Computer Physics Communications 128 (2000) 434–445

Computer Physics Communications

www.elsevier.nl/locate/cpc

Bound and quasi-bound rotation-vibrational states using massively parallel computers [☆]

Hamse Y. Mussa, Jonathan Tennyson [*]

Department of Physics and Astronomy, University College London, London WC1E 6BT, UK

Abstract

The parallel program suite PDVR3D calculates the rotation-vibration energy levels of triatomic molecules, up to the dissociation limit, using either Radau or Jacobi co-ordinates in a Discrete Variable Representation (DVR). The algorithm used to achieve a parallel version of this code, which has been substantially improved, and the performance of the codes is discussed. An extension of PDVR3D to the study of resonant (quasi-bound) states of the triatomic molecules is presented. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Discrete Variable Representation; Resonances; Parallelization; Dissociation

1. Introduction

Techniques used to solve the rotation-vibration Hamiltonian for triatomic molecules are now well developed [1–4]. However, there are applications, even for three atom systems on a single potential energy surface, which present a formidable computational challenge. Even light triatomic molecules may possess 10^6 bound rotation-vibration states [5].

In a previous article [6], henceforth referred to as I, we described how the DVR3D program suite of Tennyson and co-workers [7] was ported to massively parallel computers in manner appropriate for tackling large triatomic problems. The PDVR3D program described in I has been used to address a number of problems. In particular we have used it to obtain all the bound vibrational states of water and to obtain all states up to dissociation for a given set of rotational quantum numbers [8]. Since I, the algorithm for PDVR3D has been substantially improved.

In this article we summarize the algorithm used by PDVR3D to solve bound state rotation-vibration problem, detail the recent improvements to this algorithm and assess the current state of parallel diagonalizers. We also present an extension to PDVR3D for calculating quasi-bound (resonant) states of triatomic systems. This extension is an entirely new code as the DVR3D suite is only for purely bound states.

PDVR3D is portable across a wide range of distributed memory platforms, which use the Single Program Multiple Data (SPMD) loading strategy. Particular platforms for which we have tested PDVR3D include the 26-processor IBM SP2 machine at Daresbury Laboratory, the Edinburgh Parallel Computer Centre (EPCC) Cray Machines – the T3D which had 512 processors, and the T3E which has 328 application processors – at Edinburgh University, the 576 processor Cray T3E-1200E at Manchester University, and the 128 processor Cray-T3E in Bologna.

[☆] This paper is published as part of a thematic issue on Parallel Computing in Chemical Physics.

[*] Corresponding author. E-mail: j.tennyson@ucl.ac.uk.

0010-4655/00/$ – see front matter © 2000 Elsevier Science B.V. All rights reserved.
PII: S0010-4655(00)00058-8

We briefly overview the DVR3D suite in the following section, and its parallel version called PDVR3D. Sections 3−5 details the parallelization procedure and Section 6 considers the extension to quasi-bound states.

2. DVR3D and PDVR3D: an overview

The DVR3D program suite [7] solves the triatomic rotation-vibration problem using a discrete variable representation (DVR) in each of the vibrational coordinates. Each DVR consists of grids based on an appropriately chosen set of Gaussian quadrature points. The vibrational coordinates employed are two stretches, (r_1, r_2) and an included angle. θ. Use of an orthogonal kinetic operator, highly desirable in a DVR method, restricts these coordinates to either Jacobi (J) or Radau (R) coordinates. It is based on the Sequential Diagonalization and Truncation Approach (SDTA) [9] implementation of the DVR representation. The SDTA treats the coordinates in sequence by diagonalizing adiabatically separated reduced dimensional Hamiltonians; it gives a final Hamiltonian with high information content, i.e. with a high proportion of physically meaningful eigenvalues.

DVR3D contains a number of modules. Program DVR3DRJ solves pure vibrational problems when the rotational angular momentum, J, is 0. For rotationally excited states DVR3DRJ solves an effective 'vibrational' problem with a choice of body-fixed axes which can either be embedded along one of the radial coordinates or along their bisector [10]. Modules ROTLEV3 and ROTLEV3B are driven by DVR3DRJ and solve the fully coupled rotation-vibration Hamiltonians with the standard and bisector embeddings, respectively. A final module, DIPOLE3, computes dipole transitions but this will not concern us here.

The parallel version of the DVR3DRJ suite, PDVR3DRJ consists of independent modules PDVR3DR and PDVR3DJ for Radau coordinates with a bisector embedding, and other coordinate/embedding options, respectively. Thus PDVR3DR and PROTLEV3B solve nuclear motion Hamiltonians defined in Radau co-ordinates, with the radial motion symmetry treated via bisector embedding. PDVR3DJ and PROTLEV3 are appropriate for Jacobi and indeed Radau co-ordinates in the case where no radial symmetry is to be used with standard embeddings. Unlike DVR3D, PDVR3D has recently been extended to calculate resonance states.

Due to schedule restrictions most of the MPP machines we have tested require the use of 2^n processors per session, apart from the Cray-T3E at EPCC and the Cray T3E-1200E at Manchester University, which allow the use of some even number of processors, such as 80, 96 and 160 per session. However, it should be noted that the the code can run on any number of processors.

Because of this restriction and the memory requirement for solving a typical rotation-vibration problem, PDVR3DRJ is usually run on 64 or 128 of T3D processors, not less than 16 processors of the Cray-T3E's, and 8 or 16 of the Daresbury IBM SP2. PROTLEV3 or PROTLEV3B can use any configurable number of processors. The present versions of the parallel codes are all based on the use of MPI [11].

Our parallelization strategy addresses three crucial requirements:

(1) Good load balance. Distributing the work load equally over the processors.
(2) Good locality to avoid too much data transfer between processors.
(3) Avoiding excessive writing to or reading from a disk (I/O). As I/O is not performed in parallel on the MPP machines available to us, it can become a severe bottle-neck. It should be noted that DVR3D uses large amounts of I/O to reduce memory usage.

3. Parallelization of 'vibrational' problem

PDVR3DRJ deals with the 'vibrational' Hamiltonian as well as $J = 0$ calculations. It employs a Sequential Diagonalization and Truncation Approach (SDTA) DVR implementation to form compact 3D Hamiltonian matrices, $\widehat{H}^{\mathrm{SDTA}}$. Our experience has shown this method to be considerably more efficient than the non-STDA approach, see I, although studies on conventional architectures have found cases which favor use of non-STDA algorithms [12].

In the SDTA scheme, the order in which the intermediate diagonalization and truncation are performed can be important for achieving computational efficiency [13]. Most crucial is the choice of the final coordinate, denoted η below. The procedure used to select the intermediate eigen-pairs which are used to form the final Hamiltonian can also be important [9].

Considering these two factors, the parallelization requirements listed above, and the 2^n constraint, we have chosen a simple but efficient parallelization strategy. If the DVR has n_η 'active' (see I) grid points in co-ordinate η, then the calculation is spread over M processors by placing either n_η/M grid points on each processor or M/n_η processors for each grid point in coordinate η. As will be discussed later, the latter option is suitable when one wants to calculate many (3000 or more) rotational states for low values of J.

