JCTC Journal of Chemical Theory and Computation

Neural Network Models of Potential Energy Surfaces: Prototypical Examples

James B. Witkoskie[†] and Douglas J. Doren*

Department of Chemistry and Biochemistry, University of Delaware, Newark, Delaware 19716

Received June 30, 2004

Abstract: Neural networks can be used generate potential energy hypersurfaces by fitting to a data set of energies at discrete geometries, as might be obtained from ab initio calculations. Prior work has shown that this method can generate accurate fits in complex systems of several dimensions. The present paper explores fundamental properties of neural network potential representations in some simple prototypes of one, two, and three dimensions. Optimal fits to the data are achieved by adjusting the network parameters using an extended Kalman filtering algorithm, which is described in detail. The examples provide insight into the relationships between the form of the function being fit, the amount of data needed for an adequate fit, and the optimal network configuration and number of neurons needed. The quality of the network interpolation is substantially improved if gradients as well as the energy are available for fitting. The fitting algorithm is effective in providing an accurate interpolation of the underlying potential function even when random noise is added to the data used in the fit.

I. Introduction

The accuracy of a Monte Carlo or molecular dynamics simulation is limited by the accuracy of the potential energy surface used. Empirical models using explicit analytic potential functions are fast to evaluate but limited in accuracy. On the other hand, first-principles quantum mechanics can provide a very accurate potential energy surface, but the computational expense of these calculations limits the size and length of the simulation. An ideal potential energy surface should have the accuracy of the quantum mechanical calculations and yet be as fast to evaluate as empirical models. One approach to this problem is to construct potential energy surfaces by interpolating among the results of first-principles energy calculations at specific configurations. However, standard interpolation methods (such as splines or orthogonal polynomial methods) are difficult to apply to systems with more than a few degrees of freedom. In recent years, a number of techniques have been developed to address this problem, and substantial progress has been made.[1−17] Nevertheless, more work is needed to develop an accurate, efficient method that can be routinely applied to reacting systems with many degrees of freedom.

One method that has shown promise in a number of applications is the artificial neural network. Several efforts to use neural networks to describe potential functions have now been reported, in most cases with good results.[18−29] These efforts include some rather complex problems, for example: dissociation of H_2 on silicon[18] and palladium[24] surfaces; tetra-atomic van der Waals complexes;[20] water dimer[21] and a polarizable model of liquid water;[27] $H_2O-Al^{3+}-H_2O$ three-body interactions;[23] covalent C−C and C−H interactions in hydrocarbon molecules and pure carbon phases;[26] and a number of problems in vibrational and electronic spectroscopy.[22,25,28,29] The neural network is a very general form that does not require adaptation or special coordinates for each application. As such, it may offer a simple, general approach to potential fits.

A particular strength of the neural network is that it can effectively model data with noise. While nonsystematic errors in quantum chemistry calculations are expected to be small, there may be variation in the energy due to degrees of

* Corresponding author e-mail: doren@UDel.edu.

[†] Current address: Department of Chemistry, MIT, Cambridge MA 02139.

10.1021/ct049976i CCC: $30.25
Published on Web 10/21/2004

freedom that are not included in the model. For example, it is common to model only a relatively small, chemically active region of a larger system. The energy of the smaller subsystem still depends on the coordinates of the surrounding system, but if the smaller system and its surroundings are not strongly correlated, the larger system may be modeled as a thermal bath (as in Langevin models). This approach is likely to be useful for small molecules on metal surfaces. Even for systems where there are strong correlations between the smaller system and its surroundings, the uncorrelated variation in the larger system may be treated as noise. Indeed, this has proven to be effective in the H_2/Si(100) model discussed in ref 18, where the ab initio calculations allowed optimization of an entire Si_9H_{12} cluster, while the network modeled only the coordinates of two H and two Si atoms.

Given the effectiveness of neural networks in representing a number of realistic multidimensional potential functions, it is somewhat surprising that there has been little effort to explore applications of neural networks in simple model systems. The insight available from simple problems may be a helpful guide to understanding the limitations of neural networks and improving their performance.

This paper has two aims: to describe a method for optimizing neural network parameters that is efficient and can tolerate noise in the available data and to develop a qualitative picture of neural network representations of potential functions through applications to several prototype problems. Section II describes extensions of earlier work[18,30] on parameter optimization to allow use of information about the derivatives of the potential and to allow network architectures with multiple hidden layers. In section III, we report applications of this approach to a number of simple functions of one, two, and three dimensions. The density of data and number of parameters needed to achieve a good fit, the effect of using gradient information in addition to energy information, effects of noise in the data, and convergence properties of the Kalman filter are discussed in section IV. The network's ability to extrapolate and the relationship between the form of the potential and the properties of the optimal network were also examined. These divisions among topics are artificial, since the network structure, quality of fit, optimization routine, and data used to achieve the fit are all intricately related. Full information about the fit achieved for each function is summarized in the figure captions.

II. Theory of Neural Networks and Kalman Filtering

We begin by reviewing the terminology of neural networks and the basic concepts of the parameter optimization routine.

A. Neural Network Architecture. The neural network is a nonlinear function of many parameters that maps particular inputs (in our case, coordinates) to an output (in our case, a potential energy). The network is composed of linear and nonlinear transfer functions, called neurons. For the networks we will consider, the neurons can be classified into different "layers" depending on where the input to the neuron comes from and how the output is used. The first ("input") layer takes an m-dimensional vector of coordinates, $\boldsymbol{x}$, at which the potential is to be evaluated and outputs an affine transformation of $\boldsymbol{x}$ to the nodes in the second ("hidden") layer. Each node in the hidden layer applies a nonlinear transfer function to its input. In the work described here, we have used the sigmoid transfer function, $(1 + e^{\boldsymbol{a}\bullet\boldsymbol{x}+b})^{-1}$. The outputs from this layer may become the inputs to another hidden layer of nonlinear functions. Whether there is one hidden layer or more, the outputs from the neurons in the last hidden layer become the inputs to the single neuron in the "output" layer, which performs an affine transformation to produce the network output. This paper will primarily concern a single-layer network architecture, which is essentially a superposition of sigmoids, i.e., the potential is approximated by

$$\tilde{V}(\boldsymbol{x};\boldsymbol{\theta}) = c_0 + \sum_{i=1}^{n} c_i (1 + e^{\boldsymbol{a}_i\bullet\boldsymbol{x}+b_i})^{-1} \qquad (1)$$

for a network with n hidden nodes. In this case, the adjustable parameters are the n vectors $\boldsymbol{a}_i$ (each with m components), the n values, b_i, and the $n+1$ values, c_i. These $n(m+2)+1$ parameters are collectively denoted by the vector $\boldsymbol{\theta}$. The goal is to find network parameters such that the network output is equal to the potential energy at the coordinates specified by the network input. Note that the network is defined at all values of the inputs and can be analytically differentiated with respect to both the coordinates and network parameters.

