Math. Dokl 4 (1963) 1035; A.N. Tikhonov, V.Y. Arsenin, Solutions of Ill-Posed Problems (W.H. Winston, Washington D.C. 1977); V.A. Morozow, Methods for Solving Incorrectly Solved Problems (Springer Verlag, Berlin, 1984)

[25] C.M. Bishop, in: Proceedings of the INNC, Paris, Vol. 2 p. 749 (Kluwer, Dordrecht, Netherlands 1990); AEA Technology Report AEA FUS 162

[26] T. Poggio and F. Girosi, Proc. of the IEEE 78 (1990) 1481

[27] E.J. Kansa, Computers Math. Appl. 19 (1990) 127; R.Franke, Math. Comp. 38 (1982) 181

[28] R.O. Duda, P.E. Hart, *Pattern classification and scene analysis* (Wiley, New York, 1973)

[29] D.F. Specht, Proc. of IEEE Intern. Conference on Neural Networks 1 (1988) 525; D.F. Specht, IEEE Transactions on Neural Networks 2 (1991) 568

[30] T. Kohonen, Proceedings of the IEEE 78 (1990) 1464; A. Cherubini and R. Odorico, Comp.Phys.Comm.72 (1992) 249

[31] B. Ph. van Milligen and N. J. Lopes Cardozo, Comp.Phys.Comm.66 (1991) 243

[32] G.J. Klir and T.A. Folger, *Fuzzy Sets, Uncertainity and Information* (Prentice Hall, NJ 1988)

[33] Duch W (1992) Department of Computer Methods, Technical Reports 1-3/1992

[34] J.N. Murrell, S. Carter, S.C. Farantos, P. Huxley and A.S.C. Varandas, *Molecular Potential Energy Functions* (Wiley, NY 1984); B.R. Eggen, R.L. Johnston, S. Li and J.N. Murrell, Molecular Physics 76 (1992) 619

[35] A. Lee, S.K. Rogers, G.L. Tarr, M. Kabrisky and D. Norman, Proc. SPIE - Int. Soc. Opt. Eng. 1294 (1990) 138;

[36] R. R. Trippi, E. Turban E., Neural Networks in Finance and Investing, (Chicago, Ill, Probus 1992); Koons H.C., D.J. Gorney, J. Geophys. Res 96 (1991) 5549;

[37] J. D. Farmer, J. J. Sidorowich, Phys. Rev. Letters 59 (1987) 845-848; M. Casdagli, Physical Review A, 35 (1989) 335-356; M. Casdagli, Physica D, 35 (1989) 335; J.B. Elsner, J. Phys. A 25 (1992) 843-50

[38] J. Zupan, J. Gasteiger, *Neural networks for chemists: an introduction.* (VCH, Weinheim, Germany 1993); G.M. Maggiora, D.W. Elrod, R.G. Trenary, J. Chem. Information and Comp. Sciences 32 (1992) 732

FIG. 1. The typical architecture of a feedforward neural network. An input layer I, and output layer with a single neuron, and a hidden layer σ, with two sets of weights, between the input and hidden layer and between the hidden and output layer. This network performs the function given explicitly in Eq. 8 in the text.

FIG. 2. A neural network architecture for fitting the data to a set of basis functions Φ.

FIG. 3. An example of effectiveness of various interpolation and extrapolation techniques for $f(x) = \sin(\sqrt{2}x)\sin(2x)$ function, shown in a thin line on all drawings, with 11 uniformly spaced $(x, f(x))$ data points taken from $(0, 2\pi)$ as the training data. Approximations by: a) Fourier series, 5-th order fit; b) fit using 11 strongly non-linear sigmoidal functions centered on the data points, T=0.01; c) as above, but with much lower non-linearity, T=0.2; d) as above, with T=1.0, equivalent to linear splines; e) the best back-propagation neural network with one input, one output and 10 hidden neurons (20 non-zero weights) we have found.

