Using Simply Contracted Basis Functions with the Lanczos Algorithm to Calculate Vibrational Spectra

XIAO-GANG WANG, TUCKER CARRINGTON JR.
Département de chimie, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal (Québec) H3C 3J7, Canada

Received 11 December 2002; accepted 10 October 2003
Published online 29 December 2003 in Wiley InterScience (www.interscience.wiley.com).
DOI 10.1002/qua.10847

ABSTRACT: We demonstrate that the combination of simply contracted basis functions and the Lanczos algorithm yields an extremely efficient method for computing vibrational energy levels. We discuss ideas and present some results for HOOH and CH₄. The basis functions we use are products of eigenfunctions of reduced-dimension Hamiltonians obtained by freezing coordinates at equilibrium. The basis functions represent the desired wave functions well yet are simple enough that matrix–vector products may be evaluated efficiently. © 2003 Wiley Periodicals, Inc. Int J Quantum Chem 99: 556–566, 2003

Key words: contracted basis; Lanczos algorithm; methane; vibrational spectroscopy

Introduction

The development of good methods for calculating vibrational and rovibrational spectra of small molecules is a longstanding goal of theoretical chemistry [1–7]. The most accurate theoretical spectra are determined by computing eigenvalues and eigenvectors of a matrix representation of the Hamiltonian. If one uses a simple product basis the size of the matrix for which one must calculate eigenvalues and eigenvectors in n°, where n represents the number of basis functions in each degree of freedom and f is the number of degrees of freedom. For a triatomic molecule, with only three vibrational degrees of freedom, it is now straightforward to calculate vibrational levels. The size of the product basis matrix is smaller than about 10000; it can be easily stored in memory and its eigenvalues can be computed using standard Householder transformations. A four-atom molecule has six vibrational degrees of freedom. The product basis matrix is so large that it cannot be stored in memory. It is nevertheless possible to calculate vibrational spectra of molecules with four atoms by using one of two established strategies; (1) contracted basis functions obtained by diagonalizing reduced-dimension Hamiltonians and the usual diagonalization method [4, 8, 9]; (2) simple product basis
functions and an iterative eigensolver (usually the Lanczos [10] or the filter diagonalization method) [11–24]. Using an iterative method obviates the need to store the Hamiltonian matrix. Both these strategies are good enough to enable one to compute vibrational levels of a four-atom molecule. However, it is difficult to use either strategy to compute vibrational levels of a five-atom molecule.

The two strategies are successful for different reasons. The success of the contracted-direct strategy is due to the small size of the contracted basis used to compute the spectrum. With a product basis one often needs orders of magnitude more basis functions to compute the same spectrum. The success of the product-iterative strategy is due to two factors: First, it obviates the need to store the Hamiltonian matrix; second, the product basis makes it possible to evaluate matrix–vector products, and hence compute eigenvalues, efficiently. It is enticing to consider the possibility of using contracted basis functions with an iterative eigensolver: It might be possible to exploit both the smallness of the contracted basis and the efficiency of the iterative eigensolver [16, 25–27].

Contracted-iterative methods are attractive. It is natural to anticipate that they will be more efficient and require less memory than their product-iterative counterparts. The basis functions used in most previous attempts to combine a contacted basis and an iterative method are adiabatic contracted functions [16, 25, 26]. For a triatomic molecule an adiabatic basis function is obtained by diagonalizing (stretch) blocks \( \langle \alpha | H | \beta \rangle \) \( \alpha \) to get eigenfunctions \( \phi^\alpha_n(r_1, r_2) \), where \( \alpha(\theta) \) is a discrete variable representation (DVR) bend function, \( H(\theta, r_1, r_2) \) is the full Hamiltonian operator, and \( \theta, r_1, r_2 \) are the bend and stretch coordinates. The adiabatic contracted basis functions are \( \alpha(\theta)\phi^\alpha_n(r_1, r_2) \). In this article we use different contracted functions. The contracted functions we use, denoted “simply contracted functions” in Ref. [16], are eigenfunctions of reduced-dimension Hamiltonians obtained by fixing the values of some coordinates. For many molecules one will need more simply contracted functions than one would need if one were using adiabatic contracted functions but using simply contracted functions enables one to evaluate matrix–vector products, required to compute eigenvalues with an iterative method, efficiently. We divide the vibrational degrees of freedom of a molecule into two groups: bend coordinates and stretch coordinates. For both groups we calculate eigenfunctions. Products of the bend and stretch functions are used as basis functions for calculating energy levels of the whole molecule [28]. Recently, Yu suggested using a basis of products of bend eigenfunctions and DVR functions for the stretches [29]. Rather than using DVR functions for the stretches we use eigenfunctions of a stretch Hamiltonian.

### Bend Stretch Contraction Scheme

As vibrational coordinates we use, for an \( N \) atom molecule, the lengths of \( N - 1 \) vectors, \( N - 2 \) planar angles, and \( N - 3 \) dihedral angles [30]. The kinetic energy operator (KEO) in these coordinates is surprisingly simple. We use “orthogonal” vectors so that there is no momentum coupling between stretch and bend coordinates [31–33]. We exploit this advantage of orthogonal coordinates to design our contraction scheme. If the vectors are orthogonal then in these coordinates the \( J = 0 \) KEO is

\[
H = T_{\text{ben}} (\theta, r) + T_{\text{str}} (r) + V(\theta, r)
\]

with

\[
T_{\text{ben}} (\theta, r) = \sum_i G^{\|}_{\text{ben}} (r) T^\theta_i (\theta) \\
T_{\text{str}} (r) = \sum_i G^{\|}_{\text{str}} \frac{\partial^2}{\partial r_i^2}.
\]

\( \theta \) represents all of the bend coordinates and \( r \) represents all of the stretch coordinates. The functions \( G^{\|}_{\text{ben}} (r) \), \( G^{\|}_{\text{str}} \), and the operators \( T^\theta_i (\theta) \) are known [31–33].

