ELSEVIER

Computer Physics Communications 108 (1998) 29–37

Computer Physics Communications

Rotation–vibration calculations using massively parallel computers

Hamse Y. Mussa [a], Jonathan Tennyson [a,1], C.J. Noble [b], R.J. Allan [b]

[a] *Department of Physics and Astronomy, University College London, London WC1E 6BT, UK*

[b] *Daresbury Laboratory, Daresbury, Warrington WA4 4AD, UK*

Received 5 July 1997

Abstract

Parallel implementations of the DVR3D program suite (J. Tennyson, J.R. Henderson, N.G. Fulton, Comput. Phys. Commun. 86 (1995) 175) are presented. These programs calculate the rotation–vibration energy levels of triatomic molecules using either Jacobi or Radau coordinates based on a Discrete Variable Representation (DVR). Sample calculations are presented for water at its dissociation limit. © 1998 Elsevier Science B.V.

PACS: 33.20

Keywords: Parallelization; Vibrations; Rotations; Matrix diagonalization

1. Introduction

Calculations of rotation–vibration spectra of small molecules by direct solution of the nuclear motion Schrödinger equation are beginning to make a major impact in the area of experimental spectroscopy [1]. However, although the techniques used to solve these problems are now well developed, there are applications, even for triatomic systems on a single potential energy surface, which present a formidable computational challenge.

As an example, consider attempts to compute comprehensive linelists for water to aid radiative transport models of cool stellar atmospheres and elsewhere. The most ambitious of these calculations, by Partridge and Schwenke [2], compute over 300 million rotation–vibration transitions of water. Yet, recent analysis of their results suggest that even this linelist may not be accurate because insufficiently large basis sets were employed for the nuclear motion step of the calculation [3].

As a second example, one should consider attempts to calculate spectra of molecules at their dissociation limit. The highly complicated, infrared near-dissociation spectra of H_3^+ and its isotopomers of Carrington and co-workers [4,5] has provided a challenge to theory for more than a decade. Although a number of quantal studies have successfully treated rotationless problems for H_3^+ at dissociation [6–9], only one has managed to include rotational motion [10]. This limited calculation only considered states with rotational angular momentum, J, less than or equal to two yet still used many weeks of computer time. It is estimated [11] that H_3^+ has some 2×10^5 bound rotation–vibration states. Because of its small mass, this is probably considerably less than other chemically bound triatomic systems. Calculations

[1] E-mail: j.tennyson@ucl.ac.uk

PII S0010-4655(97)00125-2

of this magnitude clearly represent a computational grand challenge.

One class of methods which have been very successful at treating large nuclear motion problems is based on a particular finite element representation of the problem known as the Discrete Variable Representation [12]. In this paper we consider how a set of DVR programs, in particular the DVR3D package of Tennyson and co-workers [13], can be ported to massively parallel computers in a manner appropriate for tackling the problems discussed above. Very recently, Wu and Hayes [14] have also addressed the problem of performing triatomic rovibrational calculations on a Cray T3D machine. Our method is designed to tackle larger problems than theirs and therefore represents a more radical change from algorithms used on sequential machines.

2. PDVR3D and PROTLEV: an overview

The DVR3D program suite [13] contains a number of modules. Program DVR3DRJ solves the $J = 0$ vibrational problems using either atom–diatom Jacobi coordinates or Radau coordinates. It also solves the effective 'vibrational' problems required for a two-step treatment of the rotational excitation problem [15]. Programs ROTLEV3 and ROTLEV3B are driven by DVR3DRJ and solve the fully coupled rotation–vibration problem respectively for Jacobi and a symmetrized grid in Radau coordinates. A final module, DIPOLE3, computes transition dipoles but will not concern us here.

In this article parallel versions of the programs DVR3DRJ, ROTLEV3 and ROTLEV3B are presented. However, parallelization requires a number of changes in the solution algorithm which were best achieved by splitting program DVR3DRJ. The parallel version thus consists of PDVR3DR and PROTLEV3R which solve the nuclear motion problem for symmetrized Radau coordinates with a bisector embedding [16], and PDVR3DJ and PROTLEV3J which are appropriate for Jacobi coordinates or indeed Radau coordinates in the case where no symmetry is to be used.

When determining parallelization procedures it is necessary to consider both the construction of the scientific problem being addressed and the architecture of the computer being employed. The codes themselves are not molecule or problem specific but it is helpful to use specific examples as these define typical limits of a calculation. In this work we will consider water and H_3^+ at their respective dissociation limits as prototype systems to be treated in symmetrized Radau and Jacobi coordinates, respectively. In fact, for both water [17] and H_3^+ [10] the DVR3D programs have been adapted relatively easily to run very efficiently on parallel shared memory architectures. However, these machines have few processors and the benefits in throughput are limited. In this work we consider parallel distributed memory computers.