Fig. 3a

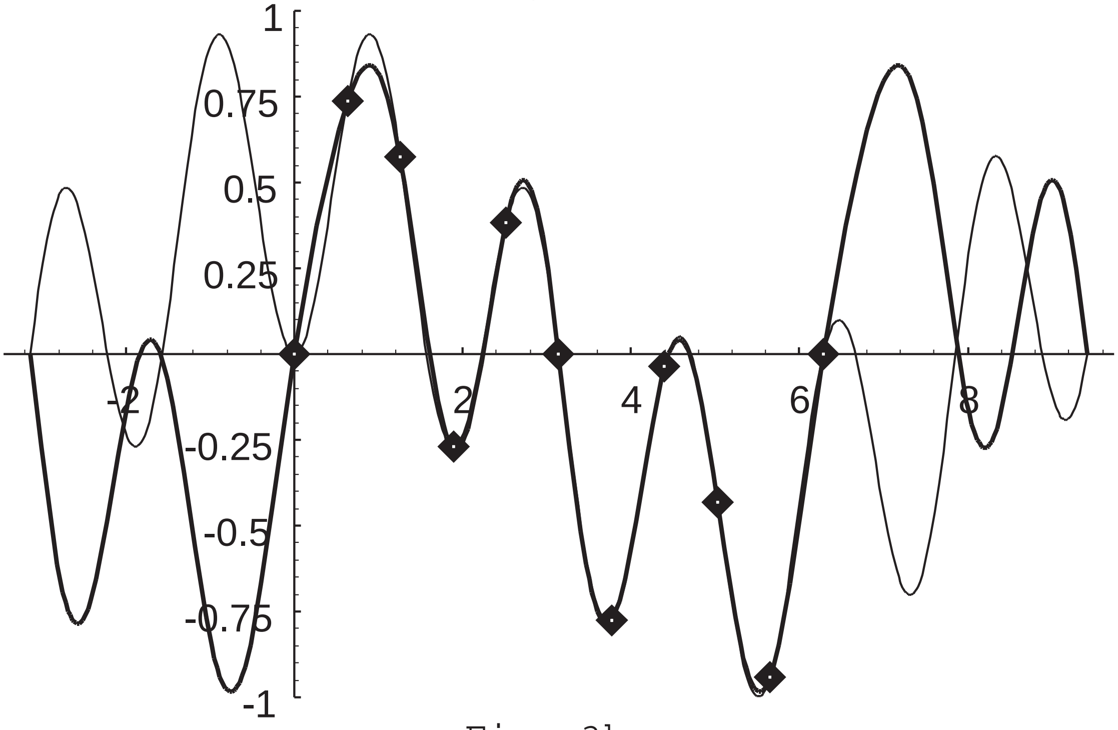
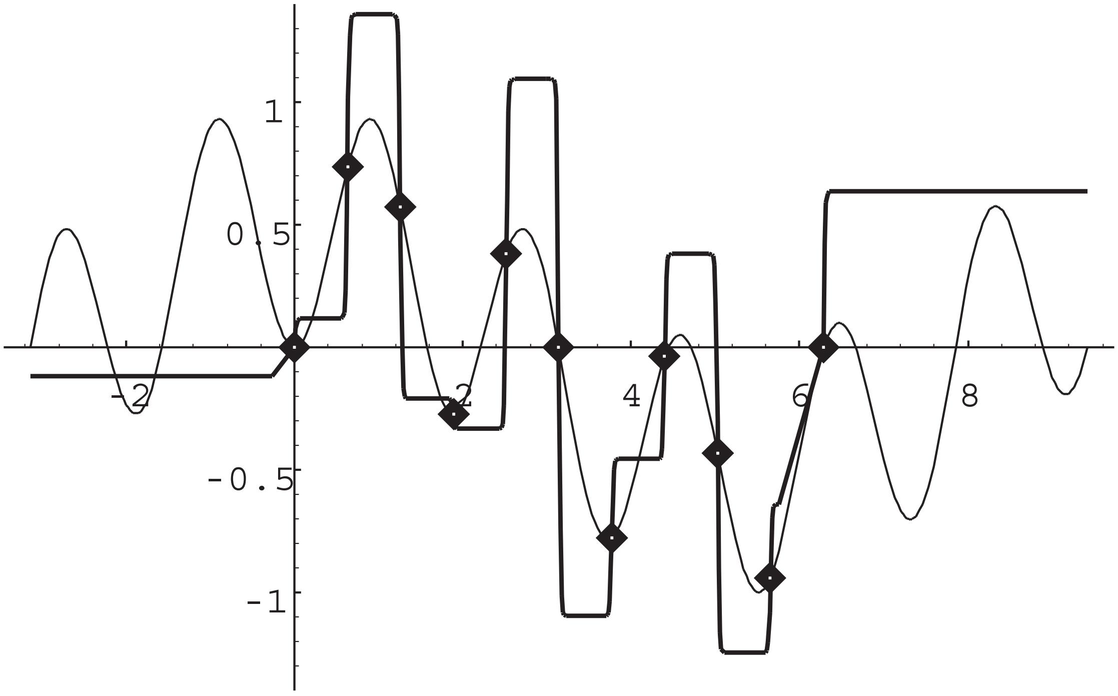