The reduced-dimension Hamiltonian for the bend contraction is

\[
H^{\text{ben}} = T_{\text{ben}} (\theta, r_\perp) + V(\theta, r_\perp).
\]

Its wave functions are denoted by

\[
X_n (\theta) = \sum_j C_{n} f_j (\theta)
\]

and the energies by \( E_n \). The \( f_j \) are primitive finite basis representation (FBR) bend basis functions (if there is more than one bend coordinate \( l \) is a composite index) and the number of retained bend wave functions is denoted by \( n_\perp \). When using the coordinates described above it is natural to use spherical harmonics as bend basis functions [32,
and its wave functions are denoted by

\[ Y_s(r) = \sum_a D_{as} g_a(r) \]  

(6)

and the energies by \( E_s \). The \( g_a \) are primitive DVR [35] stretch basis functions (if there is more than one stretch coordinate \( \alpha \) is a composite index representing a multidimensional DVR function) and the number of retained stretch wave functions is denoted by \( n_s \). Our contraction scheme relies on the use of DVR primitive stretch basis functions. We use potential optimized DVRs (PODVRs) [36, 37]. \( \theta_e \) and \( r_e \) represent equilibrium values of (respectively) all the bend coordinates and all the stretch coordinates.

The final basis is a product of the retained stretch and bend eigenfunctions:

\[ |bs\rangle = |X_b\rangle|Y_s\rangle. \]

(7)

Note that our contracted functions are direct products of functions of different coordinates or groups or coordinates [8, 38]. Of course, only the eigenvectors of the reduced-dimension Hamiltonian matrices whose eigenvalues are below a given cut-off energy are retained.

For many molecules, coupling between the contracted basis functions we use is small enough that the basis functions can be used to label the wave functions. Even when stretch–bend coupling is fairly important (e.g., methane) the zeroth-order labels appear to be good.

**MATRIX–VECTOR PRODUCTS WITH A SIMPLY CONTRACTED BASIS**

To facilitate evaluation of matrix–vector products we write the full Hamiltonian

\[ H = H^{(b)} + H^{(s)} + \Delta T + \Delta V, \]

(8)

where

\[ \Delta V(r, \theta) = V(r, \theta) - V(r_s, \theta) - V(r, \theta_e) \]

(9)

\[ \Delta T = \sum_i \Delta G^{(i)}(r)T^{(i)}(\theta) \]

(10)

with

\[ \Delta G^{(i)}(r) = G^{(i)}(r) - G^{(i)}(r_e). \]

(11)

Because \( H^{(b)} \) and \( H^{(s)} \) are diagonal in the product contracted basis and the diagonal matrix elements are known (they are eigenvalues of \( H^{(b)} \) and \( H^{(s)} \)) one only needs to calculate matrix elements of \( \Delta T \) and \( \Delta V \) in the product contracted basis.

Matrix–vector products for \( \Delta T \) are easy. Due to the lack of momentum cross-terms coupling bend and stretch coordinates, we only have to deal with the coupling term of Eq. (10). This term is diagonal in the stretch PODVR basis [note that there are no derivatives with respect to stretch coordinates in Eq. (10)]. The product structure of Eq. (10) can be exploited by storing the bend and stretch matrices separately. The matrix–vector product for \( \Delta T \) can be performed sequentially without summing over \( b \) and \( s \) labels simultaneously. Note that it not necessary to store any vectors in the primitive basis.

Matrix–vector products for \( \Delta V \) are more difficult. A matrix element of \( \Delta V \) in the product contracted basis is

\[ \langle b's'|\Delta V(\theta, r)|bs\rangle \]

\[ = \sum \sum C_{i'j'}C_{ib}D_{ai}D_{as}\langle l'|\alpha|\Delta V(\theta, r)|\alpha\rangle \]

\[ = \sum \sum C_{i'j'}C_{ib}D_{ai}D_{as}T_{i'l'}\Delta V_{\beta\alpha} \]

(12)

where

\[ T_{i'l'} = \sqrt{w_{f_j}(\theta_{i'})} \]

\[ \Delta V_{\beta\alpha} = \Delta V(\theta_{\beta}, r_{\alpha}). \]

(13)

\( (\theta_{\beta}, w_{\beta}) \) are quadrature points and weights for the angles. Note that if there is more than one bend coordinate \( l \) and \( \beta \) are composite indices and if there is more than one stretch coordinate \( \alpha \) is a composite index. It is sometimes advantageous to rewrite Eq. (12) as
\[ \langle b's'\mid \Delta V(\theta, r)\rangle_{bs} = \sum_{\alpha \beta} C_{\beta \alpha} D_{\alpha s} D_{\alpha s} \Delta V_{\beta \alpha} \]  

(14)

where

\[ C_{\beta \alpha} = \sum_{l} C_{\beta l} T_{l \alpha}. \]  

(15)

In the standard approach, the \( \Delta V \) matrix–vector product is evaluated as

\[ u'_{b s} = \sum_{b s} \langle b's'\mid \Delta V(\theta, r)\rangle_{bs} u_{s b} = \sum_{\alpha} D_{\alpha s} \sum_{s} C_{\beta \alpha} \Delta V_{\beta s} \sum_{b s} C_{\beta s} u_{s b}. \]  

(16)

The matrix–vector product can be done more efficiently by introducing an \( F \) matrix defined by

\[ F_{\beta \alpha} = \sum_{\beta} C_{\beta \alpha} \Delta V_{\beta \alpha} \]  

in terms of which

\[ \langle b's'\mid \Delta V(\theta, r)\rangle_{bs} = \sum_{\alpha} F_{\alpha \beta} D_{\alpha s} D_{\alpha s}. \]  

(18)

To use our method one calculates and stores all the \( F_{\beta \alpha} \). One then does the \( \Delta V \) matrix–vector product in three steps:

\[ u'_{b s} = \sum_{s} D_{\alpha s} u_{s b} \]

\[ u^{(2)}_{b s} = \sum_{\alpha} F_{\alpha \beta} u^{(1)}_{s b} \]

\[ u'_{b s} = \sum_{\alpha} D_{\alpha s} u^{(2)}_{s b}. \]  

(19)

The CPU cost of the matrix–vector product in Eq. (19) is \( n_{a} (n_{s}^{2} + 2n_{s}n_{d}) \). Each matrix–vector product is therefore cheap. The cost is much less than the cost of a product basis matrix–vector product \( fn^{2} + 1 \_d \) where \( n_{d} \) is the number of basis functions and quadrature points for each coordinate [15].