For clarity, we assume in the following parallelization discussions that there is exactly one grid point per processor. However, it should be noted that n_η/M & $M/n_\eta > 1$ schemes have also been implemented. The M/n_η algorithm is a new addition to PDVR3DRJ.

3.1. PDVR3DR

In Radau coordinates, both the stretches need to be treated equivalently and hence together if the permutation symmetry of AB_2 systems is to be included. The only choice of ordering in an STDA scheme thus reduces to treating the θ co-ordinate first or last. Treating θ last is the most efficient as the intermediate diagonalization will then occur after two rather than one co-ordinate has been considered. For the AB_2 system in which the mass of A is much greater than the mass of B (the assumption taken here), treating θ last is also best according to the theory of Henderson et al. [13].

In this section we present the parallelization of the SDTA 3D vibrational Hamiltonian. As parallelizing the $J > 0$ 'vibrational' Hamiltonian is practically the same as parallelizing the $J = 0$ vibrational Hamiltonian. The latter is described here and in Section 3.2.

In step one, the 2D Hamiltonian, $(\widehat{H}^{2D})^\gamma$, describing the symmetrized radial motion, at each active γ is constructed as [14]

$$
\begin{aligned}
&(H^{(2D)})^\gamma_{\alpha\alpha'\beta\beta'} \\
&\quad = (1+\delta_{\alpha\beta})^{-1/2}(1+\delta_{\alpha'\beta'})^{-1/2}\big(\Upsilon_{\alpha\alpha'}\delta_{\beta\beta'} \\
&\qquad + (-1)^q\Upsilon_{\alpha\beta'}\delta_{\beta\alpha'} + (-1)^q\Upsilon_{\beta\alpha'}\delta_{\alpha\beta'} + \Upsilon_{\beta\beta'}\delta_{\alpha\alpha'} \\
&\qquad + V(r_{1\alpha}, r_{2\beta}, \theta_\gamma)(\delta_{\alpha,\alpha'}\delta_{\beta,\beta'} + (-1)^q\delta_{\alpha,\beta'}\delta_{\beta,\alpha'})\big),
\end{aligned} \tag{1}
$$

where $q = 0$ or 1. Υ is the kinetic energy term, see I, and V is the potential; α and β are the radial grid points, whereas γ stands for the angular grid points. This Hamiltonian is diagonalized using a standard real symmetric diagonalizer, such as the NAG routine F02ABE [15] or the LAPACK [16] routine SYYEV to yield eigenvectors $(\mathbf{C}^{2D})^\gamma$ and eigen-energies $(E^{2D})^\gamma$.

As the angular grid points are mapped onto the processors, this step is very easy to parallelize. Each processor forms and then solves the $(\widehat{H}^{2D})^\gamma$ at the angular grid γ, which is then stored on the processor. This means that all the 2D Hamiltonian solutions are performed simultaneously.

In this step it is also necessary to select the lowest eigen-pairs to be used for the formation of the full and compact 3D $\widehat{H}^{\mathrm{SDTA}}$ Hamiltonian. Energy cut-off criteria are the common selection procedure in the SDTA [7,9], but in this procedure the number of functions used to construct the final 'vibrational' Hamiltonian varies with the angular grid point γ. This algorithm clearly would lead to unbalanced calculations, so the selection criteria was modified: an equal number of eigenstates of the symmetrized two-dimensional radial problem, n^{2D}, are retained for each active angular grid point. In this approach the size of the $\widehat{H}^{\mathrm{SDTA}}$ Hamiltonian is given by

$$N^{3D} = n^{2D} \times n_\gamma. \tag{2}$$

In step two, using the selected 2D eigen-pairs $(\mathbf{C}^{2D})^\gamma$ and $(E^{2D})^\gamma$, the final 3D H^{SDTA} Hamiltonian is constructed as

$$
\begin{aligned}
H^{(3D)}_{\gamma\gamma'l,l'} &= (E^{2D})^\gamma_l\delta_{\gamma,\gamma'}\delta_{l,l'} + \sum_\beta\sum_{\alpha=1}^{\beta-q}(1+\delta_{\alpha\beta})^{-1/2} \\
&\quad \times(1+\delta_{\alpha'\beta'})^{-1/2}(C^{2D})^\gamma_{\alpha\beta l}(C^{2D})^{\gamma'}_{\beta'\alpha'l'} \\
&\quad \times\big(L_{\alpha\alpha',\gamma\gamma'}\delta_{\beta\beta'} + (-1)^q L_{\alpha\beta',\gamma\gamma'}\delta_{\beta\alpha'} \\
&\quad \times(-1)^q L_{\beta\alpha',\gamma\gamma'}\delta_{\alpha\beta'} + L_{\beta\beta',\gamma\gamma'}\delta_{\alpha\alpha'}\big),
\end{aligned} \tag{3}
$$

where $l = 1, 2, 3, \ldots, n^{2D}$. The matrix $\mathbf{L}$ is the angular kinetic energy terms, see I.

Construction of the 3D $\widehat{H}^{\mathrm{SDTA}}$ Hamiltonian matrix was parallelized by making each processor construct a strip of the Hamiltonian, $\widehat{H}^{(3D)}(N^{3D}/M, N^{3D})$. This is done using the following steps:

(1) The **L** matrix is replicated on every processor.
(2) Each processor then performs the matrix-matrix multiplication, $\mathbf{L}(\mathbf{C}^{2D})^{\gamma}(\mathbf{C}^{2D})^{\gamma'}$, where $(\mathbf{C}^{2D})^{\gamma}$ is the 2D eigenvectors stored on the processor while $(\mathbf{C}^{2D})^{\gamma'}$ is the transpose of the 2D eigenvectors of every active angular grid. While doing the multiplication, the processor sends its $(\mathbf{C}^{2D})^{\gamma}$ to all the other processors. If $\gamma \neq \gamma'$, the processor uses the $(\mathbf{C}^{2D})^{\gamma'}$ it receives from another processor.
(3) As $(E)^{2D}$ is diagonal in both γ and l, it is added to the diagonal elements of the full 3D Hamiltonian.

In other words, construction of the diagonal Hamiltonian blocks is local to each processor, but the formation of the off-diagonal blocks is given by the matrix–matrix multiplication of the local set of vectors and the nonlocal $(\mathbf{C}^{2D})^{\gamma'}$. Each processor broadcasts its 2D eigenvectors ($\mathbf{C}^{2D}$) while at the same time building the local diagonal block. This means that all the blocks (diagonal or not) in the same row are constructed simultaneously. This approach is a simple and effective way of avoiding multiple data transfer and unbalanced loading, see Fig. 1. Our calculations on H_2O [8] and on O_3 [17] are examples of the use of PDVR3DR.