Fig. 3b



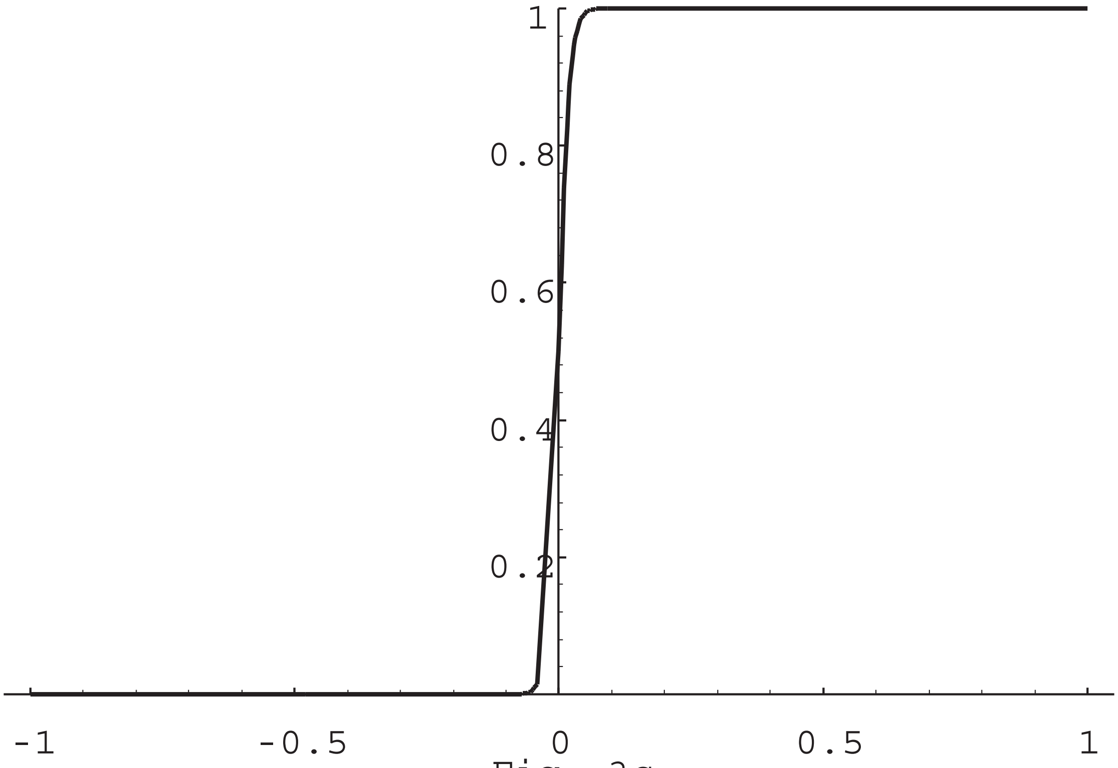
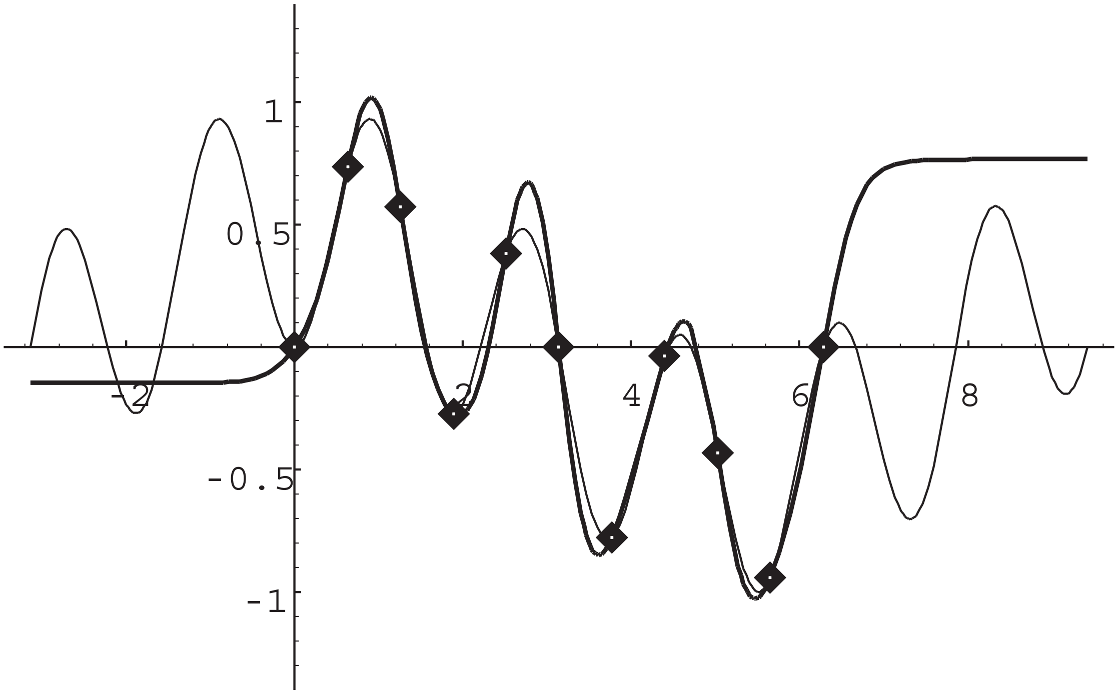


