The Use of Neural Networks for Fitting Potential Energy Surfaces: A Comparative Case Study for the H_3^+ Molecule

T. M. ROCHA FILHO,[1,2] Z. T. OLIVEIRA, JR.,[2]
L. A. C. MALBOUISSON,[2] R. GARGANO,[1] J. J. SOARES NETO[1]

[1]*Instituto de Física, Universidade de Brasíla 70919-970, Brasília, DF, Brazil*
[2]*Instituto de Física, Universidade Federal da Bahia 40210-340, Salvador, Bahaia, Brazil*

Received 18 November 2001; accepted 6 April 2003

DOI 10.1002/qua.10696

ABSTRACT: The fitting of ab initio electronic energies of polyatomic molecules for different nuclear configurations is an active field in quantum chemistry and is an important step in the study of chemical reaction dynamics and for the determination of rovibrational spectra. The choice of a good-fitting function and the decision as to which geometries are relevant for the problem remains a matter of feeling as a large number of ab initio points of good quality usually involves prohibitively large amounts of CPU times. More recently, the use of neural networks has drawn some attention for fitting potential energy surfaces (PES). Neural networks are generic function approximators for any required accuracy and are therefore well suited for fitting many-dimensional PES. In this work we present a comparative study for the ground state PES of the $H3^+$ molecule obtained fitting state-of-the-art ab initio points. The PES is obtained using both a neural network and a polynomial function in Morse-type symmetry-adapted coordinates. The quality of the surfaces is asserted by computing the associated rovibrational spectra. The resulting energies are compared with known experimental results. © 2003 Wiley Periodicals, Inc. Int J Quantum Chem 95: 281–288, 2003

Key words: H_3^+ molecule; neural networks; potential energy surface; vibrational energy

Correspondence to: T. M. R. Filho; e-mail: marciano@fis.unb.br
Contract grant sponsor: CNPq.

Introduction

In the study of properties of molecular systems such as rovibrational levels, photodissociation cross-sections and chemical reaction dynamics usually require the knowledge of a potential energy surface (PES) for the nuclear motion under the Born–Oppenheimer approximation [1] and is also required for nonadiabatic corrections of the dynamics. The PES is obtained by fitting ab initio and/or experimental data using a functional form, usually a polynomial in a coordinate system chosen to incorporate any symmetry in the system. Examples are the Morse symmetry–adapted deformation coordinates [2] and the bond-order coordinates [3], among others [1].

More recently, some articles dedicated some attention to the use of a neural network (NN) for fitting a PES [4–9]. Neural networks have also been applied for the estimation of correlation energy in diatomic molecules [10], for the determination of the PES of a macromolecule by learning the relationship between the vibrational spectra and the PES itself [11], and for constructing maps of PES and observables [12]. The use of a NN in physics and chemistry is not devoid of some criticism [13], and the pros and cons of its use still need further investigation.

The current article aims to discuss some issues on the relevance of the use of NNs for PES fitting by an application to the ground state of the H_3^+ molecule. Being a two-electron system, it allows for very accurate electronic structure calculations. In Ref. [2], Cencek et al. obtained ab initio confidence interval (CI) calculations plus adiabatic and relativistic corrections resulting in a sub–micro-Hartree accuracy for 69 geometries relevant for the determination of vibrational levels. These were obtained in a subsequent article [14] using a 7th-, 9th-, and 10th-order polynomial in Morse-type symmetry–adapted coordinates, and improved by introducing nonadiabatic corrections to the nuclear dynamics [15]. A fit using an NN was obtained in Ref. [4] but using lower precision electronic energies [16, 17]. In the current work, we obtain a PES for the H_3^+ molecule using an NN to fit the points in Ref. [2]. Both this PES and the one obtained in Ref. [14] are used to compute vibrational energy levels of the molecule. The results obtained are then compared with experimentally determined band origins. This is the first comparative study, to the authors' knowledge, of the use of NNs for PES fitting relative to a standard fit procedure, with a comparison to experimental data, and sheds some light on the issue as to whether and when to use the current approach.