In the "feed-forward" architectures we have considered, neurons do not receive input from other neurons in the same layer, or from lower layers, so there is no feedback. This simplifies parameter fitting, while retaining adequate flexibility. We have considered only architectures in which all neurons in a given layer take inputs from all neurons in the layer above. Thus, for a given number of coordinates, the network architecture is completely defined by the number of hidden neurons (or, if there are multiple hidden layers, the number in each hidden layer).

Other transfer functions might be used, but the sigmoid has the advantages that it is bounded (so that the coefficients do not have to allow for cancellation between large positive and negative values), and it has a single variable region (so that fitting to local variation does not interfere with sigmoids that fit variation in other regions). Moreover, a fundamental theoretical result guarantees the efficacy of sigmoids: On a finite interval in $\mathbf{R}^n$, every differentiable function is the limit of a uniformly convergent sequence of finite sums of sigmoids.[31] As a result, a feed-forward network with a single, finite, hidden layer can uniformly fit any smooth, continuously differentiable function on the interval to within a specified tolerance.

Finding the optimal network for fitting a particular set of data involves both trying different numbers of hidden neurons and optimizing the parameters of each network configuration. In many cases, the quality of fit may be improved for a given network configuration by increasing the amount of data used in the parameter optimization. However, when the data are obtained from electronic structure calculations, it is much faster to optimize many different networks than generate significant amounts of new data. Thus, we generally do

Table 1. Equations of the Extended Kalman Filter[a]

$\lambda_k = \lambda_0 \cdot \lambda_{k-1} + (1 - \lambda_0)$	(T1)
$(\mathbf{D}_k)_{i,j} = (\boldsymbol{d}_k)_i^2 \cdot \delta_{i,j}$	(T2)
$\mathbf{H}_k = \partial \tilde{\boldsymbol{f}}(\mathbf{x}_k;\boldsymbol{\theta})/\partial \boldsymbol{\theta}\|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$	(T3)
$\mathbf{R}_k, \mathbf{Q}_k = (1 - \alpha)\cdot(\mathbf{R}_{k-1}, \mathbf{Q}_{k-1}) + \alpha \cdot \mathbf{D}_k$	(T4)
$\mathbf{A}_k = [\mathbf{I} + \lambda_k^{-1} \cdot \mathbf{H}_k^T \mathbf{P}_{k-1} \mathbf{H}_k + \mathbf{R}_k]^{-1}$	(T5)
$\mathbf{K}_k = \lambda_k^{-1} \cdot \mathbf{P}_{k-1} \mathbf{H}_k \mathbf{A}_k$	(T6)
$\mathbf{P}_k = \lambda_k^{-1} \cdot [\mathbf{I} - \mathbf{K}_k \mathbf{H}_k^T] \mathbf{P}_{k-1} + \mathbf{Q}_k$	(T7)
$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \mathbf{K}_k [\boldsymbol{f}(\boldsymbol{x}_k) - \tilde{\boldsymbol{f}}(\boldsymbol{x}_k;\boldsymbol{\theta}_k)]$	(T8)

[a] The filter is used to update the network parameters, $\boldsymbol{\theta}_k$, so that the network output, $\tilde{\boldsymbol{f}}(\boldsymbol{x};\boldsymbol{\theta}_k)$, approximates the desired output $\boldsymbol{f}(\boldsymbol{x})$, at the coordinates, $\boldsymbol{x}_k$. The significance of the other filter variables and suitable values are discussed in the text.

thorough searches of various network architectures in attempting to optimize the fit before adding more data.

B. The Extended Kalman Filter. For a given choice of network configuration, the network parameters must be optimized to fit the function. The optimization method must avoid local minima of the squared-error surface with respect to the parameters and should be stable in the presence of noise. The extended Kalman filter has this robustness and achieves a least-squares fit to data.[32] We provide here a qualitative guide to the filter, with enough details to allow others to reproduce our method. However, this section is not essential to the remainder of the paper.

A derivation of the extended Kalman filter and the role of each element in the filter can be found in ref 33, but the final equations are given in Table 1 (eqs T1−T8). For a finite data set with specified inputs (i.e., coordinate values), $\boldsymbol{x}$, the goal is to find a network model that produces the same result, $V(\boldsymbol{x})$, obtained from ab initio calculations at those points. Here we have denoted the desired output, $\boldsymbol{f}$, as a vector. In the simplest case, this will have a single element, the potential energy, $V(\boldsymbol{x})$. More generally, the network can be used to fit a vector of outputs. In particular, we will simultaneously fit the potential energy and its gradient, so that

$$\boldsymbol{f}(\boldsymbol{x}) = \left[V(\boldsymbol{x}), \frac{\partial V}{\partial x_1}\bigg|\boldsymbol{x}, \ldots, \frac{\partial V}{\partial x_m}\bigg|\boldsymbol{x} \right] \tag{2}$$

Having specified a network estimate for the potential, $\tilde{V}(\boldsymbol{x};\boldsymbol{\theta})$, it is a simple matter to differentiate this network with respect to the coordinates to obtain the corresponding estimate of the gradient. Thus, there is a known functional relationship, $\tilde{\boldsymbol{f}}(\boldsymbol{x};\boldsymbol{\theta})$, that gives the network approximation to $\boldsymbol{f}(\boldsymbol{x})$ for a given input, $\boldsymbol{x}$, and given network parameters, $\boldsymbol{\theta}$. Starting from an initial random guess at $\boldsymbol{\theta}$, the Kalman filter uses this functional relationship and comparisons between $\tilde{\boldsymbol{f}}(\boldsymbol{x};\boldsymbol{\theta})$ and $\boldsymbol{f}(\boldsymbol{x})$ to determine optimal parameters for the model network.

Unlike a traditional least-squares approach that examines the entire data set before determining a correction, the Kalman filter examines individual pieces of data to determine the next step. That is, the disparity between the "ideal" network output and the current network model is calculated for a single point in the data set using the current estimate of $\boldsymbol{\theta}$, denoted $\boldsymbol{\theta}_k$. The parameter set is updated to a new value, $\boldsymbol{\theta}_{k+1}$, to reduce the difference between $\tilde{\boldsymbol{f}}(\boldsymbol{x};\boldsymbol{\theta})$ and $\boldsymbol{f}(\boldsymbol{x})$. The filter is sequentially applied to each element of the data set, and then the data are randomly shuffled and examined again. This process continues until a converged value of $\boldsymbol{\theta}$ is reached. The sequential approach of the filter reflects the original application of the Kalman filter, which was developed for models of discrete time sequences.[34] In the present application, where the data have no intrinsic order, the sequence is randomized on each pass through the data, to eliminate any sequence dependence.

For linear models, the final parameters are the same as for a least-squares fit. For nonlinear problems, the Kalman filter corrections to θ_k are found by (locally) linearizing the problem and finding the zero, similar to Newton's method. This requires the matrix $\mathbf{H}_k$ of derivatives of $\tilde{\boldsymbol{f}}(\boldsymbol{x};\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ (eq T3), which are explicit analytic functions.