Of course, this comparison is not really fair; to estimate the total CPU cost of the calculation, one needs to also include the cost of calculating \( F \). The \( F \) matrix is computed prior to and not during the Lanczos recursion. If one evaluates Eq. (17) as it is written the cost of computing the matrix scales as \( n_{a} n_{s}^{2} n_{d} \). In practice, we use level-3 Basic Linear Algebra Subprograms (BLAS) for each \( \alpha \) in Eq. (17) by storing \( \Delta V \) first. This speeds the calculation of \( F \) by about a factor of 5 for \( \text{H}_2\text{O}_2 \). See Table II in which we present CPU times for different parts of the calculation.

In Table II below we compute the total time by adding the time required to calculate \( F \) and the time required to evaluate the matrix–vector products, but on a parallel computer one could easily reduce the total time by parallelizing the calculation of \( F \). The calculation of \( F \) can be trivially parallelized as one can allocate the stretch DVR \( (\alpha) \) points to different computers or processors and allow each to calculate a block of \( F \) for the allocated \( \alpha \) points. The full \( F \) is then assembled from these blocks.

Now, we estimate the memory one needs to use this method. Once \( F \) has been calculated the simplest way to use Eq. (19) to calculate matrix–vector products requires storing \( F \) and \( D \). The memory cost is dominated by the cost of storing \( F \) and scales as \( n_{a} n_{s}^{2} / 2 \). If it is not possible to store \( n_{a} n_{s}^{2} / 2 \) numbers the method can nonetheless be used by storing \( F \) on disk and reading blocks of \( F \) labeled by \( \alpha \) sequentially from the disk into memory as the \( \Delta V \) matrix–vector product is computed. Note in addition that because \( C \) and \( \tilde{C} \) are discarded before matrix–vector products are evaluated one can use an extremely large bend basis to converge the contracted bend functions without increasing the memory cost of the final eigenvalue calculation (the CPU cost of the calculation of \( F \) does increase moderately). In contrast, if one uses the standard approach [Eq. (16)] increasing the size of the bending basis increases both the CPU and memory cost dramatically.

### Calculating the Bend and Stretch Contracted Functions

The bend and stretch eigenfunctions are computed using the Cullum and Willoughby (CW) Lanczos algorithm [39]. It is now common to use CW Lanczos to calculate energy levels but it is still rare to use it to obtain wave functions (eigenvectors). Using the Lanczos algorithm to calculate eigenvectors is discussed in detail by Cullum and Willoughby [39]. First, for all eigenvalues for which one wishes to compute an eigenvector one determines the number of Lanczos vectors one should use to compute the eigenvector. Hereafter, this is...
called the iteration depth. For each eigenvalue it is determined after the first Lanczos calculation. Second, during a second Lanczos calculation the desired eigenvectors are computed from the predetermined iteration depths. How does one determine the iteration depths? CW used a Sturm sequence method to estimate the number of iterations \((M_i)\) required to converge an eigenvalue and the number of iterations \((M_2)\) required to copy an eigenvalue. As their iteration depth they choose 1 of 10 values between 2 linear combinations of \(M_1\) and \(M_2\). The residual error of the calculated eigenvector can be estimated by \(t_i^{M} = |z_{M_i}|\beta_{M+1}\), where \(z_{M_i}\) is the \(M\)th (last) component of the \(i\)th eigenvector of the \(M \times M\) tridiagonal matrix, \(\beta_{M+1}\) is the subdiagonal element between row \(M\) and column \(M + 1\) of the tridiagonal matrix, and \(i\) is an eigenvalue whose eigenvector is sought. We use a different method to obtain the iteration depth. For each eigenvalue, we calculate \(t_i^{M}\) using inverse iteration for \(M = M_1, M_1 + 1, \ldots\). The iteration depth is set equal to the first value of \(M\) for which \(t_i^{M}\) is smaller than \(10^{-7}\) cm\(^{-1}\). We find that the iteration depth is always between \(M_1\) and \(M_2\). Starting at \(M_1\) and increasing \(M\) guarantees that we will find the smallest value of \(M\) for which \(t_i^{M}\) is smaller than \(10^{-7}\) cm\(^{-1}\). In general, there are other (larger) values of \(M\) for which \(t_i^{M}\) is smaller than \(10^{-7}\) cm\(^{-1}\) but we want the smallest one because this decreases the number of matrix–vector products required to compute the eigenvectors (in the second Lanczos iteration). The cost of determining the iteration depth is small because the cost of each inverse iteration calculation scales as \(M\). It is true that the cost becomes larger as \(M\) increases, but searching systematically by starting at \(M_1\) is nonetheless advantageous because it gives us the minimum iteration depth for which \(t_i^{M}\) is smaller than \(10^{-7}\) cm\(^{-1}\).

Note that one could avoid the need to determine the correct iteration depth by storing and orthogonalizing Lanczos vectors. This, however, would require a large amount of memory. If a spectral transform is used one can reduce the number of vectors but we prefer, following CW, to obtain eigenvectors without storing any Lanczos vectors.