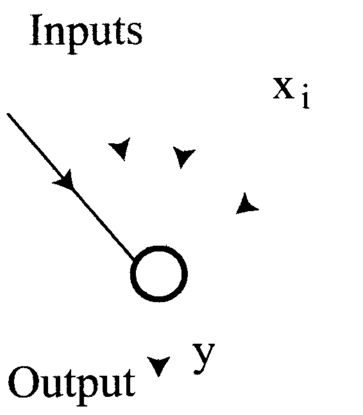

FIGURE 1. Structure of a single neuron.

The structure of the article is the following: in the next section we present the NN theory required for function fitting; afterward we obtain a PES for the H_3^+ molecule. This PES is then used to compute vibrational levels for the system, with a comparison with other theoretical results and experimental energy levels. The article is closed with some concluding remarks and perspectives for future work.

Function Approximation Using Neural Networks

A SINGLE NEURON

The basic unit of an NN is a single neuron as depicted in Figure 1. Each neuron receives a given number of inputs x_i, $i = 1, \ldots, n$. The output is computed in two steps: first the inputs are combined linearly with some prescribed weights:

$$z = \sum_{i=1}^{n} w_i x_i, \tag{1}$$

where z is the activation of the neuron and the w_i are also called the synaptic weights associated to the synaptic connections. In the second step the output is obtained as a function of the input by

$$y = \phi(z), \tag{2}$$

where ϕ is the activation function. Many choices are possible for ϕ as, for instance, the logistic function:

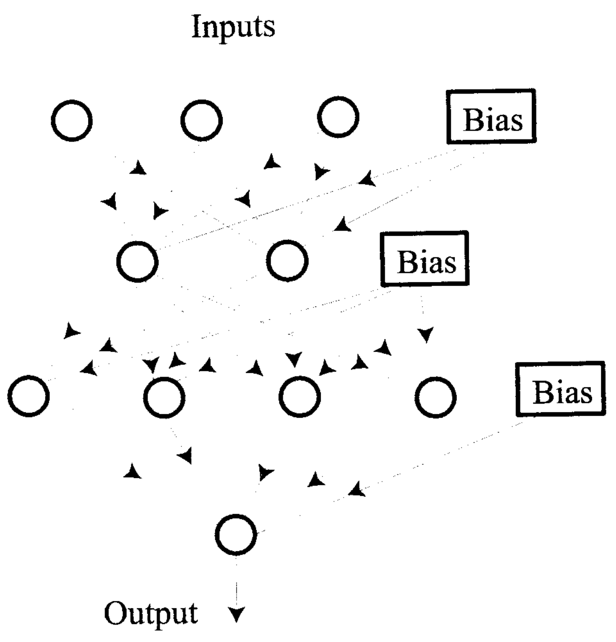

FIGURE 2. Feed-forward fully connected neural network.

$$\phi(z) = \frac{1}{1 + \exp(-z)}, \tag{3}$$

the hyperbolic-tangent function

$$\phi(z) = a \tanh(bz), \tag{4}$$

where a and b are adjustable parameters and the linear function

$$\phi(z) = z. \tag{5}$$

NETWORK STRUCTURE

An NN is formed using neurons connected such that some outputs are used as inputs of other neurons. The network then has a global set of inputs and outputs and synaptic connections. To each connection we associate a synaptic weight. For the specialized task of function approximation we are particularly interested in the feed-forward fully connected NN, illustrated in Figure 2. The network is structured in different layers. The first layer (from above, seen in Fig. 2) is formed by the input neurons such that the outputs are simply the NN inputs. The last layer is formed by the output neurons. The outputs of the NN are the outputs of the output neurons. The other layers are called hidden layers. Each layer except for the output layer has a bias that is equivalent to a neuron with unit output. Each connection has its own synaptic weight that is determined using a learning procedure, described below.