The filter does not assume the values in the data set are exact, and the updates do not jump straight to the solution as each piece of data is added. Instead, the data are assumed to include Gaussian-distributed noise, which the filter tries to remove by developing three covariance matrices, $\mathbf{P}_k$, $\mathbf{Q}_k$, and $\mathbf{R}_k$. The latter two, $\mathbf{Q}_k$ and $\mathbf{R}_k$ (eq T4), measure uncertainties of the parameters and data, respectively.[35] Even if there is no noise in the data, there will still be uncertainty in the optimal parameters because the network cannot fit the data exactly. $\mathbf{Q}_k$ and $\mathbf{R}_k$ also serve to add numerical stability by allowing the program to converge to a solution without fitting the data perfectly. These matrices are not known in advance and must be approximated,[36] though the approximations can often be crude. For this study, these matrices are approximated as a weighted average between the old matrix and an update matrix (eq T4), with $\mathbf{Q}_k$ and $\mathbf{R}_k$ initialized as zero matrices. The update matrix consists of the diagonal elements of the latest error vector. This is indicated in eq T2, where the correction to the covariance matrix ($\mathbf{Q}_k$ or $\mathbf{R}_k$) is indicated by $\mathbf{D}_k$ and the corresponding error vector is represented by $\boldsymbol{d}_k$. The error vector $\boldsymbol{f}(\boldsymbol{x}) - \tilde{\boldsymbol{f}}(\boldsymbol{x};\boldsymbol{\theta})$ is used to calculate the update for $\mathbf{R}_k$, while $\mathbf{H}_k(\boldsymbol{f}(\boldsymbol{x}) - \tilde{\boldsymbol{f}}(\boldsymbol{x};\boldsymbol{\theta}))$ is used in the update for $\mathbf{Q}_k$. α is an empirical weighting parameter (eq T4);[36] it is usually much less than unity ($\alpha \leq 0.15$).

The heart of the Kalman filter is $\mathbf{P}_k$, the covariance matrix that measures the uncertainty in the coefficients during the execution of the filter (eq T7). The exact definition of $\mathbf{P}_k$ and the derivation of its updates are given in ref 33. $\mathbf{P}_k$ stores information on the direction of previous parameter updates. The correction term in the update expression, $\mathbf{K}_k \mathbf{H}_k^T \mathbf{P}_k$, describes the change in network parameter uncertainty resulting from the update. We initialized $\mathbf{P}_k$ as a multiple of the identity matrix, with trace equal to the mean squared error, i.e.,

$$[\mathbf{P}_0]_{i,j} = \sum_{l=1}^{n} ||\boldsymbol{f}(\boldsymbol{x}_l) - \tilde{\boldsymbol{f}}(\boldsymbol{x}_l;\boldsymbol{\theta})||^2 \cdot \frac{1}{n} \cdot \delta_{i,j} \tag{3}$$

with $\boldsymbol{\theta}$ set to the initial guess of the parameter values.

With the statistics from $\mathbf{P}_k$, $\mathbf{Q}_k$, and $\mathbf{R}_k$ and the derivative, $\mathbf{H}_k$, the filter determines an optimal step to reduce the error in the network prediction for the current piece of data, without increasing the error in the fit to previous elements of the data set. This is achieved by moving in a direction (in parameter space) that satisfies an orthogonality condition while reducing the error in the current measurement. The

covariance matrices allow larger changes in those parameters with the greatest uncertainties. The net direction of the step is given by the gain matrix, $\mathbf{K}_k$, which is multiplied by the magnitude of the error to determine the correcting step size (eq T8).

We have modified the traditional extended Kalman filter by including a forgetting function, λ_k, that multiplies some terms (eqs T1 and T5−T7).[30] This gives the most recent steps in the algorithm a higher weight. During the course of the filtering operation, the forgetting function approaches unity to develop better statistics by giving high weights to a longer sequence of the previous data points. λ_0 is the constant that determines the rate at which λ_k approaches unity and λ_1 is the initial value of λ_k. The values of λ_0 and λ_1 are close to unity ($\lambda_0 \geq 0.99$, $\lambda_1 \geq 0.90$), but specific choices for their values depend on how often λ_k is updated as well as the data and potential.[30]

III. Network Architecture and Quality of Fit

In principle, a finite number of hidden-layer neurons can uniformly fit any well-behaved function to a specified tolerance.[31] We have investigated the pragmatic issue of how many neurons are needed to interpolate some common functions to a reasonable degree of accuracy. Clearly, the number of neurons needed will depend on the function being fitted. The sigmoid transfer function has a single variable region and two constant regions. To fit a function, the network uses the variable region of each sigmoid to match the variation in the function in some region, with multiple neurons used to fit the function over an extended domain. Because the sigmoid function is monotonically increasing, a single sigmoid cannot fit a function where the derivative changes sign. An interval between points where the derivative changes sign will be referred to as a monotonic region. The number of monotonic regions being fit is a lower bound on the number of hidden-layer neurons needed to fit the function.

In this section, we show the fits achieved for some exemplar functions with features that are common to typical interatomic interaction potentials. We explore the actual number of neurons needed to achieve a qualitative fit and the effect of additional neurons on improving the quantitative fit. Unless stated otherwise, the domain and range for all functions in this study is the unit interval. Data for training the function were selected from this interval as well. Errors reported are the integrated root-mean-square difference between the network and the function being fitted, calculated by integration over the unit interval (the functions are known exactly, there is no need to base errors on a finite test set).

A. One-Dimensional Model Functions. *1. Harmonic Oscillator and Morse Potentials.* As a simple example we begin with the harmonic oscillator on the positive *x*-axis, which has only one monotonic region. A single sigmoid gives a good qualitative fit to this half-oscillator, though the best fit achieved with one sigmoid had an rms error of 10^{-2}. Turning to a full harmonic oscillator (with the minimum at $x = 0.5$; see Figure 1), there are two monotonic regions and at least two hidden-layer neurons are required. A network with two sigmoids fits the function to a surprisingly high accuracy, with a rms error on the order of 10^{-3} over the unit interval (unshaded region of Figure 1).

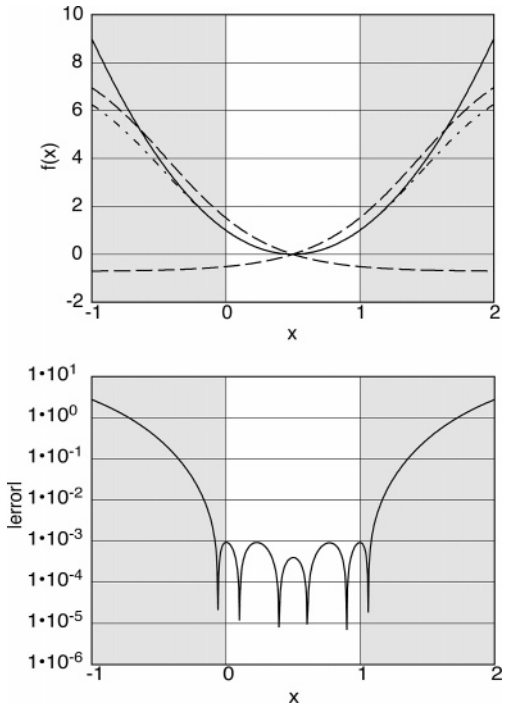

Figure 1. (Top) Harmonic oscillator potential, $V(x) = 4(x - 1/2)^2$ (solid curve), the individual outputs of the two neurons used in the network fit (dashed curves), and the total network fit (dot−dashed curve). The outputs due to the individual neurons are shifted vertically, as described in the text, to aid in visualizing the fit. (Bottom) Error in the neural network fit as a function of *x*; the integrated rms error is 0.8×10^{-3}. Energy and derivative data at six equally spaces points in the unshaded region ($0 \leq x \leq 1$) were used to optimize the network parameters. Each neuron predominantly fits one side of the oscillator. However, each neuron also makes a non-negligible contribution to the other side of the oscillator function, which prevents use of a single neuron and symmetry to determine the parameters. The network fit remains qualitatively correct as it extrapolates into the shaded region, where no data were used in parameter optimization.