The intended architecture is a parallel message passing computer. PDVR3D and PROTLEV are designed for parallel and distributed memory computers which use a Single Program Multiple Data (SPMD) loading strategy and support MPI or PVM message passing libraries. Particular architectures used are the 16-processor IBM SP2 machine at Daresbury Laboratory (now upgraded to 26 processors) and the Cray-T3D at Edinburgh Parallel Computer Centre, which was upgraded from 256 to 512 processors during the course of this work. For algorithmic reasons, described below, the PDVR3DR/PDVR3DJ programs usually run on 64 or 128 processors of the T3D while the algorithm used for PROTLEV3R/PROTLEV3J can use any number of processors up to the machine capacity.

The IBM SP2 has 256 MB (or 32 Mwords) of user accessible memory per processor, each of which has a peak performance of 266 Mflops. Communications bandwidth is approximately 35 MB s^{-1} with MPI. The Cray T3D has only 64 MB memory per processor, each of which has a peak performance of 150 Mflops. Communications are relatively fast: a bandwidth of 120 MB s^{-1} with the native SHMEM_PUT and up to 50 MB s^{-1} with MPI, but I/O is constrained to go via two specific gateways whereas the IBM has a 2 GB scratch disk per node. Originally the programs were written using PVM [18] but we switched to MPI [19] when this became available on the T3D. The DVR3D program suite was written in Fortran 77, but modifications use Fortran 90 constructs.

Both computers were used for development purposes. Since the intention of the work is to build and diagonalize large Hamiltonian matrices, our work has been done mainly on the T3D. Therefore, the results presented here are for the Cray-T3D. However, sim-

ilar results were obtained on the IBM SP2 when the size of the problem matched the available memory.

3. Symmetrized Radau coordinate system

For an AB_2 molecule, Radau coordinates [20] consist of two symmetry related stretching coordinates and an included angle. For molecules, such as water, where the central atom is significantly heavier than the others ($m_A \gg m_B$), these coordinates are similar to those given by the two AB bond distances, r_1 and r_2, and the $A\widehat{B}A$ bond angle, θ. However, Radau coordinates yield a much simpler kinetic energy operator than bond length–bond angle coordinates and are thus more suitable for a procedure, such as the DVR, where the kinetic energy gives all the off-diagonal matrix elements.

The DVR3D programs use sequential diagonalization and truncation [12,13] to create a compact final Hamiltonian matrix. For such schemes, the order in which this diagonalization and truncation is performed can be crucial in determining computational requirements [22]. Because the Radau coordinate symmetrization procedure used by DVR3DRJ is based on symmetrizing DVR grid points and not basis functions [21], the stretches need to be treated equivalently and hence together, so the only choice is whether to treat the angular θ coordinate first or last. Only one choice is considered here: that the final diagonalization should be for a grid of θ points for two reasons. For water, treating θ last is indicated as being most efficient in the analysis of Henderson et al. [22]. Furthermore, treating θ first would mean that the intermediate diagonalization would occur after one not two coordinates had been considered which would remove most of the advantage of this method. Below the DVR grids in (r_1, r_2, θ) will be denoted (α, β, γ).

3.1. Step-one: PDVR3DR

The basic method of distributing the DVR calculation over N processors was to place one angular grid point, γ, on each processor. For this reason this coordinate was usually represented by either $n_\gamma = 64$ or 128 'active' grid points.

In the standard energy-selected diagonalization and truncation procedure, the number of functions used to construct the final Hamiltonian varies with γ. This algorithm clearly leads to an unbalanced calculation, so the selection criterion was modified. In the modified version, an equal number of eigenstates of the two-dimensional problem, n^{2D}, was retained for each angular 'active' grid point. Inactive grid points were simply dropped from the calculation. One could include all γ grid points in a calculation, but for water geometries near linear OHH configurations have very high energy, and indeed the Tennyson and Sutcliffe [16] strategy for implementing the bisector embedding involves dropping at least the point nearest linearity. In practice our water calculations drop about 10% of the points, but tests we have performed on ozone involved dropping more than half, for ozone points are dropped from both ends of the angular grid.