Fig. 3c



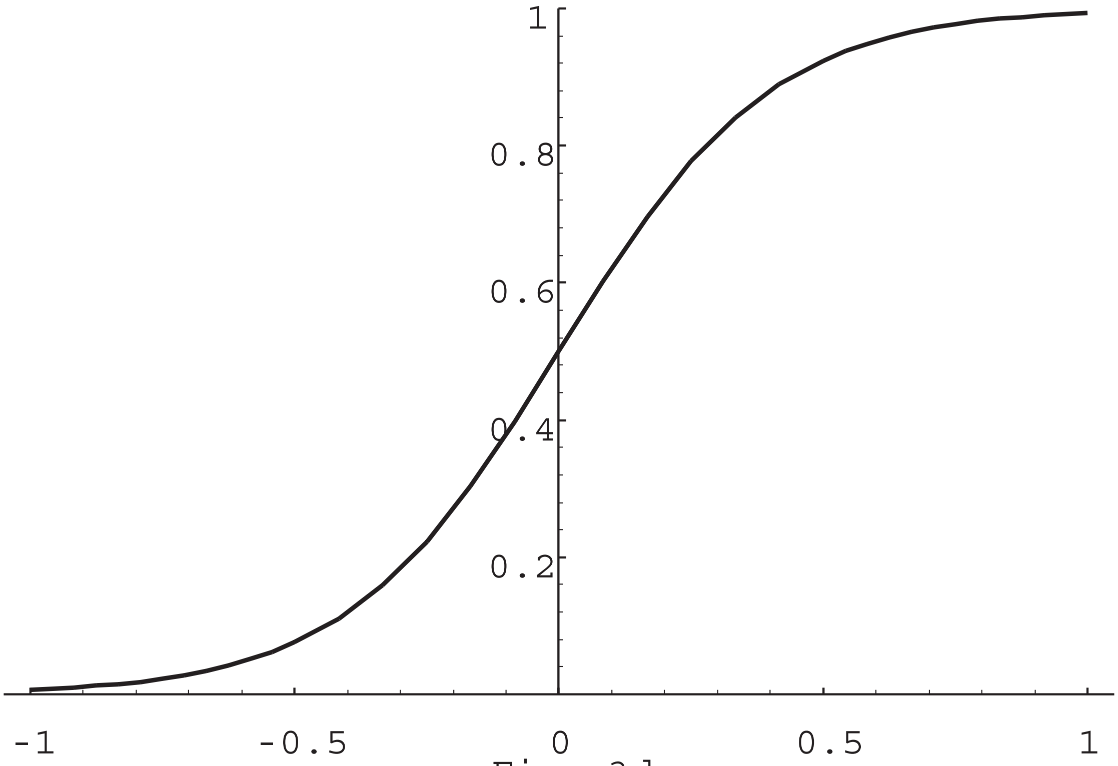
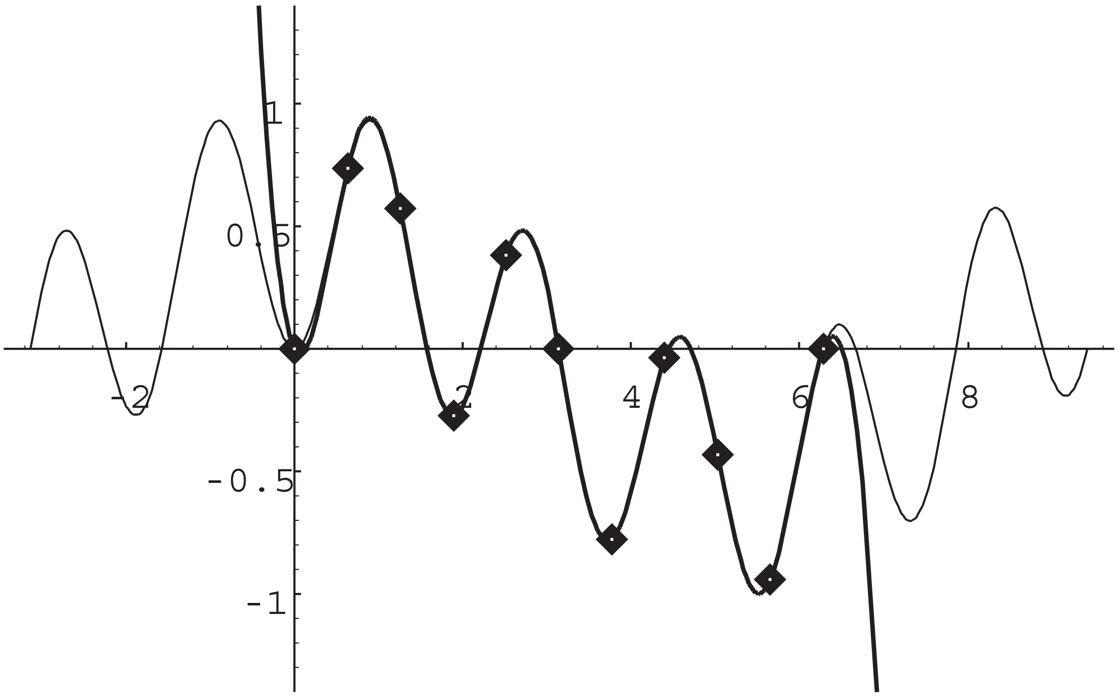