Plotting the contributions of individual neurons (Figure 1) demonstrates why the fit for the full oscillator is more accurate than that for the half-oscillator. These individual-neuron contributions are determined by turning off the output from other neurons in the hidden layer. For clarity, a constant has been added to the resulting network output is equal to the fitted function at the point where the single-neuron network and the fitted function have equal derivatives. For the full harmonic oscillator, most of the variation in each monotonic region is matched by a single neuron, as in the fit of the half-oscillator, but the second neuron makes a nontrivial contribution that improves the fit. Although no symmetry constraints were imposed on the network, the two optimal neurons are related by the reflection symmetry of the oscillator. Note that the small region of positive curvature at the bottom of each sigmoid makes the dominant contribution over the range $0 \leq x \leq 1$. This is achieved by scaling the coordinates with fairly large coefficients.

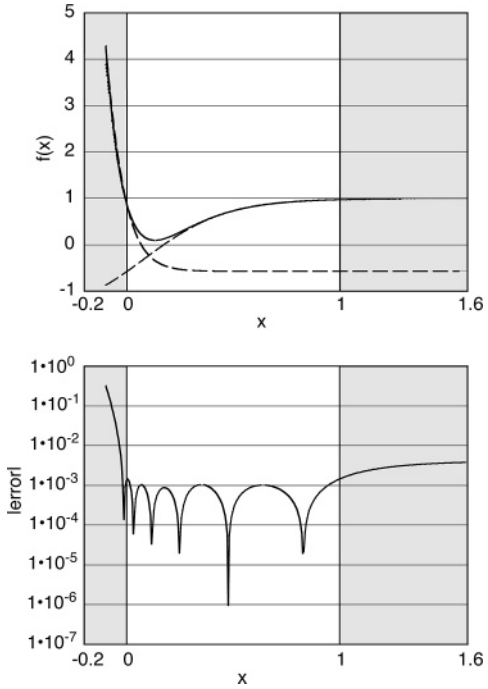

Figure 2. (Top) Morse potential, $V(x) = 0.1 + 0.9(1 - \exp(0.65-5x))^2$ (solid curve), the individual outputs due to the two neurons used in the fit (dashed curves), and the total network fit (dotted curve). (Bottom) Error in the neural network fit as a function of x; the integrated rms error is 2.0×10^{-3}. Energy and gradient data at 21 points in the unshaded region ($0 \le x \le 1$) were used for parameter optimization. Note the range over which extrapolation remains accurate. A fit with data at only 11 points had a similar rms error of 2.2×10^{-3} but does not extrapolate as well as the fit shown.

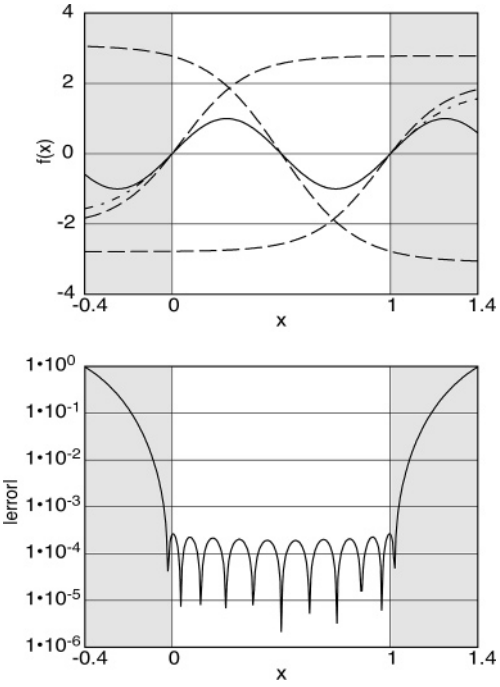

Figure 3. (Top) Single-period sine function, $V(x) = \sin(2\pi x)$ (solid curve), individual output from the three neurons used to fit it (dashed curves), and the total network fit (dot−dashed curve). (Bottom) Corresponding error plot; the integrated rms error is 1.0×10^{-3}. Energy and gradient data at 21 equally spaced points in the unshaded region ($0 \le x \le 1$) were used for parameter optimization. Each monotonic segment is primarily fit by a single neuron. The symmetry of the function being fit is reflected by symmetry relations among the neurons, with one antisymmetric neuron and the other neurons related by translation. These symmetry relations make the network fit symmetric, even when extended to the (shaded) extrapolation region.

Two neurons are also successful at fitting the Morse potential to 10^{-3} rms error (Figure 2). This illustrates the flexibility of networks with the minimum number of neurons. However, convergence was difficult to achieve for the Morse potential, and the search had to be repeated with several choices of initial network parameters. The large derivatives of the repulsive wall are the likely origin of this difficulty. Matching steep derivatives requires large network coefficients, so the filter must search a large region of coefficient space to achieve the optimal fit.

2. Sine Waves. Figure 3 illustrates the fit to a sine wave. Good accuracy (10^{-4} rms error) was achieved with three neurons, which is again the minimum number for this case. Again, most of the variation in each region of high slope is matched by a single neuron, but there is an appreciable contribution from the other neurons. The network configuration also reflects the symmetry of the function. One neuron has the antisymmetry of the sine wave and the other two neurons are related by a periodic translation.

Functions with several monotonic regions, like the double-period sine wave, are more difficult to fit (Figure 4). Over the unit interval, the double-period sine wave has 5 monotonic regions, requiring a minimum of 5 neurons. With so many neurons, several are varying at any given coordinate value, causing more interference among them. The best fit with 5 neurons had an rms error of 10^{-1} (Table 2). Four additional compensating neurons allowed a fit with 10^{-4} rms error. This is a rather subtle effect: Table 2 shows that there is a dramatic decrease in the error upon increasing the network from eight to nine neurons. This is not simply the gradual improvement expected with addition of more parameters; a full set of four compensating neurons appears to be an important addition to the model.