The method we propose in this article requires that DVR basis functions be used to solve the stretch problem. Usually, a direct product of PODVR basis functions, built from large 1-D DVR basis sets, is used. This basis can be reduced by discarding the PODVR points whose potential energy is larger than a specified cut-off energy. This saves both memory and CPU time because both the \(F\) matrix and stretch wave functions (\(D\) matrix) depend on the size of PODVR basis. Discarding points destroys the direct product nature of the stretch grid. This could affect both the calculation of the contracted stretch functions and calculation of the final levels in the contracted stretch bend basis. In our contracted iterative method we do not exploit the direct product nature of the stretch grid when computing matrix–vector products in the contracted basis [in the final Lanczos calculation [see Eq. (19)]]). Therefore, at this stage nothing is lost by giving up the direct product structure of the grid. We do use the direct product character of the grid when computing the contracted stretch functions. Giving up the direct product structure of the grid does make doing the matrix–vector products to solve the stretch problem more complicated. The simplest way to compute a matrix–vector product \(u' = Hu\) is to use a direct-product vector for \(u\), set the discarded components to zero, and compute only the components of \(u'\) that are not discarded. It would be more efficient but more complicated to use constrained indices [40]. By discarding 9D DVR functions Yu made a product DVR calculation of vibrational levels of methane [41].

When computing both the stretch and bend functions we use the symmetry-adapted Lanczos (SAL) algorithm [42, 43]. For both molecules discussed below the SAL is used within parity blocks for the bend functions.

Because the bend problem is much more demanding than the stretch problem, it is desirable to find a means of reducing its cost. Our bend quadrature grid is a direct product grid. It is possible, after computing contracted basis functions, to save CPU time by discarding quadrature points for which all the contracted functions are nearly zero. We call this compaction [16, 26]. The compaction ratio, defined as the ratio of the number of retained quadrature points to the number of direct product quadrature points, will decrease quickly as the dimension of the system is increased. Rather than computing contracted functions first and then compacting, one can compact them as they are computed. To do so one discards quadrature points for which the potential is larger than a specified cut-off energy. We compute compacted Lanczos vectors and then assemble them to obtain compacted contracted functions. For the cut-off energy a good criterion is the norm of a residual vector defined as the vector formed from the discarded points. The sum of the square of the norm of the residual vector plus the square of the norm of the compacted wave function

\[560\]
is 1. If the norm of this residual vector is small the cut-off energy is appropriate.

### Application to H$_2$O$_2$

We first applied this method to H$_2$O$_2$. There is considerable interest in H$_2$O$_2$ [44]. Good potential energy surfaces are available and H$_2$O$_2$ was one of the first four-atom molecules for which numerically exact calculations were attempted [15, 38, 45–52]. In this section we demonstrate the efficiency of our methods by computing vibrational levels of H$_2$O$_2$.

#### SYMMETRY-ADAPTED BASIS AND HAMILTONIAN MATRIX

For H$_2$O$_2$ we use diatom–diatom Jacobi coordinates. The kinetic energy operator (KEO) is well known [15, 33]. We use parity-adapted basis functions that are linear combinations of the functions

\[ |l_1 l_2 m_2 \rangle = \Theta_l^m(\pi - \theta_1) Y_{l_2}^m(\theta_2, \phi_2) \]
\[ = (-1)^{l_1+m_2} \Theta_l^m(\theta_1) Y_{l_2}^m(\theta_2, \phi_2), \]  \hspace{1cm} (20)

with

\[ Y_{l_2}^m(\theta_2, \phi_2) = \frac{1}{\sqrt{2\pi}} \Theta_l^m(\theta_2) e^{im\phi_2}, \]  \hspace{1cm} (21)

where \( \Theta_l^m(\theta) \) is a normalized associated Legendre function with the \((-1)^m\) Condon–Shortley phase factor. The angles are defined in Ref. [28].

The parity-adapted functions are

\[ \mu_{l_1 l_2 m_2}^P = \frac{1}{\sqrt{2(1 + \delta_{m_2,0})}} [ |l_1 l_2 m_2 \rangle + (-1)^P |l_1 l_2 m_2 \rangle ] \]
\[ (m_2 \approx 0), \]  \hspace{1cm} (22)

where \( P = 0 \) and \( 1 \) correspond to even and odd parity, respectively. Note that if \( P = -1 \) and \( m_2 = 0 \) then there is no basis function.

We write the KEO in terms of vibrational angular momentum operators. Matrix elements of the KEO in the parity-adapted bend basis are given in Ref. [28]. To calculate potential matrix elements we use Gauss–Legendre quadrature for the two planar angles and Gauss–Chebyshev quadrature of the first kind for the dihedral angle. It is possible to reduce the integration range of the dihedral angle to \([0, \pi]\) because of the symmetry of the potential. The number of points is set by: \( n_\phi = l_{\text{max}} + 1 \) and \( n_\phi = m_{\text{max}} + 1 \). This is a DVR-like choice in the sense that the overlap matrix elements are all calculated exactly. To calculate the stretch eigenfunctions we use a PODVR [36, 37]. The PODVR functions are obtained from sinc DVR functions [53].

### RESULTS

We use the global potential energy surface (PES) of Kuhn et al. [54]. There are a number of variational calculations for H$_2$O$_2$ [15, 38, 45–52]. Many [49–52] use the PES of Kuhn et al. and their results can be compared to ours. In particular, we find good agreement with the most recent work of Chen, Ma, and Guo (CMG) [51, 52].

In Table I we summarize our primitive and contracted basis set and the number of bend quadrature points we use. The bend and stretch primitive basis sets are the same as those used by CMG [51, 52].

In our calculations a potential ceiling [55] is set at 30000 cm$^{-1}$. The same ceiling value is used by CMG [51, 52]. We also use a single precision vector for the potential [34] because the potential accuracy is not expected to be better than seven digits. We find that the levels obtained using the single precision potential vector agree with those obtained using a double precision potential vector to more than eight digits.