Construction and diagonalization of intermediate, 2D Hamiltonians is performed simultaneously on each processor using a standard diagonalizer. We use the NAG routine F02ABE [23] for this purpose. In this approach the size of the final Hamiltonian is given by

$$N^{3D} = n^{2D} \times n_\gamma . \tag{1}$$

Construction of the final 3D Hamiltonian matrix requires all the vectors, $C_j^{2D}(\alpha\beta : \gamma), j = 1, n^{2D}$, obtained for the n_γ active angular grid points. A strip of the Hamiltonian is constructed on each processor which guarantees that one set of vectors is always correctly located. For blocks off-diagonal in γ, a second set has to be imported from another processor.

Time-wise the most critical step is diagonalization of the final 3D Hamiltonian. For this we need to get both the h ($\leq N^{3D}$) lowest, orthogonal eigenvectors, $C_i^{3D}(j, \gamma), i = 1, h$ and associated eigenvalues, and use all the available processors. This rules out both the diagonalizers Scalapack [24], which does not guarantee an orthogonal set of vectors, and HJS [25], which is designed to use n^2 processors whereas we are constrained to use 2^n. However, the code can use any other diagonalizer which satisfy the two criteria.

The (real symmetric) Hamiltonian matrix can be stored in either packed or unpacked form. The best of each category we have tested is BFG [26], which requires the Hamiltonian in the strips it is constructed in, and PeIGS [27], which needs a packed lower triangle array. In practice, the BFG diagonalizer proved impractical for final Hamiltonian matrices larger than 3000. The PeIGS diagonalizer is limited to matrices of

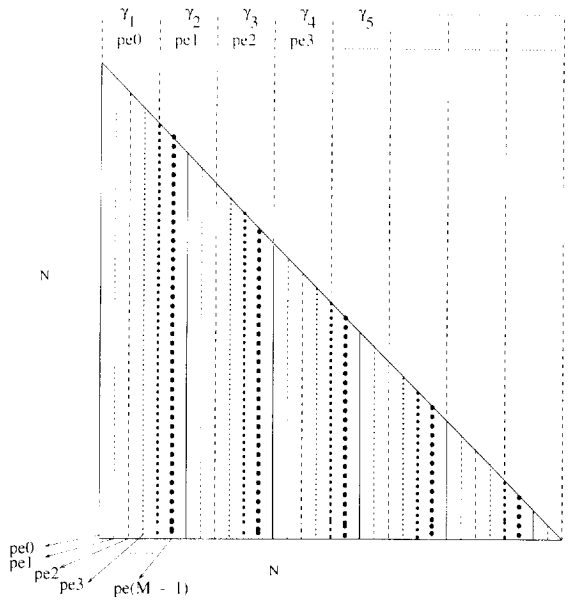

Fig. 1. Structure of Hamiltonian matrix used with the PeIGS diagonalizer. PDRV3D prepares the final Hamiltonian is strips based on the final DVR coordinate, γ, as indicated in the upper triangle. For the PeIGS diagonalizer this is redistributed, in lower triangular form, between the processors as indicated by the lower part of the figure.

dimension 9024 on 64 processors and 12160 on 128, for larger matrices the workspace required on each processor exceeded the available processor memory. To get a good load balance for PeIGS, the structure shown in Fig. 1 is used. A survey of available parallel diagonalizers has been conducted by Allan [29].

As standard real symmetric diagonalizers proved limiting in the size of calculation which could be attempted, we also explored the possibility of using an iterative diagonalization procedure. Such an approach has been advocated by a number of other workers [8,9,14,30], although, unlike many of the studies, we require eigenvectors as well as eigenvalues. To our knowledge there is only one parallel iterative eigensolver, PARPACK [32,31], generally available. In common with other iterative procedures it is designed to obtain a few eigenvalues of a large sparse matrix.

To diagonalize the Hamiltonian matrix, H, iteratively, as is usual, one needs to perform the matrix–vector product

$$y = Hx\,. \tag{2}$$

However, any zero elements of H need not be considered. In practice, the vector x is distributed over the processors and broadcast to the other processors. BLAS3 routine 'sgemv' is used to give y. This procedure is very efficient and requires little interprocessor communication.

There are two possible ways of utilizing an iterative diagonalizer. (a) One can take advantage of the natural sparseness of the DVR based Hamiltonian and diagonalize the uncontracted matrix. Such an approach has been used very successfully by Bramley and Carrington [8,30] and others [9] on sequential machines. It involves treating very sparse matrices of dimension 100 000 or larger. (b) It is possible to use the iterative diagonalizer to diagonalize the same matrix as we tested the real symmetric diagonalizers on. We tested both procedures.