The symmetry relationships between individual neurons are less clear as the number of neurons increases, but the dominant neurons of the network for the double-period sine wave retain some symmetry properties. The plots of major contributing sigmoids in Figure 4 show one antisymmetric sigmoid and two pairs of sigmoids that are related by periodic translation and reflection. However, the additional compensating neurons needed for a more accurate fit show no clear symmetry. As a result, symmetry is not maintained on extrapolation beyond the unit interval. If the network fails to reflect some of the symmetry of the system, it may indicate that the network has too many (or too few) neurons. A single neuron on one side of a symmetry plane might be matching the same variation as multiple neurons on the other side of the plane. For example, initial searches for a network with nine neurons had problems converging for the double-period sine wave, while a ten-neuron network converged quickly. However, the ten-neuron network did not reflect any of the

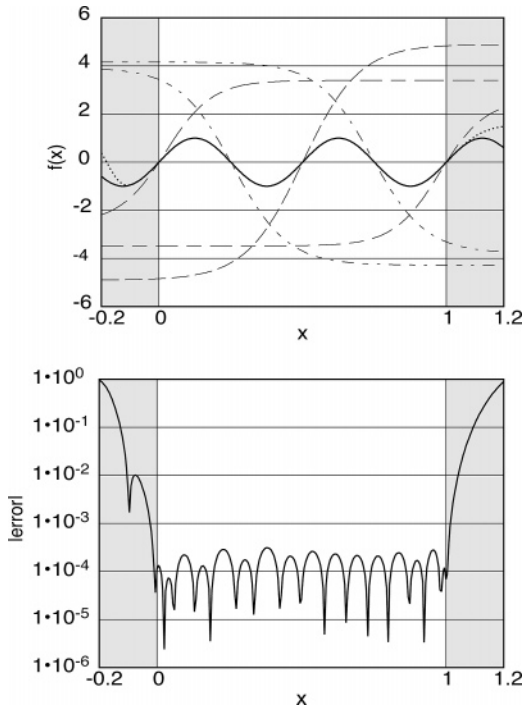

Figure 4. (Top) Double-period sine function, $V(x) = \sin(4\pi x)$ (solid curve), individual outputs from the five neurons that fit the majority of the variation (various dashed curves), and the total nine-neuron network fit (dotted curve). (Bottom) Corresponding error plot for a fit with nine neurons; the integrated rms error is 0.12×10^{-3}. One neuron is antisymmetric, and the other four neurons are related by reflection and translation operations. Even with energy and gradient data at 101 points in the unshaded region ($0 \le x \le 1$), these five neurons alone cannot achieve a quantitative fit. Four other neurons (not shown) make small contributions to the fit (magnitudes less than 0.06 in the fitted region). The additional neurons quickly grow outside the fitted region and cause the breakdown in symmetry.

symmetry of the system. Further searches for a nine-neuron network were eventually successful in achieving the fit described above, in which the dominant contributions reflect the symmetry of the function.

B. Higher Dimensional Models. As the number of dimensions increases, more neurons are generally needed, though there is no clear lower bound on their number. A single sigmoid transfer function varies only along one generalized coordinate. This direction is a linear transformation of the input coordinates for the single hidden-layer networks considered here, though a nonlinear dependence on the input coordinates could be achieved with a multiple hidden-layer network.

As in one dimension, a few neurons usually account for most of the variation in the examples we have examined. This can be seen for a two-dimensional model made by coupling a Morse potential in one direction to a sinusoidal wave in the other (Figure 5). Two neurons account for most of the repulsive wall of the Morse potential, while two other neurons account for the asymptotic approach to zero at large bond lengths. Six other neurons account for small variations caused by the more subtle sinusoidal element. Similar trends

Table 2. Sensitivity of Network Error to Number of Neurons for the Double-Period One-Dimensional Sine Function and the Two-Dimensional Anharmonic Oscillator[a]

no. of neurons	RMS error (% of scale) (%)
1-D Double-Period Sine Wave (Energy and Derivative at 101 Points; No Noise)	
5	8.38
8	0.71
9	0.012
10	0.019
2-D Anharmonic Oscillator (Energy and Gradient at 36 Points; No Noise)	
3	0.17
4	0.13
5	0.16
2-D Anharmonic Oscillator (Energy and Gradient at 100 Points; 6.5% rms Added Noise)	
3	3.0
4	0.9
5	1.6

[a] In each case, over 30 converged optimizations were performed to find the best fit.

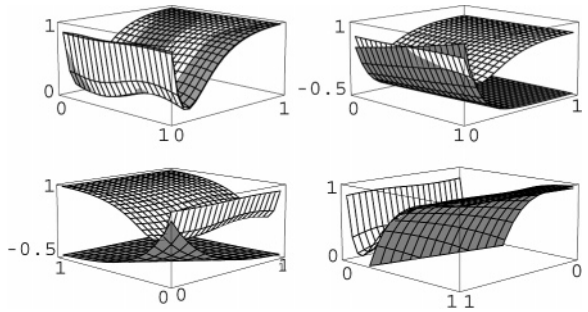

Figure 5. Two-dimensional surface model, $V(x, y) = 1 + 0.1\text{-}(\sin(2\pi x) + 9)\,((1-e^{0.65-5y})^2 - 1)$ (white surface in all figures). This function is compared to the best network fit (gray surface, upper left figure) and the individual contributions of three neurons (gray surfaces) that account for different aspects of the variation in the function. The network fit used 10 hidden-layer neurons; energy and gradient data at 121 points were used to achieve an rms error of 0.9×10^{-3}.

were observed in a three-dimensional surface model (not shown) in which the two-dimensional model of Figure 5 was coupled to a sinusoidal variation in the third direction.

The two-dimensional anharmonic oscillator (Figure 6) illustrates some of the subtleties of finding an optimal network. The quality of the fit is quite good with three, four, or five neurons, though the best fit was achieved with four neurons (Table 2). It was difficult to find a converged solution for a three-neuron model, though in the end the rms error was low. The five-neuron model has one neuron that is virtually inactive. The fifth neuron actually increases the error in the fit by allowing the network to over-fit the data. With added noise in the data, the quality of the fit is even more sensitive to the number of neurons (Table 2). If there is any noise in the data, a network with more parameters can fit the noise, so the importance of minimizing the number of neurons is magnified.

A two-dimensional sine function (Figure 7) has more sign changes than the two-dimensional oscillator or surface model.