<table>
<thead>
<tr>
<th>Basis</th>
<th>( l_{\text{max}} )</th>
<th>( m_{\text{max}} )</th>
<th>( n_0 )</th>
<th>( n_1 )</th>
<th>( n_2 )</th>
<th>( n_\beta )</th>
<th>( n_\gamma )</th>
<th>( n_n )</th>
<th>( n_\beta^a )</th>
<th>( n_\beta^b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basis</td>
<td>31</td>
<td>23</td>
<td>16</td>
<td>7</td>
<td>7</td>
<td>24,576</td>
<td>11,236</td>
<td>784</td>
<td>280</td>
<td>107</td>
</tr>
</tbody>
</table>

a Obtained from \( E_{\text{total}}^{\text{m}} = 10,000 \text{ cm}^{-1} \).

b Obtained from \( E_{\text{total}}^{\text{m}} = 15,000 \text{ cm}^{-1} \).
TABLE II
Comparison of the memory and CPU costs for the standard product-iterative method and contracted-iterative method for HOOH.

<table>
<thead>
<tr>
<th></th>
<th>Product</th>
<th>Contracted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory estimatea,b</td>
<td>$\frac{5}{2}n_p^2\rho$</td>
<td>$n_p^2/2$</td>
</tr>
<tr>
<td>Actualb</td>
<td>278 Mb</td>
<td>244 Mb</td>
</tr>
<tr>
<td>Time per iteration estimateb,d</td>
<td>$n_p\rho_{n+1}d$</td>
<td>$\rho_b$</td>
</tr>
<tr>
<td>Actual</td>
<td>25 s</td>
<td>1.4 s</td>
</tr>
<tr>
<td>pre-Lanczos timec</td>
<td>176 s</td>
<td>3250 s</td>
</tr>
<tr>
<td>Total timed</td>
<td>25,000 s</td>
<td>4650 s</td>
</tr>
<tr>
<td>Spectral range (cm$^{-1}$)</td>
<td>124E + 03</td>
<td>26E + 03</td>
</tr>
<tr>
<td>Number of Lanczos-converged levelsf</td>
<td>32</td>
<td>66</td>
</tr>
</tbody>
</table>

The actual memory and time data are for a Microway computer with a 21264 667-MHz alpha CPU.

a The memory cost is estimated from the size of the dominant arrays. The fact that we use single precision for the potential and double precision for the Lanczos vectors has been taken into account.
b Assuming that $n_p > n_x$.
c 1 Mb = 1 megabyte.
d The estimated cost of the most expensive step in a matrix-vector product.
e The pre-Lanczos time for the product method is dominated by the cost of computing the potential vector. For the contracted method it is dominated by the cost of computing $F$ (which includes the cost of computing the potential).
f For 1000 iterations.
g For 1000 iterations. We deem levels Lanczos converged if $|\psi^k| < 10^{-4}$.

The memory and CPU costs of the contracted method are compared with those of the standard product-iterative method in Table II. The cost of the contracted-iterative calculations obviously depends on $n_p$ and $n_x$. As cut-off energies we choose $E^\text{cut} = 10000$ cm$^{-1}$ and $E^\text{cut} = 15000$ cm$^{-1}$. These are sufficiently high to permit the calculation of more than 100 accurate levels (see below).

We find that with the contracted method one can calculate more than twice as many energy levels as with the product-iterative method in less than one fifth of the CPU time (1000 iterations): it is extremely fast. The efficiency is attributable to two factors. First, the cost of a single matrix-vector product is very low. In fact, the cost of a single contracted matrix-vector product is significantly less than the cost of product matrix-vector product. This is due to the reduced basis size. Second, because the spectral range is reduced by a factor of 6, with the same number of iterations we are able converge more energy levels. For the purpose of comparing the product and contracted methods, assume that doubling the number of product basis iterations would double the number of converged energy levels (in fact, more than twice as many iterations would be required). To converge as many energy levels with the product method as are converged with the contracted method one would then need about 50000 s (for 2000 iterations). To calculate these levels with the contracted method one requires 4650 s (for 1000 iterations). The contracted method is therefore at least an order of magnitude faster.

To obtain the CPU cost we included the cost of evaluating the matrix-vector products and the cost of calculating $F$. For H$_2$O$_2$ the chosen energy cut-off time (3250 s) needed to calculate $F$ is about equal to the time required for 120 product basis matrix-vector products. Given the fact that one often needs thousands of iterations to converge energy levels of interest, the larger pre-Lanczos cost of the contracted method does not appear to be a serious problem.

Choosing values for $E^\text{cut}_p$ and $E^\text{cut}_c$ that are too small would unfairly bias the comparison in favor of the contracted methods. The values of $E^\text{cut}_p$ and $E^\text{cut}_c$ must be large enough that the contracted-iterative levels are very close to the product basis levels. This is the case. The difference between the product and contracted wave numbers is less than 0.1 cm$^{-1}$ for the first 69 $A+$ levels. For many of the higher levels the difference is also smaller than 0.1 cm$^{-1}$.

**Application to CH$_4$**

For H$_2$O$_2$ it is possible to use both the product-iterative and contracted-iterative methods. For CH$_4$ the product-iterative method is not feasible (unless one discards DVR basis functions, thereby losing the direct product character of the basis [41]). A contracted-iterative calculation is possible. Yu used contracted bend functions and uncontracted stretch functions to compute vibrational levels of CH$_4$ [29].
parity-adapted basis functions that are linear combinations of the product functions

\[ |l_1 l_2 m_3 m_3\rangle = \Theta_{l_1}^{m_1}(\theta_1, \phi_1) Y_{l_2}^{m_2}(\theta_2, \phi_2) Y_{l_3}^{m_3}(\theta_3, \phi_3). \]  
(23)

The parity-adapted basis functions are

\[ u_{l(2m,2m)}^p = \frac{N_{s_{l,m}}}{\sqrt{2}} [ |l_1 l_2 m_3 m_3\rangle + (-1)^p |l_1 l_2 m_3 m_3\rangle ] \]  
(24)

with

\[ N_{s_{l,m}} = \frac{1}{\sqrt{1 + \delta_{m,0} \delta_{s,0}}}, \]

where \( m = -m, m_3 \geq 0 \), and \( p = 0 \) and 1 correspond to even and odd parity, respectively. The quantum number \( m_2 \) assumes the values

\[
m_2 \in [0, m_{\text{max}}] \quad (m_3 = 0) \\
m_2 \in [-m_{\text{max}}, m_{\text{max}}] \quad (m_3 > 0)
\]

and the combination \( p = 1 \) and \( m_2 = m_3 = 0 \) is not allowed. \( m_{\text{max}} \) is the maximum value of \( m_2 \) and \( m_3 \). It is straightforward to derive equations for the bend kinetic energy matrix elements. They will be given in a future publication [57]. The simple and factorized form of kinetic matrix in the parity-adapted basis makes it easy to evaluate bend kinetic matrix–vector products with an order \( N \) scaling, where \( N \) is the size of the parity-adapted bend basis.