One virtue of method (a) is that in its simplest form it is very simple to program. However, the symmetrization of the radial grids points employed by DVR3D presents a significant complication as it destroys much of the structural simplicity of the uncontracted Hamiltonian matrix. This simplicity is important for the efficiency of this algorithm [30]. For this reason we implemented this method without using the radial symmetry. In this case the size of the matrix is given as the product of the number of grid points $n_\alpha n_\beta n_\gamma$. Distributing the matrix as before over the angular grid, γ, there is a local memory balance between the number of eigenvectors required and the size of the matrix stored on each processor. The maximum size for which we performed calculations was for 100 eigenstates of a matrix of dimension 102400. These proved considerably more expensive to obtain than method (b).

Using the PARPACK to diagonalize the contracted Hamiltonian, method (b), has one immediate advantage over use of PeIGS. PARPACK has little local memory requirement meaning that, in practice, matrices of twice the size could be diagonalized.

If eigenvectors are required, as is always true for rotationally excited states, then it is necessary to transform the eigenvectors obtained from the final diagonalization back to vectors which give the amplitude of the wavefunction on the original, raw grid points. As only h ($< N^{3D}$) eigenvectors need to be transformed, it was necessary to redistribute the eigenvectors so that those required were spread equally between the pro-

cessors. This procedure was complicated by the unpredictable manner in which PeIGS distributes eigenvectors on the processors.

The transformation can be written

$$d^{ik}_{\alpha\beta\gamma} = \sum_{j=1}^{n^{2D}} C^{2D}_j(\alpha\beta : \gamma) C^{3D}_i(j,\gamma) . \quad (3)$$

The 2D vectors, $\underline{C}^{2D}$, were mapped on to processors using γ as they are generated during the Hamiltonian construction. The 3D vectors, $\underline{C}^{3D}$, were mapped by distributing vectors between processors. The multiplication was performed using MPI broadcast-gather routines and the results saved on disk. Since I/O is not parallel, two processors were used to perform I/O to avoid bottle-necks.

When considering states of rotational excitation J, it is necessary to repeat all the above steps for each k, the projection of J on the body-fixed z-axis, required. In general, if both even and odd Wang parity rotational states are to be calculated using the same first step vectors, then $J+2$ separate calculations are performed, one for $k = 0, 1, \ldots, J$, plus an extra $k = 1$ calculation as this must be treated as a special case [16].

3.2. Step-two: PROTLEV3R

The second step of the calculation introduces full rotation–vibration coupling by including the Coriolis operators [16], $\widehat{H}_{kk'}$, which are off-diagonal in k. The matrix construction step consists of a series of transformations of the form

$$\langle k, i|\widehat{H}_{kk'}|k', i'\rangle = \sum_{\alpha\beta\gamma\gamma'} d^{ik}_{\alpha\beta\gamma} H_{kk'}(\alpha\beta\gamma\gamma') d^{i'k'}_{\alpha\beta\gamma'} \quad (4)$$

using the vectors created in the first step. The Coriolis operators are diagonal in the radial DVR's, $(\alpha\beta)$, but not the angular DVR, γ. Full details are given elsewhere [13,16]. The transformation uses h ($< N^{3D}$) eigenvectors from each first step diagonalization [28]. In the present procedure it was assumed that h is the same for all k.

For symmetrized Radau coordinates, symmetry considerations dictate that the Cartesian axes of the system should be placed so one (taken as x here) bisects the angle θ and the z-axis is then in the plane of the molecule and perpendicular to x. This is the so-called bisector embedding [16]. In this embedding, the Coriolis coupling links blocks of the matrix labeled by k with those for with $k \pm 1$ and $k \pm 2$ [16]. One needs to consider $J + 1$ values of k.

Because a four-dimensional transformation, Eq. (3), is used to construct each matrix element, this step can be quite time consuming. It is therefore necessary to consider carefully how one might parallelize it. Goldfield and Gray [33] mapped the triatomic rearrangement reaction problem using k (or Ω) blocks. However, in our case 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 the processors a possibility. Alternatively, one can distribute over the stretching coordinates $(\alpha\beta)$ or the angle γ or the eigenvectors h. Parallelizing over $(\alpha\beta)$ is difficult because of the symmetrization. Conversely parallelizing over γ 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 γ. 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 vectors $\underline{d}^{ik}$ takes about 0.5 GB of memory, so these vectors must be distributed. This method leads to an excessive amount of inter-processor communication.