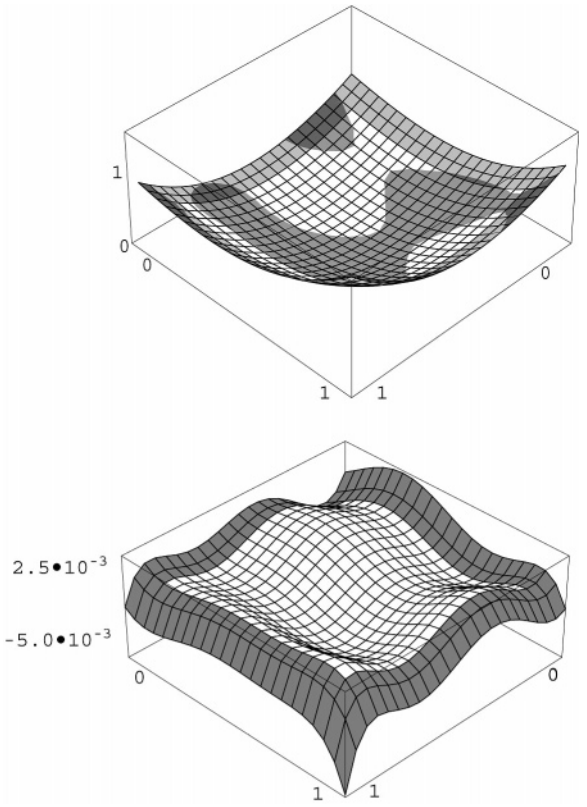

Figure 6. (Top) Two-dimensional anharmonic oscillator (with coefficients chosen as rational approximations to random numbers, with the condition that there is a single minimum and no maxima in the region being fit), $V(x, y) = (4/9)x^3 + (2/27)x^2y + (2/45)xy^2 + (16/27)y^3 + (124/135)x^2 + (4/75)xy + (358/675)y^2 - (1042/1125)x - (76/125)y + (1684/5625)$ (white surface) and the four-neuron network fit to the function (gray surface). (Bottom) Corresponding error plot. Energy and gradient data at 36 equally spaced data points were taken from the unshaded region (note in this case, $0 \leq x, y \leq 0.9$) to fit the network parameters. The network extrapolates this surface into the shaded region and achieves an rms error of 1.3×10^{-3} in the unshaded region.

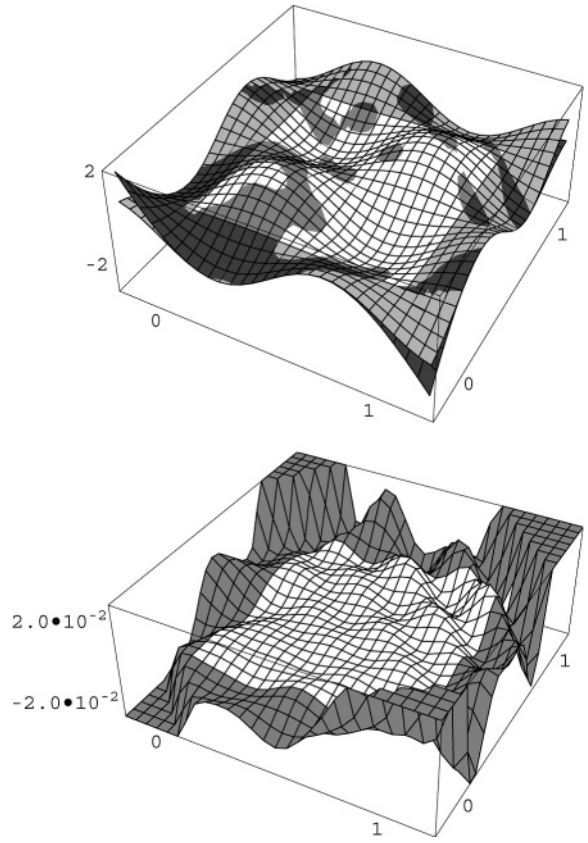

Figure 7. (Top) Two-dimensional sinusoidal surface model, $V(x, y) = \sin(2\pi x)\sin(2\pi y)$ (white surface). (Bottom) Corresponding error plot. The function was fit with 12 neurons. Energy and gradients at 121 points (for a total of 363 pieces of data) were required to achieve the fit in the unshaded region ($0 \leq x, y \leq 1$) to an rms error of 3×10^{-3}. The network extrapolation breaks down at the corners, the points farthest from the data used for parameter optimization.

In this case, an error of 3×10^{-3} is achieved with 12 sigmoids. This can be compared to the three sigmoids needed for comparable accuracy in the one-dimensional case.

C. Extrapolation. As with any interpolation method, neural networks are not expected to make accurate predictions beyond the region where data are available for parameter optimization. However, extrapolation is accurate over a surprising range for many of the functions (Figures 1−7). Typically, the network fit is qualitatively reasonable to about 10% beyond the domain where the function was fitted, though in some cases it can be more. For the two-dimensional sinusoid (Figure 7), this extrapolation breaks down around the corners of the extrapolation, i.e., the points farthest from the data used for parameter optimization. The network extrapolation of the higher curvature functions, like the double period sine wave (Figure 4), is accurate over a much shorter range than lower curvature functions, like the single-period sine wave and the large-x side of the Morse potential (Figures 2 and 3). Since the sigmoid eventually becomes constant, it extrapolates best for functions that approach a constant (e.g., the Morse oscillator at large x, Figure 2).

D. Degeneracy. We have found many examples where different optimizations of the same network with the same data yield different coefficients, while the rms errors of the different optimizations are almost identical. These network models are distinct in the sense that no simple permutation of the neurons makes them identical. We refer to such sets of networks as degenerate. One might expect degeneracy in networks with too many parameters, since making the network too flexible would allow it to fit the data many different ways. However, we have observed degeneracy in networks that do not over-fit the data. For the two-dimensional anharmonic oscillator in Figure 6, three neurons fit the function very well (rms error of 1.7×10^{-3}) and do not over-fit the function, since the optimal fit was achieved with four neurons (rms error of 1.3×10^{-3}). Nevertheless, we have found 10 distinct sets of coefficients for the three-neuron model. Plots of the error surface show at least two distinct topologies, though several sets of network parameters produce similar fits to the function. Some cases of degen-

eracy may result from the symmetry of the sigmoid function

$$1-(1+e^{-ax+b})^{-1} = (1+e^{ax-b})^{-1}$$

but this cannot explain the different error topologies.

E. Multiple Hidden Layers. In principle, using multiple hidden layers can provide an alternate model that may require fewer parameters than a single-layer network. Additional layers act as nonlinear coordinate transformations that become the input to the final layer. The shape of the network fit is still a superposition of the sigmoids from the final layer, though the nonlinear coordinate transformation can add curvature to the sigmoids in multiple directions. However, our results for multiple layers in one- and two-dimensional examples are not encouraging. We find that more parameters are required to fit functions with multiple layers than a single layer. On the other hand, other groups studying higher-dimensional systems have found multiple layers to be useful,[23,24] so this issue deserves further exploration.

IV. Parameter Optimization and Data Density

The examples above show the type of fits that can be achieved with neural networks, though we have not addressed the kind of data needed for successful parameter optimization. The number and distribution of elements in the data set, the type of data available (energies or energies and derivatives), and conditioning of the data will all have a significant impact on the ability of the optimization algorithm to find network parameters to fit the function. In this section we describe the quality of fit that can be achieved when different types of data are used to determine network parameters. To isolate the effects of the data sets, we have used only the network architecture that yields the best fit for each function.

A. Functional Form and Data Density. Since functions requiring more neurons for an accurate fit will have more parameters in the network, we expect that more data will be needed to accurately determine all the parameters. This is observed in the two sine waves described above. For a single-period sine wave (Figure 3), which is fit with three neurons and a total of 10 parameters, only 42 pieces of information (the energy and derivative at 21 different points) are required to achieve a rms error of less than 10^{-3}. For the shorter wavelength sine wave (Figure 4), 202 pieces of data (energies and derivatives at 101 points) are required to get a comparable fit with nine neurons and 28 parameters. In every case reported, including those with noise added to the data, fewer than 10 pieces of data per parameter (whether energy or derivative data) were needed for accurate fits. The greatest amount of data is needed when there are large network coefficients associated with high gradients (as for the Morse potential) or curvature, because the output is sensitive to small changes in the large parameters.