We now establish some notations: the output of the jth neuron on the kth layer is denoted by $y_j^{(k)}$, where $k = 0$ corresponds to the input layer, $k = 1$ to the first hidden layer, and so on up to the output layer with $k = N$. The activation of this neuron is denoted by $z_j^{(k)}$ and, hence,

$$y_j^{(k)} = \phi^{(k)}(z_j^{(k)}), \tag{6}$$

where $\phi^{(k)}$ is the activation function for the kth layer. Here all neurons in the same layer have the same activation function. Different levels can have different activation functions. The activation of the neuron is then given by:

$$z_j^{(k)} = \sum_{i=1}^{n_k} w_{ji}^{(k)} y_i^{(k-1)} + w_{j0}^{(k)}, \tag{7}$$

where $i = 0$ corresponds to the bias, n_k is the number of neurons in the kth layer, and $w_{ji}^{(k)}$ is the synaptic weight of the connection from the ith neuron of the $(k-1)$th layer to the jth neuron of the kth layer. Thus, the NN corresponds to the following functional form:

$$y_i^{(N)} = \phi^{(N)}\left(\sum_{i=1}^{n_N} w_{ji}^{(N)} \phi^{(N-1)}\left(\sum_{l=1}^{n_{N-1}} w_{il}^{(N-1)} \phi^{(N-2)}(\ldots) + w_{i0}^{(N-1)}\right) + w_{j0}^{(N)}\right). \tag{8}$$

This is the functional form used to fit data points to obtain a PES, and its importance is given by the following theorem [19]:

> For any continuous function $F(x_1, \ldots, x_n)$ on n variables, there exists a neural network of the form (8), with linear output activation function $\phi^{(N)}(z) = z$, such that
>
> $$|F(x_1, \ldots, x_n) - y_1^{(N)}| < \epsilon,$$
>
> for any arbitrarily small ϵ.

Our task is therefore to determine the NN structure and synaptic weights that best fit a given set of data points. This is the subject of the next subsection.

LEARNING METHODS

Now we address the question of how to determine the weights $w_{ji}^{(k)}$ such that the functional form in Eq. (8) gives a good approximation, within an allowed error, for a PES fitting the energies of different nuclear configurations. The nuclear configuration given by the coordinates Q_i, $i = 1, \ldots, m$, of the nucleus are used as the inputs $y_i^{(0)}$ of the NN and the corresponding electronic energy is obtained as the single output $y_1^{(N)}$. Let us suppose we have a number N_{ex} of input–output points, called the examples, denoted by $y_i^{(0)}(n)$ and $E(n)$, with $n = 1, \ldots, N_{ex}$. We want the NN to fit these points by minimizing the error function

$$\mathscr{E} = \sum_{n=1}^{N_{ex}} \mathscr{E}(n), \tag{9}$$

with

$$\mathscr{E}(n) = \frac{1}{2} \sum_{j=1}^{m} e_j^2(n), \tag{10}$$

and

$$e_j(n) = E(n) - y_1^{(N)}(n). \tag{11}$$

In this way the error $\mathscr{E}$ is a function of the weight parameters $w_{ji}^{(k)}$ and the fitting is equivalent to a function minimization in the weight space. One says that the NN learns the input–output relationship and can then be used to generalize, that is, to obtain the output for inputs not used in the previous process.

The most commonly used method is the back-propagation algorithm, which is a gradient-descent method in the weight space. The gradient of $\mathscr{E}$ is given by:

$$\frac{\partial \mathscr{E}}{\partial w_{ji}^{(k)}} = -\delta_j^{(k)} y_i^{(k-1)}, \tag{12}$$

with

$$\delta_j^{(k)} = \phi'^{(k)}(z_j^{(k)}) \sum_{i=1}^{n_{k+1}} \delta_i^{(k+1)} w_{ij}^{(k+1)}; \quad k \neq N,$$

$$\delta_j^{(N)} = e_j \phi'^{(N)}(z_j^{(N)}), \tag{13}$$

where $\phi'^{(k)}$ is the derivative of the activation function. The back-propagation prescription is done in two steps for each training example: a forward pass where the activation $z_i^{(k)}$ and output $y_i^{(k)}$ of each neuron is computed starting from the input layer ($k = 0$) up to the output layer ($k = N$). The second step is a backward pass where the gradients in Eq. (12) are computed from the output to the input layers. The corrections of the synaptic weights in each cycle is:

$$w_{ji}^{(k)}(l+1) = w_{ji}^{k}(l) + \eta \delta_j^{(k)} y_i^{(k-1)}, \tag{14}$$

where η is the learning parameter (typically between 0 and 1) and controls the rate at which we descend to the minimum in the weight space and l counts the number of cycles performed. In Eq. (14) each example is used in turn to correct the weights. A complete cycle with all the examples being used is called an epoch. The network is initialized with random weights, and then the repeated application of Eq. (14) is used to minimize the error. This prescription can be slightly modified by

$$w_{ji}^{(k)}(l+1) = w_{ji}^{k}(l) + \alpha \Delta w_{ji}^{k}(l-1) + \eta \delta_j^{(k)} y_i^{(k-1)}, \tag{15}$$

where $\Delta w_{ji}^{k}(l-1)$ is the weight correction when the same example was used in the previous epoch. The new term in Eq. (15) helps to avoid oscillations in $\mathscr{E}$ during the gradient descent, and α is called the momentum constant.

The main drawback of the back-propagation algorithm is that it gels exceedingly slow near the minimum. Also, to ensure convergence, the parameters α and η must take increasingly smaller values as they approach the minimum. This makes the method too time-consuming if great accuracy is required, as is the case in the current article. Other learning methods are available in the literature [19]. We used an efficient alternative namely, the conjugate-gradient method, which we summarize now [19, 20]. Near the minimum, the error function can be written as

$$\mathscr{E} = c + \mathbf{b}^T \cdot \mathbf{w} + \frac{1}{2} \mathbf{w}^T \cdot \mathbf{A} \cdot \mathbf{w} + \ldots, \tag{16}$$

where the dots stand for cubic- and higher-order terms and $\mathbf{w}$ is a vector with the weights $w_{ji}^{(k)}$ as components. Two vectors $\mathbf{s}(i)$ and $\mathbf{s}(j)$ are said to be A-conjugate if

$$\mathbf{s}(i)^T \cdot \mathbf{A} \cdot \mathbf{s}(j) = 0. \qquad (17)$$

The conjugate-gradient method then works to generate a sequence of A-conjugate vectors and minimize the error along each of the directions given by these vectors. We start at a point $\mathbf{w}(0)$ and take as the initial direction $\mathbf{s}(0) = -\partial\mathscr{E}/\partial\mathbf{w}(0)$ computed using Eq. (12). The algorithm is given by the following steps:

1. Minimize $\mathscr{E}$ along the direction $\mathbf{s}(i)$ starting from $\mathbf{w}(i)$ and set $\mathbf{w}(i + 1)$ to this minimum.
2. Compute the new direction vector by

   $$\mathbf{s}(i+1) = \mathbf{r}(i+1) + \beta(i+1)\mathbf{s}(i),$$

   where $\mathbf{r}(i + 1)$ is

   $$\mathbf{r}(i+1) = -\frac{\partial\mathscr{E}}{\partial\mathbf{w}(i+1)},$$

   and $\beta(i + 1)$ is given by the Polak–Ribière formula [19]:

   $$\beta(i+1) = \max\left(\frac{\mathbf{r}^T(i+1)\cdot(\mathbf{r}(i+1) - \mathbf{r}(i))}{\mathbf{r}^T(i)\cdot\mathbf{r}(i)}, 0\right).$$
3. Verify for the stopping condition and go back to step 1.

As stopping condition we require that $\mathscr{E}$ is lower as a prescribed value. The main advantage of the conjugate-gradient method is that for a purely quadratic function it is guaranteed to converge to the minimum in at most n iterations for a function in n variables. In our applications, we start the learning by doing a few back-propagation cycles to avoid straying too far from the minimum and then proceed with the conjugate-gradient method.