As shown by Wu and Hayes (WH) [18], it might not be necessary to construct the $\widehat{H}^{\mathrm{SDTA}}$ Hamiltonian matrix explicitly if an iterative parallel eigen-solver is used to yield the required eigen-pairs. This idea, parallel eigen-solvers and diagonalization of the Hamiltonian are discussed later.

3.2. PDVR3DJ

In the Jacobi formulation of the STDA procedure each co-ordinate is treated separately. This separable treatment is retained even when permutation symmetry of identical atoms is considered, but the symmetry treatment halves the number of angular DVR grid points used. Similarly for the Sutcliffe–Tennyson Hamiltonian [10] formulation in Radau co-ordinates where no symmetry is to be used, each co-ordinate is treated separately. This means that the $\widehat{H}^{\mathrm{SDTA}}$ Hamiltonian is formed in three stages. Hence in the SDTA DVR implementation scheme, there are six possible co-ordinate orderings.

Although selecting equal number of eigen-pairs from stage one and two, for each DVR grid of the co-ordinate treated last, avoids both load in-balance and I/O operations, we found it to be slow. Therefore, a different strategy from I was pursued whereby there is no separate treatment of the first co-ordinates. Instead the first two co-ordinates are treated together. In this case it is only necessary to decide which co-ordinate to treat last. Here cases where the θ co-ordinate or r_2 co-ordinate is treated last are considered. However, as explained before, the MPP machine load balance requirements and the physics of the system should determine which co-ordinate is to be treated last and distributed over the processors. Thus our studies on the HCP and H_3^+ [17] molecules used the ordering where r_2 is treated last, whereas our HN_2^+ calculation treats θ last [17,19].

The $\widehat{H}^{\mathrm{SDTA}}$ formation procedure is similar to that described in the previous section. However, here all the grid points of the co-ordinate treated last are tuned to the system, so there is no reason to reject particular points. Thus all DVR grids in the co-ordinate are taken as active. When θ is chosen as the last co-ordinate, the 3D vibrational $\widehat{H}^{\mathrm{SDTA}}$ is given by

$$\begin{aligned}H^{(3D)}_{\gamma\gamma' l,l'} &= (E^{2D})^{\gamma}_{l}\,\delta_{\gamma,\gamma'}\delta_{l,l'} + \sum_{\beta\beta'\alpha\alpha'} (C^{2D})^{\gamma}_{\alpha\beta l}(C^{2D})^{\gamma'}_{\beta'\alpha' l'} \\ &\quad\times \left(L^{(1)}_{\alpha\alpha'\gamma\gamma'}\delta_{\beta\beta'} + L^{(2)}_{\beta\beta'\gamma\gamma'}\delta_{\alpha\alpha'}\right) \qquad (4)\end{aligned}$$

while in the case where r_2 is the last co-ordinate, the 3D vibrational $\widehat{H}^{\mathrm{SDTA}}$ can be written as

$$\begin{aligned}\widehat{H}^{(3D)}_{\beta\beta' l,l'} &= (E^{2D})^{\beta}_{l}\,\delta_{\beta,\beta'}\delta_{l,l'} \\ &\quad + \sum_{\alpha\alpha'\gamma\gamma'} (C^{2D})^{\beta}_{\alpha\gamma l}(C^{2D})^{\beta'}_{\alpha'\gamma' l'}\Upsilon^{(2)}_{\beta\beta'}, \qquad (5)\end{aligned}$$

where, $l = 1, 2, 3, \ldots, n^{2D}$, E^{2D} and $\widehat{C}^{2D}$ are the 2D eigen-pairs.

4. Parallel diagonalizers

Obtaining a suitable parallel diagonalizer has proved a major problem. There are four parallel diagonalizers which one might regard as suitable for finding many or all solutions of a large Hamiltonian real symmetric problems: Scalapack [20], HJS [21], BFG [22], and PeIGS [23].

The real symmetric $\widehat{H}^{\mathrm{SDTA}}$ needs to be diagonalized to yield the final eigen-pairs. For this we need

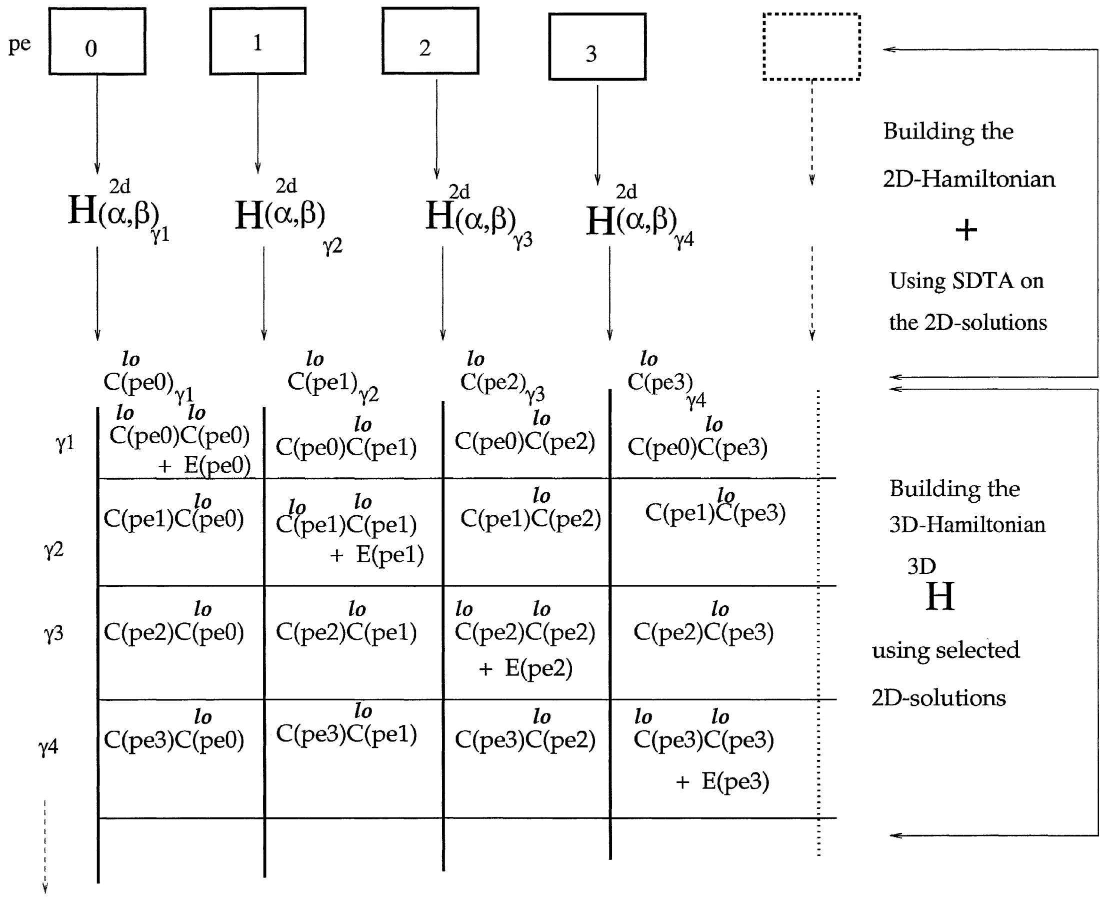

Fig. 1. Structure of the First-Step Hamiltonian matrix constructed by PDVR3DR. The SDTA matrix Hamiltonian is constructed by distributing the angular DVR grid points γ over the processors. The small boxes at the top of the figure indicate a processor while the label inside the box shows the identity of the processor. $H^{2D}(\alpha,\beta)_{\gamma_i}$; $i=1,2,\ldots,n_\gamma$, stands for the 2D-Hamiltonian of a particular γ. $C^{lo}(pe0)$, for example, is the set of eigen-vectors selected $C_j^{2D}(\alpha\beta;\gamma_1)$; $j=1,n^{2D}$, which is local to processor $pe0$. $E(pei)$; $i=0,1,\ldots,n_\gamma-1$ are the corresponding eigenvalues to the local eigenvectors.