B. Gradient Data. In the prior section, we described fits using a combination of energy and derivative data. Most prior work with neural networks has used energy data alone for parameter optimization. Here we explore the effectiveness of using gradient data in addition to the energy. Intuitively, one expects that a fit to both energy and gradient data will lead to smaller fluctuations between points in the data set than a fit to energy values alone. This expectation is borne out in fits to the Morse potential where energy and derivative data at 11 points results in a better fit (2×10^{-3} rms error) than energy data alone at 41 points (1×10^{-2} rms error). Thus, gradient data can dramatically reduce the density of data needed, an important consideration if these data are to obtained from ab initio calculations. Moreover, many ab initio methods can calculate the gradient at a computational expense comparable to that of the energy, so m components of the gradient vector can be obtained for the effort of one energy (m denotes the number of coordinate inputs to the network). In many-dimensional systems, this may partly compensate for the increased size of the space to be sampled. Other partial derivative information (such as the curvature) could be used as well, though such data are less likely to be available.

Table 3. Effects of Gradient Information and Data Density on the Fit to the Two-Dimensional Anharmonic Oscillator (Figure 6) with 4 Neurons

no. of data points	RMS error (% of range)	
	no gradients (%)	with gradients (%)
	Date with No Noise	
36	0.4	0.1
100	0.2	0.1
	Data with 6.5% rms Added Noise	
36	3.2	1.0
49	2.4	1.0
100	1.4	0.9

Several other examples confirm that use of gradient data allows sampling at many fewer points than needed with energy data alone. Table 3 shows the results of several fits to a two-dimensional anharmonic oscillator using uniformly spaced data. A fit with four neurons using energy data at 36 points gives an rms error of 0.4%. This error is cut in half if energy data are used at 100 points. However, using both energy and gradient data at only 36 points gives a fit with even smaller error (0.1%). Since the gradient is a two-dimensional vector, there are three pieces of data at each of the 36 points or 108 pieces of information in total. Thus, in this example, a lower-density data set with energy and gradient information is more effective in allowing a good fit than energy data alone at higher-density (with a similar total amount of information).

The two-dimensional surface model (Figures 5) also shows how gradient information improves the fit. Using energy data alone at 361 points, the filter was unable to match the sinusoidal variation of the function and a best fit of 2.4×10^{-2} rms error was achieved. Using gradient information (in addition to energies) at 121 points allowed a fit to less than 10^{-2} rms error. Again, while the total amount of information is the same in these two data sets, the combination of energy and gradient information is more effective in achieving a good fit, and (if analytic derivatives are available in an ab initio calculation) the combined data will be more economical to calculate.

C. Noise. The Kalman filter is designed to allow for noise in the data, as might be caused by variation in unmodeled

coordinates. To test this capability, we have added white noise with a root-mean-square value of 6.5% of the total range to both the energy and gradient data from the two-dimensional anharmonic oscillator. With energy and derivative information at 36 points, the network removed most of the added noise, fitting the oscillator function to 1% error (Table 3). Increasing the data density to 100 points helps very little, with a remaining error of 0.9%. Even with noise added to the gradients, they improve the fit: without gradients, energies at 100 points are needed to achieve 1.4% rms error. Clearly, this method is effective even when there is significant variation due to unmodeled coordinates.

D. Filter Convergence. The initial guess at the network parameters is random, so there is a risk that the optimizations will converge on a local minimum in parameter space. The worst case in our experience was for the 3-dimensional surface model (Morse potential coupled to 2D sinusoidal variation) discussed above. Local minima were found in 90% of the runs. A simple solution is to do a series of optimizations and take the best results, though it is never clear how many examples might be needed. Nevertheless, the computing requirements for a network optimization are relatively small. Most of the calculations described here take only a few minutes on a typical desktop computer. The number of iterations through the whole data set depends on the function being fit and the size of the data set. Fewer iterations are needed when more data are used, presumably because of redundancy in the data.

We have found examples where the network parameters do not converge at all for some initial parameter sets. For example, if there are not enough neurons to match the variation of the function, the coefficients of the network will oscillate. This behavior has been observed even in networks that can fit almost all of the variation, like the three-neuron fit to the two-dimensional anharmonic oscillator. On the other hand, if the network has too many neurons, there are more likely to be local minima.

V. Summary and Final Comments

The neural network using sigmoidal neurons is very flexible and can fit a range of functions typical of interatomic potentials in low dimensions, with a small number of neurons and a modest amount of data. The unbiased nature of the network fit is illustrated by the remarkable fact that a two-neuron network can model both the harmonic and Morse oscillators with comparable accuracy over a wide range. No assumptions about the functional form of the potential are required to achieve a good representation. The resulting function is infinitely differentiable and globally defined. The fit is not only merely local, like a spline, but can also reproduce global features, such as symmetry, without special choices of coordinates. Even in the presence of random noise, the network can recover the underlying relationship between coordinates and potential. Thus, neural networks remain promising candidates for fitting models of interatomic interactions.

The extended Kalman filter is a robust method of achieving a least-squares fit of network parameters to the data. However, the number of data points needed to sample the variation in a function will increase when there is significant variation in more dimensions. Using gradient data is an economical way to obtain information about variation in higher-dimensional functions, which partly compensates for the need to sample more coordinates. For many systems, it may also be possible to isolate the dependence on a few degrees of freedom, treating most degrees of freedom as a statistically defined bath. However, more efficient methods for data sampling will ultimately have to replace the uniform sampling used here. Collins[9] has developed an approach that samples configurations that are likely to occur in a thermal reaction process. By running classical molecular dynamics (MD) trajectories on an estimate of the potential, important regions of configuration space can be found, and new ab initio data can be calculated in those regions. Another example of trajectory-based sampling has been reported by Wood et al.[37] in an application to solvation thermodynamics. Classical molecular dynamics (MD) simulations are run with an approximate potential, followed by ab initio calculations of solute−solvent interactions at a random sample of configurations. Thermodynamic perturbation theory is used to calculate the difference in free energy between the approximate and ab initio potentials. However, if the initial approximate potential is not reasonably close to the ab initio potential, there will be large sampling errors. In such a case, the approximate potential can be improved using information from the ab initio calculations (followed by new MD simulations with the improved potential).[37] Since these ab initio calculations are done at configurations sampled from a thermal ensemble (albeit with an approximate potential), they are inherently more dense at the low-energy configurations that dominate the thermal ensemble. Neural networks may be a convenient tool for modeling these improvements to the approximate potential.

Acknowledgment. This material is based upon work supported by the National Science Foundation under Grants No. 9971241 and 0316223.