Potential matrix elements in the parity-adapted basis are evaluated by Gauss quadrature. We use Legendre weights and points for the polar angles and the trapezoidal rule for the azimuthal angles (see Ref. [57]). The number of required quadrature points can be reduced by a factor of two by exploiting the property

\[
V(\theta_{\alpha}, \theta_{\beta}, \phi_{\alpha}, \phi_{\beta}) = V(\theta_{\alpha}, \theta_{\beta}, \phi_{\alpha}, 2\pi - \phi_{\beta}, 2\pi - \phi_{\beta}).
\]

For a basis with maximum \( l \) quantum number \( l_{\text{max}} \) and maximum \( m \) quantum number \( m_{\text{max}} \), we choose \( n_\alpha = l_{\text{max}} + 1 \) and \( n_\beta = 2(m_{\text{max}} + 1) \) quadrature points. The total number of points is therefore \( 2(l_{\text{max}} + 1)^3(m_{\text{max}} + 1)^2 \). Although the parity-adapted basis functions are not products of functions of a single variable we have been able to evaluate matrix–vector products efficiently [57].

As primitive stretch basis functions we again use PODVR functions computed from a sinc DVR. The stretch primitive basis is much smaller than the bend primitive basis.

In addition to the parity symmetry, we also exploit the exchange symmetry between hydrogen atoms \( H_2 \) and \( H_3 \) (see Fig. 1 of Ref. [57]). Because we work in the parity-adapted basis, we can use \( G_2 \), composed of \( \{|E, (23)^\pm\| \} \) to label levels within a given parity block. The \( G_2 \) group only has two irreps: \( A \) and \( B \). By using the correlation between \( G_2 \) and \( T_d \), we can unambiguously assign the \( T_d \) symmetries to our calculated levels. In particular, two of each set of triply degenerate \( F_2 \) (or \( F_3 \)) levels are in the \( A \) block (or \( B \) block) and the other is in the \( B \) block (or \( A \) block).

**RESULTS**

We use the T8 potential surface of Schwenke described in Ref. [58]. It is an accurate ab initio global surface and has been used for some previous vibrational calculations [58, 59]. On this surface, the equilibrium bond length is \( 2.05791 \ a_0 \).

The bend energy and wave functions are obtained by using a parity-adapted FBR basis with \( l_{\text{max}} = m_{\text{max}} = 25 \). We use a potential ceiling value of \( 20000 \ \text{cm}^{-1} \). We retain the bend levels up to \( (n_2, n_3) = (4, 1) \), all of which are below \( 7670 \ \text{cm}^{-1} \). There are 133 \( A \) levels and 113 \( B \) levels. The bend wave functions are compacted with a cut-off energy equal to the potential ceiling value. This compaction cut-off energy is appropriate because the norm of all residual vectors is less than \( 3 \times 10^{-5} \).

The stretch energies and wave functions are obtained from a product of 10 PODVR basis functions for each stretch coordinate; these in turn are built from a sinc DVR basis in the range \([1.2a_0, 7.0a_0]\). We discard all the PODVR points whose potential energy is above \( 30000 \ \text{cm}^{-1} \). This reduces the basis size from \( 10000 \) to \( 5049 \). We retain all the stretch levels up to \( 20000 \ \text{cm}^{-1} \). There are 161 \( A \) levels and 99 \( B \) levels. We compute \( A \) and \( B \) stretch energy levels and wave functions. The stretch basis we use is so good that many of the low-lying doubly degenerate levels in \( A \) or \( B \) block are very nearly degenerate. This near degeneracy makes it impossible to calculate wave functions. To avoid this problem, we lift the degeneracy by multiplying the mass of the \( H_0 \) atom by \( (1 + 10^{-6}) \) (effectively making methane a \( C_{3v} \) molecule). We find that \( 10^{-6} \)
TABLE III
Vibrational wave numbers of methane (in cm\(^{-1}\)).