Before closing this section we discuss an important issue in fitting functions from data points. In the iterative process of minimizing $\mathscr{E}$, the functional form of the NN gets closer to the points used in the learning. The same is not necessarily true for other points. To ensure a good overall fit, we separate the data set in two disjoint sets: an example set used to train the network, and a test set used to verify the quality of the generalization of the fit. When the learning begins, the error associated with both sets starts diminishing, but after a certain number of epochs the error for the test set starts growing while the error for the example set continues to converge to its minimum. The learning must be stopped at this point, after which we say that the network is overfitted. The same type of problem is also present in fitting any other type of functional forms.

Potential Energy Surface for H_3^+

For obtaining a PES for the H_3^+ molecule we used the 69 ab initio points in Ref. [2], converting from the Morse-type coordinate to internuclear distances. Because we have only a relatively small number of points, it is difficult to separate some of them to use as test data. To avoid this problem we use the multifold cross-validation, where we take only a few points as test patterns and train the NN and then consider all (or a significant part) partitions of the points in example and test sets. Then, looking for overfitting in each case allows determination of an optimal stopping point for learning for all the points. In our case we considered in each partition an example set of 66 points and a test set of 3 points. This results in 23 different partitions in such a way that all points are used at least once as test and example data.

For the structure of the net, a hidden two-layer NN has better convergence. This can be explained heuristically by noting that the first hidden layer reveals global features of the PES, whereas the second layer reveals local details. The number of neurons in each hidden layer was determined by experimenting with different configurations and considering the lowest possible error in a reasonable CPU time without overfitting. The best structure we obtained has 3 input neurons, 12 neurons in the first hidden layer, 3 neurons in the second hidden layer, and one output neuron, with a total of 91 weights. The activation of the output neuron is the linear function in Eq. (5). For the hidden layers we use a hyperbolic tangent function in Eq. (4) with $a = 1.7159$ and $b = 2/3$ [19].

The permutation symmetry of the molecule can be considered in a number of ways. The NN can reveal the original 69 points plus all different permutations of the internuclear distances, which amounts to 208 points, with the corresponding increase in the CPU time required to reveal the extra examples. In Ref. [4] a symmetric NN was introduced in such a way that the final functional form is already in the symmetrized form, without having to artificially introduce extra data in the examples. Here we consider a simpler alternative by symmetrizing the input space, that is, by rearranging the points such that the NN reveals the examples with

TABLE I
Synaptic weights for the H_3^+ PES.

k	j	i	$w_{ji}^{(k)}$	k	j	i	$w_{ji}^{(k)}$	k	j	i	$w_{ji}^{(k)}$
1	1	0	0.375510532	1	1	1	−1.70043409	1	1	2	0.455163557
1	1	3	−0.793956725	1	2	0	0.303926169	1	2	1	−0.157734918
1	2	2	−2.12419247	1	2	3	−0.0538461535	1	3	0	0.0226537177
1	3	1	0.826187203	1	3	2	0.587865257	1	3	3	0.627611882
1	4	0	−0.244220911	1	4	1	0.791588196	1	4	2	0.729209407
1	4	3	0.818622876	1	5	0	−0.15761748	1	5	1	−0.212018197
1	5	2	0.326923088	1	5	3	−0.733097008	1	6	0	0.0503493976
1	6	1	0.896491076	1	6	2	0.333824979	1	6	3	−0.0524267651
1	7	0	−0.229645401	1	7	1	2.15296563	1	7	2	−0.0814328207
1	7	3	0.134303161	1	8	0	−0.371938632	1	8	1	0.00549235323
1	8	2	−0.246447306	1	8	3	−0.229665144	1	9	0	−0.255905757
1	9	1	0.276417328	1	9	2	−0.204703445	1	9	3	−1.45691204
1	10	0	−0.09581898	1	10	1	−0.673445085	1	10	2	−0.111535606
1	10	3	0.182339981	1	11	0	0.15211796	1	11	1	0.117505871
1	11	2	0.469840601	1	11	3	1.54250597	1	12	0	0.619813374
1	12	1	−0.992249303	1	12	2	0.279952426	1	12	3	−1.9027655
2	1	0	−0.38040978	2	1	1	−0.0866025603	2	1	2	−0.617434174
2	1	3	−0.408066751	2	1	4	0.263874675	2	1	5	0.0768341751
2	1	6	0.532070302	2	1	7	0.900838818	2	1	8	0.707992089
2	1	9	0.360217565	2	1	10	0.00935472058	2	1	11	−0.432323118
2	1	12	−0.328723798	2	2	0	−0.001614275	2	2	1	0.904230576
2	2	2	0.439883386	2	2	3	−0.293905037	2	2	4	−0.547526572
2	2	5	0.207919775	2	2	6	−0.568323183	2	2	7	−0.890011564
2	2	8	−0.167335093	2	2	9	−0.99983859	2	2	10	0.64372784
2	2	11	0.858283376	2	2	12	0.818341429	2	3	0	0.289572135
2	3	1	0.338081277	2	3	2	0.772004138	2	3	3	0.272043377
2	3	4	0.281829982	2	3	5	−0.0180026416	2	3	6	0.127608918
2	3	7	1.06702826	2	3	8	0.0710791893	2	3	9	0.772117602
2	3	10	−0.380046919	2	3	11	−0.434865275	2	3	12	0.600470984
3	1	0	0.46885215	3	1	1	−1.05029002	3	1	2	−0.279254423
3	1	3	0.890284642								