References

(1) Ho, T.-S.; Rabitz, H. *J. Chem. Phys.* **1996**, *104*, 2584. Hollebeek, T.; Ho, T.-S.; Rabitz, H. *Annu. Rev. Phys. Chem.* **1999**, *50*, 537. Ho, T.-S.; Rabitz, H. *J. Chem. Phys.* **2000**, *113*, 3960.

(2) Frishman, M.; Hoffman, D. K.; Rakauskas, R. J.; Kouri, D. J. *Chem. Phys. Lett.* **1996**, *252*, 62. Frishman, D. K.; Frishman A.; Kouri, D. J. *Chem. Phys. Lett.* **1996**, *262*, 393.

(3) Boothroyd, A. I.; Keogh, W. J.; Martin, P. G.; Peterson, M. R. *J. Chem. Phys.* **1991**, *95*, 4343; **1996**, *104*, 7139.

(4) Mielke, S. L.; Garrett, B. C.; Peterson, K. A. *J. Chem. Phys.* **2002**, *116*, 4142.

(5) Salazar, M.; Simons, J. *J. Chem. Phys.* **1996**, *105*, 10919.

(6) Hack, M. D.; Truhlar, D. G. *J. Chem. Phys.* **1999**, *110*, 4135.

(7) Pu, J. Z.; Truhlar, D. G. *J. Chem. Phys.* **2002**, *116*, 1468.

(8) Szalay, V. *J. Chem. Phys.* **1999**, *111*, 8804.

(9) Bettens, R. P. A.; Collins, M. A. *J. Chem. Phys.* **1999**, *111*, 816. Duncan, A. H.; Collins, M. A. *J. Chem. Phys.* **1999**, *111*, 1346. Zhang, D. H.; Collins, M. A.; Lee, S.-Y. *Science* **2000**, *290*, 961. Moyano, G. E.; Collins, M. A. *J. Chem. Phys.* **2003**, *119*, 5510. Crespos, C.; Collins, M. A.; Pijper, E.; Kroes, G. J. *J. Chem. Phys.* **2004**, *120*, 2392.

(10) Lopez, M. J.; Jellinek, J. *J. Chem. Phys.* **1999**, *110*, 8899.

(11) Berweger, C. D.; van Gunsteren, W. F.; Müller-Plathe, F. *J. Chem. Phys.* **1998**, *108*, 8773.

(12) Mattson, T. R.; Wahnström, G.; Bengtsson, L.; Hammer, B. *Phys. Rev. B* **1997**, *56*, 2258.

(13) Gross, A.; Scheffler, M.; Mehl, M. J.; Papaconstantopoulos, D. A. *Phys. Rev. Lett.* **1999**, *82*, 1209.

(14) Olsen, R. A.; Busnengo, H. F.; Salin, A.; Somers, M. F.; Kroes, G. J.; Baerends, E. J. *J. Chem. Phys.* **2002**, *116*, 3841.

(15) Liu, Y.-P.; Kim, K.; Berne, B. J.; Friesner, R. A.; Rick, S. W. *J. Chem. Phys.* **1998**, *108*, 4739.

(16) Burnham, C. J.; Xantheas, S. S. *J. Chem. Phys.* **2002**, *116*, 1479; **2002**, *116*, 1493; **2002**, *116*, 1500; **2002**, *116*, 5115.

(17) Groenenboom, G. C.; Mas, E. M.; Bukowski, R.; Szalewicz, K.; Wormer, P. E. S.; van der Avoird, A. *Phys. Rev. Lett.* **2000**, *84*, 4072. Mas, E. M.; Bukowski, R.; Szalewicz, K.; Groenenboom, G. C.; Wormer, P. E. S.; van der Avoird, A. *J. Chem. Phys.* **2000**, *113*, 6687. Mas, E. M.; Bukowski, R.; Szalewicz, K. *J. Chem. Phys.* **2003**, *118*, 4386.

(18) Blank, T. B.; Brown, S. D.; Calhoun, A. W.; Doren, D. J. *J. Chem. Phys.* **1995**, *103*, 4129.

(19) Skinner, A. J.; Broughton, J. Q. *Modell. Simul. Mater. Sci. Eng.* **1995**, *3*, 371.

(20) Brown, D. F. R.; Gibbs, M. N.; Clary, D. C. *J. Chem. Phys.* **1996**, *105*, 7597.

(21) No, K. T.; Chang, B. H.; Kim, S. Y.; Jhon, M. S.; Scheraga, H. A. *Chem. Phys. Lett.* **1997**, *271*, 153.

(22) Prudente, F. V.; Soares Neto, J. J. *Chem. Phys. Lett.* **1998**, *287*, 585. Prudente, F. V.; Acioli, P. H.; Soares Neto, J. J. *J. Chem. Phys.* **1998**, *109*, 8801.

(23) Gassner, H.; Probst, M.; Lauenstein, A.; Hermansson, K. *J. Phys. Chem. A* **1998**, *102*, 4596.

(24) Lorenz, S.; Gross, A.; Scheffler, M. *Chem. Phys. Lett.* **2004**, *395*, 210.

(25) Muñoz, C.; Niño, A. *Comput. Chem.* **1998**, *22*, 355.

(26) Hobday, S.; Smith, R.; Belbruno, J. *Modell. Simul. Mater. Sci. Eng.* **1999**, *7*, 397.

(27) Cho, K.-W.; No, K. T.; Scheraga, H. A. *J. Mol. Struct.* **2002**, *641*, 77.

(28) Rocha Filho, T. M.; Olivera, Z. T., Jr.; Malbouisson, L. A. C.; Gargano, R.; Soares Neto, J. J. *Int. J. Quantum Chem.* **2003**, *95*, 281.

(29) Bittencourt, A. C. P.; Prudente, F. V.; Vianna, J. D. M. *Chem. Phys.* **2004**, *297*, 153.

(30) Blank, T. B.; Brown, S. D. *J. Chemometrics* **1994**, *8*, 391.

(31) Cybenko, G. *Mathematical Control Signals Systems* **1989**, *2*, 303.

(32) Puskorius, G. V.; Feldkamp, L. A. *IEEE Trans. Neural Networks* **1994**, *5*, 279.

(33) Gelb, A. *Applied Optimal Estimation;* MIT Press: Cambridge, 1974.

(34) Kalman, R. E. *J. Basic Eng. Ser. D* **1960**, *5*, 35.

(35) Grewal, M. S.; Andrews, A. P. *Kalman Filtering: Theory and Practice*; Prentice Hall: Englewood Cliffs, 1993.

(36) Tsoi, A. C. Gradient Based Learning Methods. In *Adaptive Processing of Sequences and Data Structures*. (Lecture Notes in Computer Science; Vol. 1387: Lecture notes in artificial intelligence). Giles, C. L., Gori, M., Eds.; Springer-Verlag: New York, 1998.

(37) Wood, R. H.; Yezdimer, E. M.; Sakane, S.; Barriocanal, J. A.; Doren, D. J. *J. Chem. Phys.* **1999**, *110*, 1329. Sakane, S.; Yezdimer, E. M.; Liu, W.; Barriocanal, J. A.; Doren, D. J.; Wood, R. H. J. *Chem. Phys.* **2000**, *113*, 2583.

CT049976I