<table>
<thead>
<tr>
<th>((n_1,n_2,n_3))</th>
<th>Symmetry</th>
<th>This work</th>
<th>Schwenke</th>
<th>Column 4 – Column 3</th>
<th>Exp.(^a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(00)(00)</td>
<td>1 A1</td>
<td>9688.94</td>
<td>9691.56</td>
<td>2.62</td>
<td>1310.76</td>
</tr>
<tr>
<td>(00)(01)</td>
<td>1 F2</td>
<td>1311.39</td>
<td>1311.76</td>
<td>0.37</td>
<td></td>
</tr>
<tr>
<td>(00)(10)</td>
<td>1 E</td>
<td>1532.84</td>
<td>1533.26</td>
<td>0.42</td>
<td>1533.33</td>
</tr>
<tr>
<td>(00)(02)</td>
<td>2 A1</td>
<td>2589.11</td>
<td>2589.92</td>
<td>0.81</td>
<td>2587.04</td>
</tr>
<tr>
<td>(00)(02)</td>
<td>2 F2</td>
<td>2615.56</td>
<td>2616.30</td>
<td>0.74</td>
<td>2614.26</td>
</tr>
<tr>
<td>(00)(02)</td>
<td>2 E</td>
<td>2626.61</td>
<td>2627.40</td>
<td>0.79</td>
<td>2624.62</td>
</tr>
<tr>
<td>(00)(11)</td>
<td>3 F2</td>
<td>2830.78</td>
<td>2831.77</td>
<td>0.99</td>
<td>2830.32</td>
</tr>
<tr>
<td>(00)(11)</td>
<td>1 F1</td>
<td>2846.16</td>
<td>2847.04</td>
<td>0.88</td>
<td>2846.08</td>
</tr>
<tr>
<td>(01)(00)</td>
<td>3 A1</td>
<td>2912.95</td>
<td>2913.77</td>
<td>0.82</td>
<td>2916.48</td>
</tr>
<tr>
<td>(01)(01)</td>
<td>4 F2</td>
<td>3012.82</td>
<td>3014.20</td>
<td>1.38</td>
<td>3019.49</td>
</tr>
<tr>
<td>(00)(01)</td>
<td>4 A1</td>
<td>3062.68</td>
<td>3063.74</td>
<td>1.06</td>
<td>3063.65</td>
</tr>
<tr>
<td>(00)(10)</td>
<td>3 E</td>
<td>3064.20</td>
<td>3065.27</td>
<td>1.07</td>
<td>3065.14</td>
</tr>
<tr>
<td>(00)(03)</td>
<td>5 F2</td>
<td>3873.78</td>
<td>3873.34</td>
<td>4.56</td>
<td>3870.49</td>
</tr>
<tr>
<td>(00)(03)</td>
<td>5 A1</td>
<td>3911.30</td>
<td>3912.83</td>
<td>6.53</td>
<td>3909.18</td>
</tr>
<tr>
<td>(00)(03)</td>
<td>2 F1</td>
<td>3923.10</td>
<td>3929.98</td>
<td>6.88</td>
<td>3920.50</td>
</tr>
<tr>
<td>(00)(03)</td>
<td>6 F2</td>
<td>3934.34</td>
<td>3940.95</td>
<td>6.61</td>
<td>3930.92</td>
</tr>
<tr>
<td>(00)(12)</td>
<td>4 E</td>
<td>4103.41</td>
<td>4109.08</td>
<td>5.67</td>
<td>4105.15</td>
</tr>
<tr>
<td>(00)(12)</td>
<td>3 F1</td>
<td>4130.22</td>
<td>4136.63</td>
<td>6.41</td>
<td>4128.57</td>
</tr>
<tr>
<td>(00)(12)</td>
<td>6 A1</td>
<td>4134.71</td>
<td>4140.19</td>
<td>5.48</td>
<td>4132.99</td>
</tr>
<tr>
<td>(00)(12)</td>
<td>7 F2</td>
<td>4143.78</td>
<td>4150.21</td>
<td>6.43</td>
<td>4142.86</td>
</tr>
<tr>
<td>(00)(12)</td>
<td>5 E</td>
<td>4152.66</td>
<td>4159.19</td>
<td>6.53</td>
<td>4151.22</td>
</tr>
<tr>
<td>(00)(12)</td>
<td>1 A2</td>
<td>4163.26</td>
<td>4169.73</td>
<td>6.47</td>
<td>4161.87</td>
</tr>
<tr>
<td>(00)(01)</td>
<td>8 F2</td>
<td>4220.74</td>
<td>4222.23</td>
<td>1.49</td>
<td>4223.46</td>
</tr>
<tr>
<td>(00)(01)</td>
<td>9 F2</td>
<td>4313.10</td>
<td>4315.90</td>
<td>2.80</td>
<td>4319.21</td>
</tr>
<tr>
<td>(00)(01)</td>
<td>6 E</td>
<td>4316.46</td>
<td>4318.93</td>
<td>2.47</td>
<td>4322.15</td>
</tr>
<tr>
<td>(00)(01)</td>
<td>4 F1</td>
<td>4316.71</td>
<td>4319.61</td>
<td>2.90</td>
<td>4322.58</td>
</tr>
<tr>
<td>(00)(01)</td>
<td>7 A1</td>
<td>4317.30</td>
<td>4320.23</td>
<td>2.93</td>
<td>4322.72</td>
</tr>
<tr>
<td>(00)(21)</td>
<td>10 F2</td>
<td>4348.96</td>
<td>4355.27</td>
<td>6.31</td>
<td>4348.77</td>
</tr>
<tr>
<td>(00)(21)</td>
<td>5 F1</td>
<td>4363.59</td>
<td>4369.53</td>
<td>5.94</td>
<td>4363.31</td>
</tr>
<tr>
<td>(00)(21)</td>
<td>11 F2</td>
<td>4378.60</td>
<td>4384.32</td>
<td>5.72</td>
<td>4379.10</td>
</tr>
<tr>
<td>(00)(10)</td>
<td>7 E</td>
<td>4431.10</td>
<td>4432.60</td>
<td>1.50</td>
<td>4446.41</td>
</tr>
<tr>
<td>(01)(10)</td>
<td>6 F1</td>
<td>4530.19</td>
<td>4538.85</td>
<td>8.66</td>
<td>4537.57(^a)</td>
</tr>
<tr>
<td>(01)(10)</td>
<td>13 F2</td>
<td>4536.64</td>
<td>N/A</td>
<td></td>
<td>4543.76(^b)</td>
</tr>
<tr>
<td>(00)(30)</td>
<td>8 E</td>
<td>4591.44</td>
<td>4596.10</td>
<td>4.66</td>
<td>4592.03</td>
</tr>
<tr>
<td>(00)(30)</td>
<td>2 A2</td>
<td>4594.46</td>
<td>4599.15</td>
<td>4.69</td>
<td>4595.32</td>
</tr>
<tr>
<td>(00)(30)</td>
<td>8 A1</td>
<td>4594.73</td>
<td>4599.45</td>
<td>4.72</td>
<td>4595.56</td>
</tr>
<tr>
<td>(20)(00)</td>
<td>15 A1</td>
<td>5780.78</td>
<td>5783.20</td>
<td>2.42</td>
<td>5790.25</td>
</tr>
<tr>
<td>(11)(00)</td>
<td>27 F2</td>
<td>5852.46</td>
<td>5856.91</td>
<td>4.45</td>
<td></td>
</tr>
<tr>
<td>(02)(00)</td>
<td>18 A1</td>
<td>5959.44</td>
<td>5962.76</td>
<td>3.32</td>
<td>5968.09</td>
</tr>
<tr>
<td>(02)(00)</td>
<td>30 F2</td>
<td>5991.96</td>
<td>5996.29</td>
<td>4.33</td>
<td>6004.69</td>
</tr>
<tr>
<td>(02)(00)</td>
<td>20 E</td>
<td>6030.30</td>
<td>6034.60</td>
<td>4.30</td>
<td>6043.87</td>
</tr>
<tr>
<td>(00)(05)</td>
<td>33 F2</td>
<td>6401.71</td>
<td>6428.34</td>
<td>26.63</td>
<td></td>
</tr>
<tr>
<td>(00)(05)</td>
<td>20 A1</td>
<td>6429.52</td>
<td>6451.34</td>
<td>31.82</td>
<td></td>
</tr>
<tr>
<td>(00)(05)</td>
<td>34 F2</td>
<td>6474.02</td>
<td>6778.67</td>
<td>304.65</td>
<td></td>
</tr>
<tr>
<td>(00)(05)</td>
<td>35 F2</td>
<td>6529.48</td>
<td>6824.36</td>
<td>294.88</td>
<td></td>
</tr>
<tr>
<td>(00)(14)</td>
<td>5 A2</td>
<td>6702.66</td>
<td>6906.35</td>
<td>203.69</td>
<td></td>
</tr>
</tbody>
</table>