increasing internuclear distances for each example. Later, when computing the output of the NN, the input internuclear distance should be put in the same order. This approach allows one to use the same computer code for systems with and without specific symmetries.

For the structure used we obtained as final error $\mathscr{E} = 8.3 \times 10^{-11}$ after some 200,000 cycles of the conjugate-gradient algorithm. Each cycle took 0.04 s of CPU time on a Pentium III 700-MHz computer. The weights for the PES are given in Table I. The error obtained must be compared with the error $\mathscr{E} = 3.7 \times 10^{-12}$ obtained using a 10th-order polynomial in Morse-type coordinates (here we used the same definition for $\mathscr{E}$ for each PES). Because the polynomial fit is obtained by linear algorithm, the CPU time it requires is much less than that required for the NN fit, and the error is one order of magnitude lower. Nevertheless, a decisive test is to use each PES to compute an observable quantity that is very sensitive to the details of the PES. We chose to compute vibrational levels and compare the results to experimental results. This is the subject of next section.

Vibrational Levels

For a comparison of the fit using an NN, we computed the first low-lying vibrational levels for vanishing total angular momentum ($J = 0$), using the discrete variable representation (DVR) program TRIATOM of Tennyson and Miller [21] and a finite-

TABLE II
Vibrational levels (in cm^{-1}) for the H_3^+ molecule for the NN fit and 10th degree polynomial fit (POL) of Ref. [14].

Level	NN–DVR	NN–FE	POL–DVR	POL–FE	Exp.
1	2521.397	2520.924	2521.528	2521.998	2521.409
2	3176.439	3175.697	3178.402	3178.142	3178.29
3	4780.626	4779.203	4776.517	4774.495	
4	4995.994	4994.768	4997.585	4994.627	4997.965
5	5550.930	5549.332	5560.995	555.044	5554.063

element (FE) method in hyperspherical coordinates [22]. The results are shown in Table II. The experimental results are taken from Refs. [14], [23], and [24]. The values of the energy levels depend slightly on the method used. For the ground state the best agreement with the experimental value is obtained using the DVR method with the neural network fit. Both fits result in a good agreement with the experimental data, usually with an error of the order of 2 cm^{-1}.