We therefore distribute the building of the Hamiltonian matrix over the M processors by placing a h/M bra and h/M ket vectors on each processor. Each processor thus builds rows (columns) of each block, see Eq. (3). This is done using the vectors in local memory for the bra and vectors from other processors for the ket. The resulting structure, see Fig. 2, is appropriate when the matrix is to be diagonalized iteratively. However, redistribution is necessary for the in-core PeIGS procedure into the form given by Fig. 1. The blocks computed contain all the nonzero off-diagonal elements of the Hamiltonian. However, it is also necessary to distribute the diagonal elements, which correspond to the eigenvalues generated in the first step in the fashion used for the off-diagonal blocks.

Fig. 2 also shows the structure of the Hamiltonian matrix: it is sparse and with a predictable structure. The choice of diagonalization procedure depends on a number of factors:

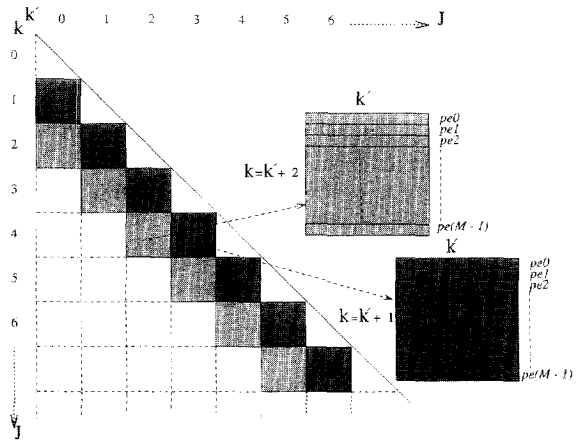

Fig. 2. Structure of second-step Hamiltonian constructed by PROTLEV3R. Only nonzero elements, those on the diagonal and shaded blocks, for the lower triangle are computed. The blocks are spread over the processors as indicated in the enlargements.

(i) The sparsity of Hamiltonian: the Hamiltonian matrix becomes increasingly sparse with increasing J.
(ii) The size of the Hamiltonian, $N = (J+1) \times h$, as the iterative diagonalizer requires significantly less, by a factor $\sim J/2$, memory.
(iii) The number of eigenvalues required. The iterative diagonalizer is most efficient for obtaining a small proportion of the eigenvalues.

Again we tested the PeIGS [27] in-core and PARPACK [31] iterative diagonalizer. For the sparse iterative eigensolver, the size of the Hamiltonian is not that important as the many zero elements of H need not be considered when multiplying by the image vector, Eq. (2). In practice, a copy of the image vector x is sent to every processor where it is multiplied with the portion of matrix stored locally (and its transpose [34]).

For an in-core diagonalizer, the size of the Hamiltonian matrix is important since the full matrix is needed. The diagonalization arguments are similar to those discussed in Section 3.1.

4. Jacobi coordinates

For a triatomic, Jacobi or atom–diatom scattering coordinates comprise an atom–atom distance, r, the distance from the centre-of-mass of these two atoms to the third atom, R, and the angle between $\underline{r}$ and $\underline{R}$, θ. As each coordinate is treated separately, there are six possible coordinate orderings. DVR3D implemented four of these options. Here we pursue a slightly different strategy whereby there is no separate treatment (diagonalization and truncation) of the first coordinate, instead the first two coordinates are treated together. In this case it is only necessary to decide which coordinate to treat last. Here we will consider only R last as this is appropriate for H_3^+ [22]. Implementing the other orders would be relatively straightforward using the strategy given below. Indeed we have recently implemented a θ last ordering to study the molecule HCP.

4.1. Step-one: PDVR3DJ

The procedure is similar to that described in Section 3.1 except that here it is the radial R coordinate, rather than the angle θ that is distributed over the processors. Furthermore, in the R coordinate the grid can tuned to the system [13], so there is no reason to reject particular points, thus all DVR points in R are taken as active.

4.2. Step-two: PROTLEV3J

In this code (and ROTLEV3 [13]), the z-axis is taken to be parallel to one of the two radial coordinates, which is a program option. This leads to a simpler Hamiltonian in which the Coriolis coupling only couples neighbouring k blocks, i.e. k with $k \pm 1$. Furthermore, these off-diagonal elements have a particularly simple structure if the angular coordinate is first transformed to finite basis representation (FBR) in associated Legendre polynomials, $|j, k\rangle$ [28].