to get both the h ($\leqslant N^{3D}$) lowest, orthogonal eigenvectors, C_i^{3D}, i = 1, h and associated eigenvalues, and use all the available processors. This rules out both the Scalapack [20], which does not guarantee an orthogonal set of vectors, and HJS [21], which is designed to use n^2 processors whereas we are constrained to use 2^n. However, the parallel program can use any other parallel diagonalizer which satisfy the two criteria.

The $\widehat{H}^{\mathrm{SDTA}}$ Hamiltonian matrix can be stored in either unpacked or packed triangular form. We have tested BFG [22] for the unpacked form and PeIGS [23] for the other. To get a good load balance for these eigen-solvers, PDVR3DR and PDVR3DJ redistribute the $\widehat{H}^{\mathrm{SDTA}}$ in lower or upper triangular form.

BFG is a one-sided Jacobi parallel diagonalizer. It requires the Hamiltonian in the strips it is constructed. In practice, we need to solve Hamiltonians of size 10,000 or more, however, the BFG diagonalizer proved incapable of diagonalizing 3D $\widehat{H}^{\mathrm{SDTA}}$ Hamiltonian matrices larger than 3000.

PeIGS, on the other hand, is a collection of commonly used linear algebra subroutines for computing the eigen-systems of the real standard problem

$$\mathbf{A}x = \lambda\mathbf{x} \qquad (6)$$

and the general eigen-system

$$\mathbf{A}x = \lambda \mathbf{B}x, \tag{7}$$

where **A** and **B** are dense and real symmetric matrices with **B** positive definite and **x** is an eigenvector corresponding to eigenvalue λ. PeIGS uses a flexible column distribution with packed storage for real symmetric matrices which are similar to the $\widehat{H}^{\mathrm{SDTA}}$. A serious drawback of PeIGS is its requirement for large amount of memory as workspace.

As standard real symmetric diagonalizers limited the size of calculations which could be attempted for 3D compact dense Hamiltonians, we also explored the possibility of using an iterative parallel eigen-solver. Such an approach has been advocated by a number of other workers [18,24–26], although, unlike many of the studies, we require eigenvectors as well as eigenvalues. To our knowledge there is only one parallel iterative eigen-solver, PARPACK [27], generally available.

PARPACK is a portable implementation of ARPACK [28] for distributed memory parallel architectures. ARPACK is a package which implements the Implicit Restarted Arnoldi Method used for solving large sparse eigenvalue problems [27].

Typically the blocking of the initial vector V is commensurate with the parallel decomposition of the matrix $\widehat{H}^{\mathrm{SDTA}}$. The user is free, however, to select an appropriate blocking of V such that an optimal balance between the parallel performance of PARPACK and the user supplied matrix–vector product is achieved. Interprocessor communication is inevitable, which can create large overheads, to perform the matrix-vector product as both the Hamiltonian matrix and the vector are distributed over the processors. To avoid any unnecessary interprocessor communications, we perform our matrix–vector product

$$w = H^{\mathrm{SDTA}} V \tag{8}$$

by using $\mathit{mpi\_allreduce}(V, x, n, \mathit{mpi\_real}, \mathit{mpi\_sum}, \mathit{comm}, \mathit{ierr})$. Each processor then does the matrix–vector multiplication on its local portions using the sgemv BLAS [29].

In the matrix–vector description given above, the H^{SDTA} Hamiltonian is constructed beforehand. However, as suggested by WH [18], one can perform the matrix–vector without explicitly constructing H^{SDTA}.

Table 1
Comparison between PeIGS and PARPACK performance in the solution of vibrational Hamiltonian matrix of dimension $N = 3200$ for the first 300 eigenvalues of different systems. The comparison was conducted using 8 EPCC Cray-T3E processors

Molecule	PeIGS/sec	PARPACK/sec
H_2O	245.3	104.7
HN_2^+	236.6	78.2
H_3^+	236.3	564.2
O_3	236.0	177.0

In this approach the multiplication (in matrix notation), see Eqs. (3)–(5), can be given by

$$y = \left(E^{2\mathrm{D}} + C^{(2\mathrm{D})\gamma} L C^{(2\mathrm{D})\gamma'}\right) V. \tag{9}$$

The advantage of WH's scheme is that their method requires less memory than ours, but as is clear from Eqs. (8) and (9), our approach needs a smaller number of operations per iteration. Therefore, as most of the CPU time is spent on the diagonalization, our method should perform better. As discussed in Section 5.3, we have used other strategies to overcome memory problems when they arise.

In addition to the memory requirements and flexibility, speed of the eigen-solver is another factor for which one might choose the eigen-solver. When tested on different systems, we found the relative performance of PeIGS and PARPACK to be molecule dependent, see Table 1.

In particular we found that for a given matrix size, PeIGS timings are almost system independent. Conversely the iterative diagonalizer shows strong system dependence, performing much worse for the strongly coupled H_3^+ molecule, in which the spectra is dense and the Hamiltonian eigenvalue distribution is low, but very well for the others. Furthermore, it is known that the speed of the PARPACK algorithm is dependent on the H^{SDTA} Hamiltonian eigenvalue distribution [18]. So this explains the convergence speed difference for PARPACK in the table. Similar behaviour for iterative diagonalizers on conventional computers was observed by Bramley and Carrington [24]. It should be noted that the comparisons are performed for a relatively small final matrix and on only 8 processors.

5. Rotational excitation

Once the eigenvectors of the 3D Hamiltonian have been obtained, the next step is to transform the compact eigenvectors to yield values for the amplitude of the eigenvectors at the original DVR grid points. As our SDTA DVR implementation uses two stages instead of three to form the compact and dense Hamiltonian, H^{SDTA}, the transformed eigenvector is simply

$$\psi^{ik}_{\alpha\beta\gamma} = \sum_{j=1}^{n^{2D}} C^{2D}_{jk}(\alpha\beta:\gamma) C^{3D}_{ik}(j,\gamma), \tag{10}$$

where k is the projection of J on the body-fixed z-axis ($k=0$ for $J=0$) and i is a particular eigenstate.