Concluding Remarks and Perspectives

The NN and polynomial fit yield good results for the vibrational energy level of the H_3^+ molecule. The error associated with the polynomial fit is one order of magnitude smaller than for the NN fit. Also, the CPU time required for a polynomial fit is much smaller than to train an NN. Therefore, for a smooth PES, using a polynomial fit seems to be a better choice than to use the more computer-demanding NN fit. Nevertheless, it is not an objective of the current article to obtain a better PES for the H_3^+ system than that available in the literature but to show that our approach allows one to obtain high-quality PES from ab initio data. This is required for describing more complicated structures in a PES that would be difficult to be fitted by a simple polynomial function. The main advantage of an NN is its ability in approximating complicated functions as stated by the approximation theorem for NNs. An improvement of this approach can be obtained by considering preliminary information on the PES structure. For instance, in the many-body expansion method [1], the PES is written as a sum of two- and three-body terms (for a triatomic molecule). Each term is then fitted with a functional form. This ensures a correct asymptotic behavior for the PES. Fitting each many-body term with an NN results in a better NN fit than simply fitting the whole set of ab initio points using a single NN. Preliminary work in this direction is under way and will be the subject of a forthcoming publication.

ACKNOWLEDGMENTS

One of the authors (Z.T.O.J.) is supported by and two of the authors (T.M.R.F. and J.J.S.N.) are partially supported by CNPq (a Brazilian government agency).

References

1. Murell, J. N.; Carter, S.; Farantos, S. C.; Huxley, P.; Varandas, A. J. C. Molecular Potential Energy Functions; Wiley, Chichester, 1984.
2. Cencek, W.; Rychlewski, J.; Jaquet, R.; Kutzelnigg, W. J Chem Phys 1998, 108, 2831.
3. Garcia, E.; Laganà, A. Mol Phys 1973, 56, 3229.
4. Prudente, F. V.; Acioli, P. H.; Soares Neto, J. J. J Chem Phys 1998, 109, 8801.
5. Prudente, F. V.; Soares Neto, J. J. Chem Phys Lett 1998, 287, 585.
6. Blank, T. B.; Brown, S. D.; Calhoun, A. W.; Doren, D. J. J Chem Phys 1995, 103, 4129.
7. Brown, D. F. R.; Gibbs, M. N.; Clary, D. C. J Chem Phys 1996, 105, 7597.
8. Bettens, R. P. A.; Collins, M. A. J Chem Phys 1999, 111, 816.
9. No, K. T.; Chang, B. H.; Kim, S. Y.; Jhon, M. S.; Scheraga, H. A. Chem Phys Lett 1997, 271, 152.
10. Magela e Silva, G.; Acioli, P. H.; Pedroza, A. C. J Comp Chem 1997, 18, 1407.
11. Sumpter, B. G.; Noid, D. W. Chem Phys Lett 1992, 192, 455.
12. Geremia, J. M.; Rabitz, H.; Rosenthal, C. J Chem Phys 2001, 114, 9325.
13. Duck, W.; Diercksen, G. H. F. Comp Phys Comm 1994, 82, 91.
14. Jaquet, R.; Cencek, W.; Kutzelnigg, W.; Rychlewski, J. J Chem Phys 1998, 108, 2837.
15. Polyansky, O. L.; Tennyson, J. J Chem Phys 1999, 110, 5056.
16. Dykstra, C. E.; Swope, W. C. J Chem Phys 1979, 70, 1.
17. Meyer, W.; Botschwina, P.; Burton, P. J Chem Phys 1986, 84, 891.

18. Sutcliffe, B. T.; Tennyson, J. Int J Quantum Chem 1991, 29, 183.

19. Zupan, J.; Gasteiger, J. Neural Networks for Chemists; VHC: Weinheim, 1993.

20. Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; Flannery, P. Numerical Recipes, 2nd edition, Cambridge University Press (New York, 1992).

21. Tennyson, J.; Miller, S. Comp Phys Comm 1989, 55, 149.

22. Soares Neto, J. J.; Linderberg, J. J Comp Chem 1991, 12, 1237.

23. Dinelli, B. M.; Neale, L.; Polyansky, O. L.; Tennyson, J. J Mol Spectrosc 1997, 181, 142.

24. Kao, L.; Oka, T.; Miller, S.; Tennyson, J. Astrophys J Suppl Ser 1991, 77, 317.