The DVR to FBR transformation can be written

$$f^{ik}_{\alpha\beta j}|j,k\rangle = \sum_{\gamma} T^{\gamma}_{j} d^{ik}_{\alpha\beta\gamma}. \tag{5}$$

A copy of the transformation matrix, T^{γ}_{j}, is stored on every processor; as the vectors are distributed between the processors, see above, they can be transformed without any need for inter-processor communication. The matrix construction step now consists of a series of transformations of the form

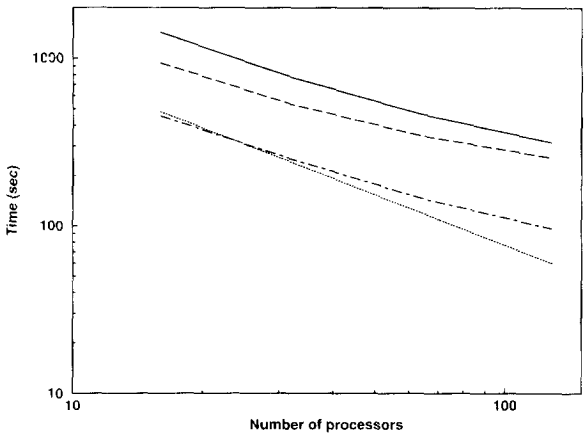

Fig. 3. Performance of PDVR3DRJ on the Cray-T3D in seconds for a water calculation with $J = 0$ and a final Hamiltonian of dimension $N = 3200$. Curves from the right-hand side bottom up are (a) time for building the Hamiltonian, (b) time for diagonalizing the Hamiltonian computing eigenvectors only, (c) time for diagonalizing the Hamiltonian using PeIGS [27] and saving 1000 eigenvalues and eigenvectors, and (d) total time for full calculation.

$$\langle k, i|\widehat{H}_{kk'}|k', i'\rangle = \sum_{\alpha\beta j} f^{ik}_{\alpha\beta j}\langle j, k|H_{kk'}(\alpha\beta)|j, k'\rangle_\theta \times f^{i'k'}_{\alpha\beta j}, \quad (6)$$

where the subscript θ means integration only over the angular coordinate. These are performed in a similar fashion to PROTLEV3R but, as they are only three-dimensional, are considerably less time consuming. Furthermore, the final matrix obtained by PROTLEV3J is sparser than that produced by PROTLEV3R because of the absence of the $k' = k \pm 2$ blocks.

5. Performance

Only performance for the PDVR3DR and PROTLEV3R will be discussed. PDVR3DJ behaves essentially the same as PDVR3DR and PROTLEV3J is simpler and quicker than PROTLEV3R.

Fig. 3 illustrates the scalability of the vibrations program PDVR3DR. It is well known from scalar implementations that this calculation is dominated by the time of the final diagonalization. Our Hamiltonian construction procedure shows near-perfect scalability, because the algorithm above minimizes data transfer. The full matrix diagonalizer – results for PeIGS are given – requires considerable interprocessor communication so some degradation with increasing numbers of processors is to be expected. 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 [35].

Times for the transformation step are not shown in Fig. 3 but this step adds about 10% to the CPU requirement. The test results are indeed satisfactory and we are well placed to take advantage of any future improvements in parallel matrix diagonalization algorithms.

As the diagonalization dominates the calculation, an important comparison is between in-core results obtained with PeIGS and iterative diagonalization with PARPACK. Obviously the results of such a comparison depend somewhat on what is required, particularly in terms of eigenstates. The unsymmetrized, uncontracted Hamiltonian of method (a) above with $n_\alpha = n_\beta = 40$ grid points and $n_\gamma = 64$ 'active' points gives a matrix of dimension 102400. Using PARPACK to obtain the lowest 100 eigenvectors of this matrix on 64 CRAY T3D processors took 771 minutes. Using PARPACK for the same grid but a 7680-dimensional, contracted final Hamiltonian matrix, method (b), to obtain the same 100 eigenvectors took only 84 minutes. In both cases Hamiltonian construction times are included. Method (a) is implicitly more accurate than method (b) but in practice the eigenvalues agreed to nine figures, something of a triumph for the DVR-based sequential diagonalization and truncation procedure. Construction and diagonalization of the 7680 dimension problem using PeIGS took 847 minutes. Although PeIGS automatically gives all eigenvectors, it would appear that using PARPACK to diagonalize the contracted Hamiltonian is the most promising approach available at present. This method is both quickest and allows larger matrices to be diagonalized.