These transformed eigenvectors are then used as basis for calculating the resonances states and the rotational excited states. As in either case only h ($\leqslant N^{3D}$) eigenvectors need to be transformed, it is necessary to redistribute the eigenvectors so that those required are spread equally between the processors.

The 2D vectors, $\mathbf{C}^{2D}$, are already mapped on to processors using γ as they are generated during the Hamiltonian construction. When the 3D vectors, $\mathbf{C}^{3D}$, are generated by PeIGS, they are mapped by distributing vectors among the processors. No redistribution of the $\mathbf{C}^{3D}$ is required if the PARPACK eigensolver is used.

When considering rotationally excited state, $J > 0$, it is necessary to repeat all the above steps for each k. If both even and odd Wang parity, p, rotational states are to be calculated using the same First-Step vectors, then $J+2$ separate calculations are performed, one for each $k = 0, 1, \ldots, J$, plus an extra $k = 1$ with $p = 1$ calculation as this must be treated as a special case [7, 30].

5.1. PDVR3DR

For rotationally excited states, the Second-Step of the calculation introduces full rotation-vibration coupling by including the Coriolis operators, which are off-diagonal in k [7,30]. When the full Hamiltonian is expressed in terms of the First-Step solutions, the problem of constructing the fully coupled Hamiltonian matrix reduces to one of constructing the terms off-diagonal in k, which can be denoted, in the symmetrized Radau representation, by $B^{k,k'}_{i',i}$ $(i = 1, h)$,

$$\begin{aligned} B^{k,k'}_{i',i} = {} & -(1+\delta_{k,0}+\delta_{k'0})^{-1/2}\delta_{k',k\pm1}\delta_{q',1-q} C^{\pm}_{J,k} \\ & \times \sum_{\alpha\beta\gamma\gamma'} \psi^{J,k,i}_{\alpha\beta\gamma}\psi^{J,k',i'}_{\alpha\beta\gamma'}\big(M^{(1)}_{\alpha\alpha'} - M^{(2)}_{\beta\beta'}\big) J^{(1)}_{k\pm1,k,\gamma\gamma'} \\ & -(1+\delta_{k,0}+\delta_{k'0})^{-1/2}\delta_{k',k\pm2}\delta_{q',q} C^{\pm}_{J,k\pm1} C^{\pm}_{J,k} \\ & \times \sum_{\alpha\beta\gamma\gamma'} \psi^{J,k,i}_{\alpha\beta\gamma}\psi^{J,k',i'}_{\alpha\beta\gamma'} \\ & \times \big(M^{(1)}_{\alpha\alpha'} + M^{(2)}_{\beta\beta'}\big) J^{(2)}_{k\pm2,k,\gamma\gamma'}, \end{aligned} \tag{11}$$

where the $\mathbf{M}$ matrix is the familiar centrifugal distortion term, $C^{\pm}_{J,k}$ is the Coriolis coupling coefficient and $\mathbf{J}^{(i)}$ are angular integrals, see Ref. [7].

Because a four-dimensional transformation is needed to construct each matrix element, this step can be quite time consuming. It is therefore necessary to consider carefully how one might parallelize this problem. Wu and Hayes (WH) [18] , and Goldfield and Gray (GG) [31] have used MPP machines for $J > 0$ calculations of triatomic systems, although they have used different parallelization strategies.

WH defined a conceptual 3D-mesh where k is used as one of the indices, while GG have used a scheme mapping k to processors. WH use a single step solution procedure in contrast to our two-step procedure so their 3D-mesh scheme is not suitable for our solution strategy. The GG scheme is also not suitable in our case as there is insufficient memory to treat more than one J at a time, and only for large J are there enough k blocks to make distributing over processors a possibility. Alternatively, one can distribute over the stretching co-ordinates, (r_1, r_2), the angle θ or the eigenvectors h. Parallelizing over (r_1, r_2) is complicated by the radial DVR grid symmetrization. Conversely parallelizing over θ is superficially attractive as this is done in the First-Step and the number of γ's has already been chosen to map conveniently onto the number of processors. However, there are problems with distributing on θ. In this mode, each processor will build contributions to every element of the block, which will have to be assembled on a single processor once constructed. More seriously, a typical set of h $\psi^{ik}_{\alpha\beta\gamma}$ takes about 0.5 GB of memory, so these must be distributed. This method leads to an excessive amount of inter-processor communication.

In PDVR3DR we therefore distributed the building of the off-diagonal blocks in k, $\mathbf{B}^{k,k'}(h,h')$, over the M processors by placing $\mathbf{\Psi}^k(n_\alpha n_\beta n_\gamma, h/M)$ and $\mathbf{\Psi}^{k'}(n_{\alpha'} n_{\beta'} n_{\gamma'}, h'/M)$ vectors on each processor. Each processor thus builds rows of each block, $\mathbf{B}^{k,k'}(h/M, h')$. This is done using the vectors, $\mathbf{\Psi}^k(n_\alpha n_\beta n_\gamma, h/M)$ and $\mathbf{\Psi}^{k'}(n_{\alpha'} n_{\beta'} n_{\gamma'}, h/M)$, which are in the local memory and non-local $\mathbf{\Psi}^{k'}(n_{\alpha'} n_{\beta'} n_{\gamma'}, h'/M)$ vectors. For example, for processor 0 to build its portion of the block, $\mathbf{B}^{k,k'}(h/M,h')$, it uses its local $\mathbf{\Psi}^k(n_\alpha n_\beta n_\gamma, h/M)$ and $\mathbf{\Psi}^{k'}(n_{\alpha'} n_{\beta'} n_{\gamma'}, h'/M)$ vectors from other processors, see Fig. 2. This procedure is repeated for each block formed.

However, it should be noted that the above scheme requires two transformed vectors, $\mathbf{\Psi}^k$ and $\mathbf{\Psi}^{k'}$ each time. As it is necessary to minimize I/O on the employed MPP machines and keeping all the $\mathbf{\Psi}'$s, $J+1$ of them for each J, would require more real memory than what is available, we employ an algorithm which ensures that no more than three sets of vectors are retained in memory at any one time. The following steps show how the procedure works:

(1) For $k=0$; create $\psi^{k=0}$.
(2) For $k=1$: create $\psi^{k=1}$;
form $\mathbf{B}^{k'=1,k=0}$, using Eq. (11) , $k'=k+1$.
Gather all the portions on processor 0 and then let it write full block on a disk.
(3) For $k=2$ to J: create ψ^k;
form $\mathbf{B}^{k',k}$, using Eq. (11), $k'=k+2$;
replace ψ^{k+2} by ψ^{k+1};
form $\mathbf{B}^{k',k}$, using Eq. (11), $k'=k+1$;
Gather all the portions on processor 0 and then let it write full block to disk.

This algorithm uses significantly less I/O than the one presented in I.