Fig. 3d



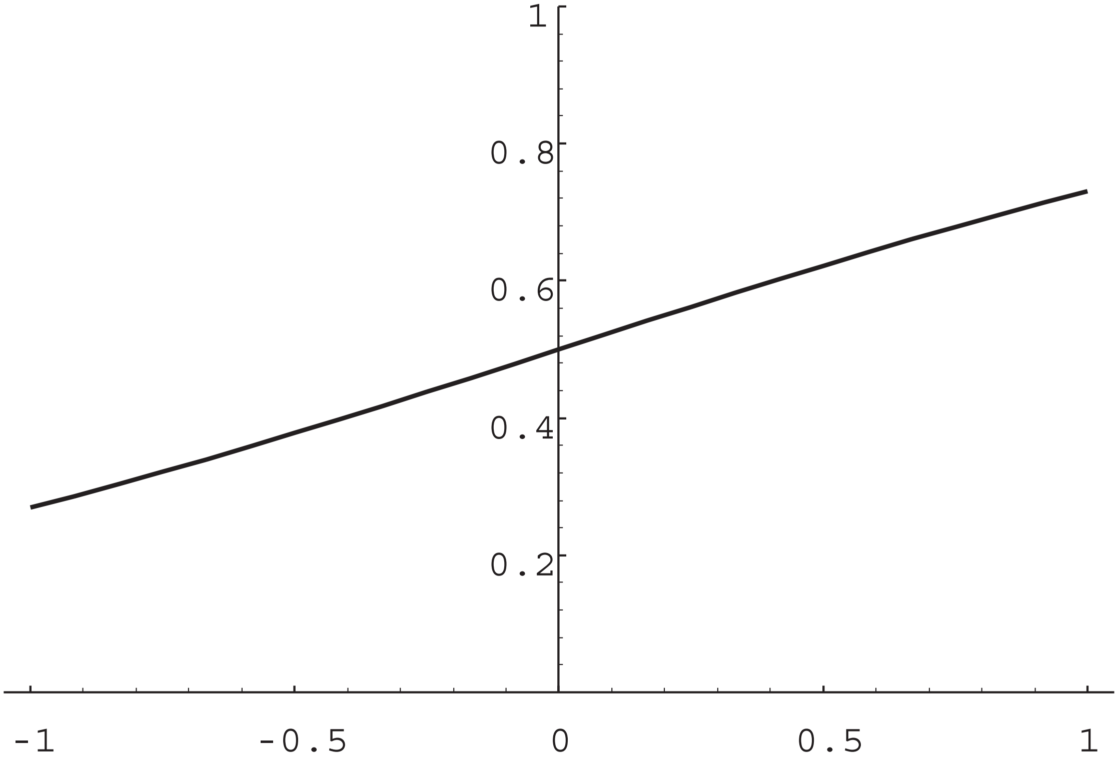
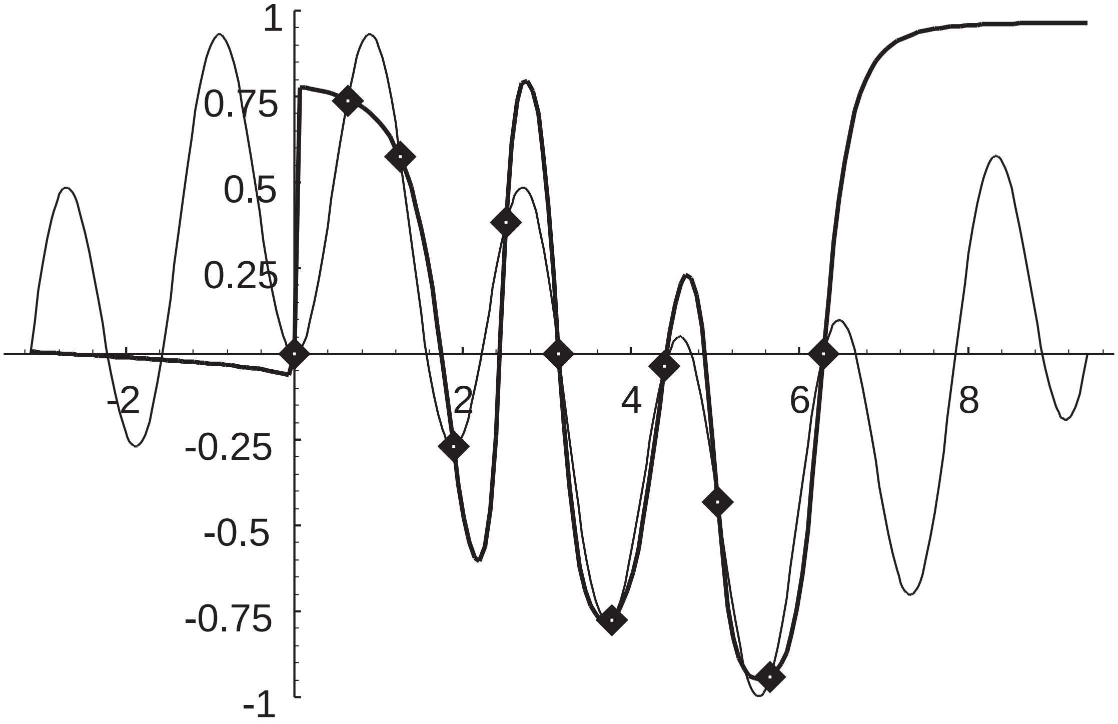


Fig.3e


1
0.75
0.5
0.25
-0.25
-0.5
-0.75
-1
-2
2
4
6
8