Fig. 4 shows the equivalent behaviour for PROTLEV3R. For a matrix of only $N = 1000$ only in-core diagonalization using PeIGS is considered as it is rapid. In this case matrix construction is the rate limiting step as it both requires both inter-processor communication and I/O. Since the I/O gates are not dedicated to us, there is some arbitrariness in the fig-

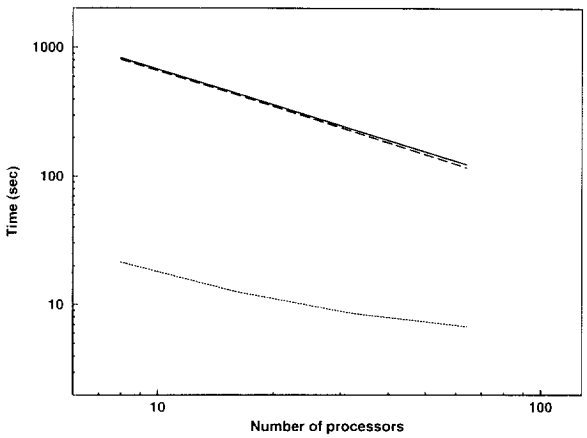

Fig. 4. Performance of PROTLEV3R on the Cray-T3D in seconds for a water calculation with $J = 5$ and a final Hamiltonian of dimension $N = 1000$. Curves from bottom up are (a) time for diagonalizing the Hamiltonian computing eigenvectors, (b) time for building the Hamiltonian, (c) total time for full calculation.

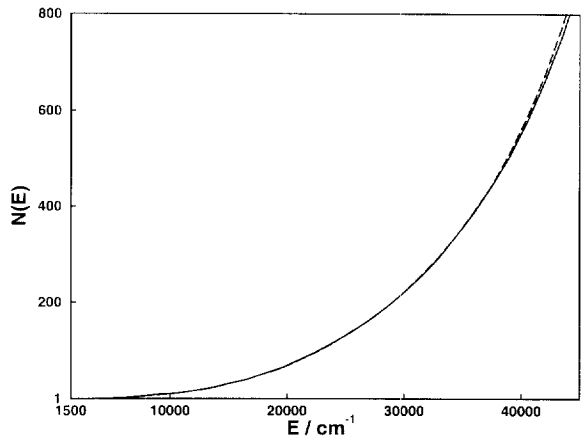

Fig. 5. Number, $N(E)$, of energy levels lying below energy E up to dissociation. Results are for $J = 0$ even and where obtained using the potential of Varandas [36], lower curve, and Ho and Rabitz [37], upper curve.

ures shown for this step. However, we have managed to demonstrate satisfactory scaling. We note that for PROTLEV3J, the very much simpler Hamiltonian construction means that diagonalization times usually predominate.

One aim of parallelizing the DVR3D program suite is to help us study molecules such H_3^+ and water at their dissociation limits. Recently two new potential energy surfaces for water have been published by Varandas [36], and Ho and Rabitz [37]. These aim to represent the potential up to the dissociation limit and beyond. A further surface is being calculated by Dobbyn and Knowles within the ChemReact HPCI Consortium using a version of the MOLPRO code [38] which has been parallelized.

Fig. 5 gives results on all the bound vibrational states of water using the Varandas and Rabitz surfaces. These calculations were performed using symmetrized Radau coordinates with 40 DVR points in each radial coordinate, 64 active angular DVR points out of 69 and $n^{2D} = 100$ giving a final Hamiltonian size of 6400. Such a calculation takes only 40 minutes on 64 T3D processors. A calculation using PARPACK on a contracted Hamiltonian matrix with dimension 12800 took a similar amount of time. It suggested that these results converged to about $\sim 0.5\ \mathrm{cm}^{-1}$ all the way to dissociation, although some further tests using different grid sizes are required to confirm this results. Full results of this study will be presented elsewhere.

6. Conclusion

We have developed parallel programs for treating the vibration–rotation motion of three-atom systems using either Jacobi or Radau coordinates. These programs are based on the published DVR3D program suite [13] which is designed for computers with traditional architectures. Significant algorithmic changes were required, in particular to reduce I/O in the original programs. The new suite shows good scalability and can be used for more challenging calculations. However, diagonalization remains a bottleneck in these calculations. Tests of presently available software favour the use of iterative diagonalization procedures, although further significant improvements in real symmetric matrix diagonalizers could alter this view.

Acknowledgements

This work was performed as part of the ChemReact High Performance Computing Initiative (HPCI) Consortium. We thank Prof. Antonio Varandas for advice on implementing his potential, Dr. Stephen Gray for supplying results of his water calculation and mem-

bers of HPCI Centre at Daresbury Laboratory for their help.