\(^a\) From Ref. [61].

\(^b\) There are typos in these two levels in Ref. [58].
LANCZOS ALGORITHM AND VIBRATIONAL SPECTRA

is large enough to make computing eigenvectors possible but small enough that errors in the energy levels are smaller than $10^{-3}$ cm$^{-1}$.

The size of the final product basis of bend and stretch wave functions is 63960; there are 32600 $A$ symmetry functions and 31360 $B$ symmetry functions. In the final Lanczos calculation a potential ceiling value of 30000 cm$^{-1}$ is used.

For the Schwenke surface the compaction ratio for the 5-D bend problem is about 0.01 if the cut-off energy is 20000 cm$^{-1}$. Although the compacted grid is much smaller than the direct product grid it is only useful if it is also smaller than the FBR basis. For the methane bend problem the FBR basis is only about a factor of 8 smaller than the direct product grid so in this case the compacted grid is useful.

One can think of the calculation as being divided into three steps: (1) Solve the stretch and bend problems; (2) compute matrix elements of $F$ and $T_i^0$; (3) use the Lanczos algorithm with the product contracted basis to get the final energy levels (and, if desired, wave functions). Calculating the bend wave functions in step (1) is the most time-consuming part of our calculation. We parallelized the calculation of the $F$ matrix by dividing it into blocks labeled by stretch DVR points. Matrix elements of $T_i^0$ in the bend contracted basis were also calculated block by block in parallel. This reduces the memory cost of the second stage so that the memory cost of the first and third stages is dominant. To calculate the bend basis we require about 1200 Mb (660 Mb is used to store the bend wave functions on the quadrature grids). To calculate the final levels we require about 1200 Mb (99% of which is used to store the $F$ matrix).

Some of our levels are reported in Table III. We show three groups of levels. The first group contains all the levels up to the $(00)(30)$ band where the notation is defined as $(n_1,n_3)(n_1,n_L)$. The second group contains the $v = n_1 + n_3 = 2$ stretch overtone. The third group contains the highest five levels reported by Schwenke [61].$^*$ Our levels are better converged than those of Schwenke. Up to 7000 cm$^{-1}$, the largest convergence error is about 0.28 cm$^{-1}$. Some of the high-energy levels reported by Schwenke are higher than our levels by as much as 300 cm$^{-1}$. Except for the pure stretch levels, our levels are closer to the experimental results than those of Schwenke. We have deliberately chosen our contracted basis such that the stretch cut-off energy is much higher than the bend cut-off energy. This will enable us to converge many high stretch overtones and combination bands with one or two quanta of bend excitation. Finally, due to the quality of our contracted basis, we can assign all the levels up to 8000 cm$^{-1}$ and many higher stretch overtone levels. For more detail on the methane calculation see Ref. [62].

**Conclusion**

There are two well-developed classes of methods for calculating vibrational energy levels. One relies for its efficiency on contracted basis functions that represent wave functions compactly. These are contracted-direct methods. The other relies for its efficiency on the favorable scaling of iterative eigensolvers (e.g., the Lanczos algorithm). These are product-iterative methods. The fundamental problem with which users of contracted-direct methods must contend is the $N^3$ scaling relation of direct linear algebra methods for the computation of eigenvalues or eigenvectors and eigenvectors. No matter how clever one is at devising good contraction schemes, $N^3$ is debilitating. The essential difficulty that users of product-iterative methods must face is the huge number of product basis functions required to converge energy levels of interest. True, it is possible to calculate levels of four-atom molecules with product basis sets with millions of functions [14, 15, 52, 60] but for larger molecules so many product basis functions would be required that prospects look somewhat bleak for the use of product-iterative methods. Iterative methods have key advantages but nonetheless it is probably fair to say that most calculations of vibrational energy levels done with iterative methods could also have been done with a contracted-direct method. By combining contracted basis functions and iterative eigensolvers it is possible to solve problems that cannot be solved with either contracted-direct or product-iterative methods. In this article, we demonstrate the power of the contracted-iterative method by using it to calculate numerically exact vibrational levels of HOOH and CH$_4$.

**ACKNOWLEDGMENTS**

The authors thank Martin Quack for providing the HOOH potential energy surface and for hospitality during their stay in Zurich. They are grateful to David Schwenke for providing his potential en-

$^*$David Schwenke has found an error in the code used to obtain his published numbers. Private communication.
ergy surface. Some of the calculations were done on a computer of the Réseau Québécois de Calcul de Haute Performance. This work has been supported by the Natural Sciences and Engineering Research Council of Canada.

References