5.2. PDVR3DJ

In PDVR3DJ the body-fixed z-axis is taken to be parallel to one of the radial co-ordinates. This embedding leads to a simpler Hamiltonian in which the Coriolis coupling only couples neighboring k blocks, i.e. k with $k \pm 1$ [7,30]. These off-diagonal elements have a particularly simple form if the angular co-ordinates is first transformed to associated Legendre polynomials, $|dj,k\rangle$. The back transformed eigenvector is [7]

$$Q^{ik}_{\alpha\beta j} = \sum_\gamma T^\gamma_j \psi^{ik}_{\alpha\beta\gamma}. \tag{12}$$

As for PDVR3DR, the construction of the fully coupled Hamiltonian is reduced to constructing the terms off-diagonal in k [7]. Here it is also necessary to parallelize the DVR to associated Legendre polynomial transformation. As the vectors are distributed over the processors, a copy of the transformation matrix, T^γ_j, is duplicated on every processor. As a result the transformation is done without any need for inter-processor communications. The off-diagonal block construction step then consists of a series of transformations which are performed in a similar fashion to PDVR3DR, but as they are only three-dimensional, see Ref. [7], are considerably less time consuming.

5.3. PROTLEV

The fully coupled rotation-vibration Hamiltonian matrix is given by

$$\mathbf{H}^{k,k'} = \delta_{k,k'}\mathbf{E}^k + \mathbf{B}^{k,k'}, \tag{13}$$

where $\mathbf{E}^k$ is a vector and $\mathbf{B}^{k,k'}$ is a matrix. PROTLEV3/PROTLEV3B reads the eigenvalues and the off-diagonal blocks from the disk. One processor does the reading and then broadcast the data to the rest. Both $\mathbf{E}^k$ and $\mathbf{B}^{k,k'}$ matrix elements are redistributed over the nodes, such that there is an equal portion of the full Hamiltonian on each processor.

However, the resulting structure depends on the eigen-solver used, see I for more details. As the size of the Hamiltonian is given by $h \times (J+1)$, and typically h is between 1500 (H_2O) and about 4000 (HN_2^+), for low J's the PeIGs is more suitable.

At this point it is worthwhile explaining why we extended I by introducing the M/n_η algorithm, see Section 3. Consider our HN_2^+ calculations [17,19] as an example. As the final $J>0$ Hamiltonian is expressed in terms of the First-Step eigen-pairs, the more first-step solutions used, the better the states converge. To converge all the 4900 or so even parity rotational states for $J=1$, it is necessary to use 3000 to 4000 'vibrational' eigenvectors for each k.

As explained in Section 4, PARPACK is preferred over PeIGS for solving the 'vibrational' problems. In

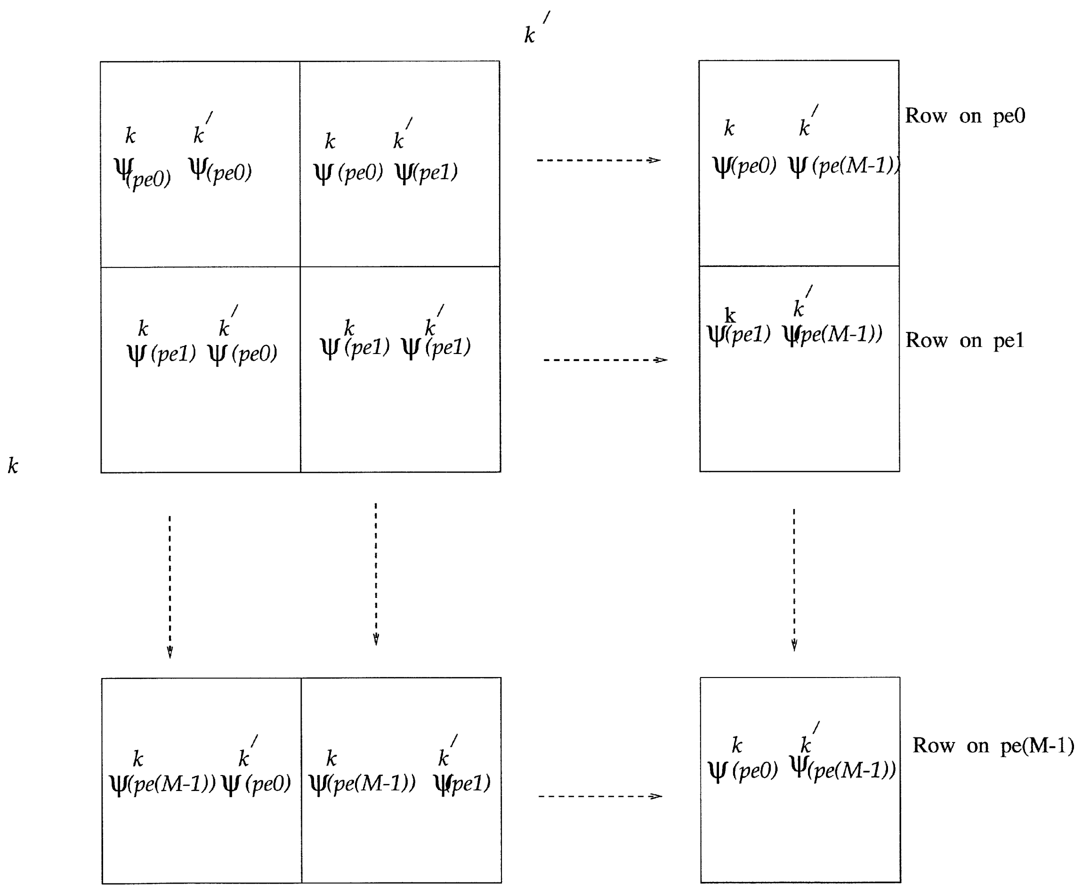

Fig. 2. Construction of the off-diagonal block in *k*, $\mathbf{B}^{k,k'}$(*h*,*h*). $\boldsymbol{\Psi}$ stands for the transformed eigenvectors which are explained in the text. *pei*; $i = 0, 1, \ldots, n-1$ are the processors.

PARPACK Upper Hessenberg matrix [27] $U_{\mathrm{h}}(n_{\mathrm{v}}, n_{\mathrm{v}})$ is replicated on every processor. Where $n_{\mathrm{v}} = 2 \times h$. Therefore for $h = 4000$, the $U_{\mathrm{h}}(n_{\mathrm{v}}, n_{\mathrm{v}})$ requires more memory per node than is available. Instead of trying to deal with the PARPACK Upper Hessenberg problem, as WH did [18], we modified our earlier scheme of $n_\eta/M \geqslant 1$, so that we can employ the PeIGS diagonalizer.

In the new algorithm, $\widehat{H}^{\mathrm{SDTA}}$ is distributed over M processors. Where $M \geqslant n_\eta$. In this scheme, a M/n_η number of processors build and solve the $(\widehat{H}^{\mathrm{2D}})^\eta$ for each η grid, but each processor retains a different portion of the lowest n^{2D} 2D eigen-pairs. After this the construction of the 3D-Hamiltonian is carried out in a fashion similar to that of the $n_\eta/M \geqslant 1$ method, see Section 3.1. The only notable difference is that the new method requires more inter-processor communication. This is due to the fact that for each η the chosen set of eigen-pairs from the $(\widehat{H}^{\mathrm{2D}})^\eta$ solution is spread over M/n_η processors. As $N^{\mathrm{3D}} = n^{\mathrm{2D}} \times n_\eta$, see Eq. (2), the bigger M is the smaller the $\widehat{H}^{\mathrm{SDTA}}$ portion on each processor becomes. Consequently PeIGS requires less memory per node and big Hamiltonians, which could not be solved with the $n_\eta/M \geqslant 1$ method, can be diagonalized on a larger number of processors.

This trick enabled us to calculate all the rotational states of HN_2^+ for $J = 1$; within a tenth of a wavenumber or better [17,19]. This calculation showed that a previous study on rotationally excited states of this system gave results which were not converged [32].

6. Resonances

In principle, quasi-bound states can be defined using different properties of the wavefunction [33]. However, recent methods have employed optical potentials, finite square integrable bases and diagonalization procedures [26,34,35].

Using such a procedure to obtain resonance states, it is necessary to solve

$$\widehat{\mathbf{H}}' = \widehat{\mathbf{H}}^{\mathrm{SDTA}} - i\mathbf{U}(\eta). \tag{14}$$

U is the so-called optical potential which is introduced in the asymptotic region of the real potential V. η is the dissociative co-ordinate and usually is the co-ordinate treated last, see Section 3.

As constructing and diagonalizing the complex Hamiltonian in its above form could be both time and computer memory consuming, we first solve the $\widehat{H}^{\mathrm{SDTA}}$ and then use some of the eigen-pairs as basis for constructing the $\widehat{H}'$. For $J = 0$ in this basis, matrix elements of the complex Hamiltonian can be written as

$$\langle \psi^n_{\alpha\beta\gamma} | \widehat{H}' | \psi^m_{\alpha\beta\gamma} \rangle = e_n \delta_{nm} - i \langle \psi^n_{\alpha\beta\gamma} | U | \psi^m_{\alpha\beta\gamma} \rangle. \tag{15}$$

Within a DVR, the above equation can be simplified further as

$$\langle \psi^n_{\alpha\beta\gamma} | \widehat{H}' | \psi^m_{\alpha\beta\gamma} \rangle = e_n \delta_{nm} - i \sum_{\alpha\beta\gamma} \psi^n_{\alpha\beta\gamma} \psi^m_{\alpha\beta\gamma} U(\eta_\gamma). \tag{16}$$

Where ψ^i is defined by Eq. (10).

Constructing the full Hamiltonian in parallel is quite simple. It is similar to the $\widehat{H}^{\mathrm{SDTA}}$ formation discussed in Section 3. When $n = m$, the diagonal blocks are formed using the locally available transformed eigenvectors, whereas for $n \neq m$ case, each processor should receive the required $\psi^m_{\alpha\beta\gamma}$ from another processor. In this procedure each processor builds a $\widehat{H}'(N/p, N)$ portion of the Hamiltonian; N being the size of the Hamiltonian and p is the number of processors employed.

PARPACK is used for diagonalizing the complex Hamiltonian matrix where the imaginary part of the resulting eigenvalues yields an estimate for the width of the resonance and the real part gives the energy.

However, to obtain well converged resonance states, it is necessary to vary the range and height of $\mathbf{U}$, and possibly the number of basis functions used. As diagonalizing the $\widehat{H}^{\mathrm{SDTA}}$ Hamiltonian again and again, or writing on and reading from a disk in each time is costly, we solve $\widehat{H}^{\mathrm{SDTA}}$ once and keep the ψ's in core. Only the construction and diagonalization of $\widehat{H}'$ with different complex potential parameters and different number of basis is repeatedly performed.

Recently Skokov and Bowman [35] computed HOCl resonance states for the potential energy surface (PES) of Skokov et al. [36]. We have tested our resonance algorithm for HOCl using the PES of Skokov et al. For Cl–OH Jacobi coordinates, we used 35, 80 and 80 DVR grid points for $r_1 (= r_{\mathrm{OH}})$, r_2 and θ, respectively. r_2, the dissociating coordinate, was treated last.

Building, diagonalizing and truncating the 2D-Hamiltonian takes 18.4 minutes. Constructing a final compact 3D Hamiltonian of 6400 dimension, costs less than 2.5 minutes. PARPACK is then employed for solving the $\widehat{H}^{\mathrm{SDTA}}$ to yield 1200 eigen-pairs, well above the dissociation limit [35]. The diagonalization wall time, on 80 T3E-1200 processors, is 21 minutes. Performing the the eigen-vectors transformation, see Eq. (12), takes about 2 minutes.

For particular optical potential height and range, all the 1200 transformed eigenvectors were used for forming the complex Hamiltonian, $\widehat{H}'(1200, 1200)$. Only 3 minutes of real time was required to construct and diagonalize the Hamiltonian on the 80 processors. It should be noted that only the eigenvalues were calculated.

These results are only preliminary. We are currently working on their optimization. Comparisons with the HOCl results of Skokov and Bowman give excellent agreement for resonance positions and widths which agree within the accuracy of the method.

7. Program performance

The speed up of a program on M processors is given by,

$$S_M = \frac{T_1}{T_M}, \tag{17}$$

where T_M is the time it takes the program to run on M processors. The speed up directly measures the effects of synchronization and communication delays on the performance of a parallel algorithm. Ideally S_M should grow linearly with M. In practice, this is very

Table 2
PDVR3DJ performance on the Manchester University Cray T3E-1200E in seconds for a triatomic molecule 'vibrational' calculations with $J = 1$ and a compact final Hamilt onian of dimension $N = 5120$ for HN_2^+. Note that only 512 eigenvectors were transformed (see the text). np = number of processors. hcet = time for Hamiltonain construction and the transformation of selected eigenvectors. bb = time for the *k* off-diagonal bloks formation. Finally diag stands for the "vibrational" Hamiltonian diagonalization time. Note that the time is in seconds

np	hcet	bb	diag
16	1330.14	2180.25	1657.06
32	626.24	1113.88	885.99
64	300.67	566.20	498.02
128	147.88	285.38	311.92

Table 3
PROTLEV3 performance on the Manchester University Cray T3E-1200E in seconds for a triatomic molecule with $J = 1$ and a final Hamiltonian of dimension $N = 1024$ for HN_2^+. See the text for more details. np = number of processors and fc = fully foupled Hamiltonian Solution. The time is in seconds

np	fc
2	84.05
4	46.12
8	27.55
16	16.83
32	12.92

unlikely because the time spent on each communication increases with the number of processors and thus the scalability deteriorates when a large number of processors is employed.

Here only the scalability for the PDVR3DJ and PROTLEV3 will be discussed. We have tested the performance of both PDVR3DJ and PROTLEV3 using the Cray T3E-1200E at Manchester University.

Table 2 illustrates the scalability of the 'vibration'program PDVR3DJ using PeIGS for solving the Hamiltonian. The Hamiltonian construction and the transformation of the chosen eigenvectors procedure shows near-perfect scalability, because the parallelization strategy chosen minimizes data transfer, and usage of BLAS routines, which are optimized for the MPP machines, maximizes the performance of the floating point operations. The in-core diagonalizer, PeIGS, requires considerable interprocessor communication so some degradation with increasing number of processors is to be expected, see I. This problem is magnified when eigenvectors are also required. This is an intrinsic problem of parallel direct diagonalizers where an implicit Gram-Schmidt step must be implemented to conserve orthogonality between eigenvectors extracted on different processors, see [37].

It should be noted that only 512 'vibrational' eigenvectors, 10% of the total eigenvectors, were transformed (see Section 5).

Table 3 shows the equivalent behaviour for PROTLEV3, only in-core diagonalization, using PeIGS is considered as it is rapid and gives all the eigen-pairs. In this case matrix construction requires I/O. Since the I/O gates are not dedicated to us, this may introduce some arbitrariness to the performance. However, the actual Hamiltonian construction time is small. Therefore the diagonalization is the main feature in PROTLEV3. So the above argument applies and indeed confirmed by the figure.

8. Conclusions

We have developed parallel programs for treating the vibration-rotation motion of the three-atom system using either Jacobi or Radau co-ordinates. These programs are based on the published DVR3D program [7] which is designed for computers with traditional architectures. Significant algorithm changes were required, in particular to reduce I/O interfaces in the original programs. The parallel suite shows good scalability and can be used for big and challenging calculations. Generally tests of presently available parallel eigensolvers favor the iterative eigen-solvers for their low workspace requirements.

Employing the parallel code we have studied the rotation-vibration of a number of key triatomic system, such as H_2O, O_3, HN_2^+, H_3^+, and HCP. We have extended the parallel program to look at vibrational resonances which we have studied for HOCl. We are currently extending our method to the resonances in rotationally excited molecules.

Acknowledgement

This work was performed as part of the ChemReact High Performance Computing Initiative (HPCI) Consortium. We thank Robert Allan and the members of HPCI centres at Daresbury Laboratory, Manchester University and Edinburgh University for their help. We also thank Dr. Serge Skokov for supplying his HOCl potential energy surface.

References

[1] J. Tennyson, B.T. Sutcliffe, J. Chem. Phys. 77 (1982) 4061.
[2] S. Carter, N.C. Handy, Mol. Phys. 52 (1984) 1367.
[3] J.S. Lee, D. Secrest, J. Chem. Phys. 92 (1988) 182.
[4] D. Estes, D. Secrest, Mol. Phys. 59 (1986) 569.
[5] G.J. Harris, S. Viti, H.Y. Mussa, J. Tennyson, J. Chem. Phys. 109 (1998) 7197.
[6] H.Y. Mussa, J. Tennyson, C.J. Noble, R.J. Allan, Comput. Phys. Commun. 108 (1998) 29.
[7] J. Tennyson, J.R. Henderson, N.G. Fulton, Comput. Phys. Commun. 86 (1995) 175.
[8] H.Y. Mussa, J. Tennyson, J. Chem. Phys. 109 (1998) 10 885.
[9] Z. Bacic, J.C. Light, Annu. Rev. Phys. Chem. 40 (1989) 469.
[10] B.T. Sutcliffe, J. Tennyson, Int. Quantum Chem. 29 (1991) 183.
[11] Message Passing Interface; the MPI Standard is available from netlib2.cs.utk.edu by anonymous ftp.
[12] M.J. Bramley, T. Carrington, Jr., J. Chem. Phys. 101 (1994) 8494.
[13] J.R. Henderson, C.R. Le Sueur, S.G. Pavett, J. Tennyson, Comput. Phys. Commun. 74 (1993) 193.
[14] N.G. Fulton, PhD Thesis, London University (1995).
[15] NAG Fortran Library Manual, Mark 17, Vol. 4 (1996).
[16] LAPACK Users' Guideis available in html form from WWW URL http://www.netlib.org/lapack/lug/lapack_lug.html.
[17] H.Y. Mussa, PhD Thesis, London University (1998).
[18] X.T. Wu, E.F. Hayes, J. Chem. Phys. 107 (1997) 2705.
[19] H.Y. Mussa, S. Schmatz, M. Mladenovic, J. Tennyson, J. Chem. Phys. (to be submitted).
[20] Scalapack Users' Guide is available in html form from WWW URL http://www.netlib.or. org/scalapack/.
[21] B.A. Henderson, E. Jessup, C. Smith, A parallel eigensolver for a parallel eigensolver for dense symmetric matrices, available via anonymous ftp from ftp.cs.sandia.gov/pub/papers/bahendr/eigen.ps.Z.
[22] I.J. Bush, Block factored one-sided Jacobi routine, following: R.J. Littlefield, K.J. Maschhoff, Theor. Chim. Acta 84 (1993) 457.
[23] G. Fann, D. Elwood, R.J. Littlefield, PeIGS Parallel Eigensolver System, User Manual available via anonymous ftp from pnl.gov.
[24] M.J. Bramley, T. Carrington Jr, J. Chem. Phys. 99 (1993) 8519.
[25] M.J. Bramley, J.W. Tromp, T. Carrington Jr, G.C. Corey, J. Chem. Phys. 100 (1994) 6175.
[26] V.A. Mandelshtam, H.S. Taylor, J. Chem. Soc. Faraday Trans. 93 (1997) 847.
[27] R. Lehoucq, K. Maschhoff, D. Sorensen, C. Yang, PARPACK, available from ftp://ftp.caam.rice.edu/pub/allowbreak people/kristyn.
[28] R.B. Lehoucq, D.C. Sorensen, P.A. Vu, C. Yang, ARPACK: Fortran subroutines for solving large scale eigenvalue problems. The ARPACK package is available from http://www.caam.rice.edu/software/ARPACK/index.html.
[29] BLAS Users' Guide is available in html form from WWW URL http://www.netlib.org/blas/lug/blas.html.
[30] J. Tennyson, B.T. Sutcliffe, Int. Quantum Chem. 42 (1992) 941.
[31] E.M. Goldberg, S.K. Gray, Comput. Phys. Commun. 98 (1996) 1.
[32] S. Schmatz, M. Mladenovic, Ber. Bunsenges. Phys. Chem. 101 (1997) 372.
[33] B.R. Junker, Advan. At. Mol. Phys. 18 (1982) 287.
[34] G. Jolicard, E.J. Austin, Chem. Phys. 103 (1986) 295.
[35] S. Skokov, J.M. Bowman, J. Chem. Phys. 110 (1999) 9789.
[36] S. Skokov, J. Qi, J.M. Bowman, K.A. Peterson, J. Chem. Phys. 109 (1998) 2662.
[37] G. Fann, R.J. Littlefield, Parallel inverse iteration with reorthoganalisation, in: Proc. 6th SIAM Conf. Parallel Processing for Scienticif Computing (SIAM, Philadelphia, PA, 1993) p. 409.