References

[1] O.L. Polyansky, N.F. Zobov, S. Viti, J. Tennyson, P.F. Bernath, L. Wallace, Science 277 (1997) 346.
[2] H. Partridge, D.W. Schwenke, J. Chem. Phys. 106 (1997) 4618.
[3] O.L. Polyansky, N.F. Zobov, S. Viti, J. Tennyson, P.F. Bernath, L. Wallace, J. Mol. Spectrosc., in press.
[4] A. Carrington, R.A. Kennedy, J. Chem. Phys. 81 (1984) 91.
[5] A. Carrington, I.R. McNab, Y.D. West J. Chem. Phys. 98 (1993) 1073.
[6] J.R. Henderson, J. Tennyson, Chem. Phys. Lett. 173 (1990) 133.
[7] J.R. Henderson, J. Tennyson, B.T. Sutcliffe, J. Chem. Phys. 98 (1993) 7191.
[8] M.J. Bramley, J.W. Tromp, T. Carrington Jr, G.C. Corey, J. Chem. Phys. 100 (1994) 6175.
[9] V.A. Mandelshtam, H.S. Taylor, J. Chem. Soc. Faraday Trans. 93 (1997) 847.
[10] J.R. Henderson, J. Tennyson, Mol. Phys. 89 (1996) 953.
[11] L. Neale, J. Tennyson, Astrophys. J. 454 (1995) L169.
[12] Z. Bacic, J. C. Light, Ann. Rev. Phys. Chem. 40 (1989) 469.
[13] J. Tennyson, J.R. Henderson, N.G. Fulton, Comput. Phys. Commun. 86 (1995) 175.
[14] X.T. Wu, E.F. Hayes, J. Chem. Phys. 107 (1997) 2705.
[15] J. Tennyson, B.T. Sutcliffe, Mol. Phys. 58 (1986) 1067.
[16] J. Tennyson, B.T. Sutcliffe, Int. J. Quantum Chem. 42 (1992) 941.
[17] S. Viti, J. Tennyson, unpublished.
[18] Parallel Virtual Machine; the PVM3 Users' Guide, Reference Manual is available from `netlib2.cs.utk.edu/pvm3/ug.ps` via anonymous ftp.
[19] Message Passing Interface; the MPI Standard is available from `netlib2.cs.utk.edu` by anonymous ftp. Note: it is a very large document!
[20] B.R. Johnson, W.P. Reinhardt, J. Chem. Phys. 85 (1986) 4538.
[21] J. Tennyson, J.R. Henderson, J. Chem. Phys. 91 (1989) 3815.
[22] J.R. Henderson, C.R. Le Sueur, S.G. Pavett, J. Tennyson, Comput. Phys. Commun. 74 (1993) 193.
[23] NAG Fortran Library Manual, Mark 17, Vol. 4 (1996).
[24] Scalapack Users' Guide is available in html form from WWW URL `http://netlib.org/scalapack/`.
[25] B.A. Henderson, E. Jessup, C. Smith, a parallel eigensolver for dense symmetric matrices, available via anonymous ftp from `//ftp.cs.sandia.gov/pub/papers/bahendr/eigen.ps.Z`.
[26] I.J. Bush, Block factored one-sided Jacobi routine, following: R.J.Littlefield, K.J.Maschhoff, Investigating the performance of parallel eigensolvers for large processor counts, Theor. Chim. Acta 84 (1993) 457–73.
[27] G. Fann, D. Elwood, R.J. Littlefield, PeIGS Parallel Eigensolver System, User Manual available via anonymous ftp from `pnl.gov`.
[28] J. Tennyson, J. Chem. Phys. 98 (1993) 9658.
[29] R.J. Allan, I.J. Bush, Parallel Application Software on High Performance Computers: Parallel Diagonalization Routines, ed. 3, Daresbury Laboratory HPCI Centre, 22/8/96, available on WWW URL `http://www.dci.clrc.ac.uk/Publications`.
[30] M.J. Bramley, T. Carrington Jr, J. Chem. Phys. 99 (1993) 8519.
[31] R. Lehoucq, K. Maschhoff, D. Sorensen, C. Yang, PARPACK is available from `ftp://ftp.caam.rice.edu/pub/people/kristyn`.
[32] K. Maschhoff, D. Sorensen, A portable implementation of ARPACK for distributed memory parallel architectures, Preliminary proceedings, Copper Mountain Conference on Iterative Methods (1996).
[33] E.M. Goldberg, S.K. Gray, Comput. Phys. Commun. 98 (1996) 1.
[34] B.T. Sutcliffe, J. Tennyson, S. Miller, Theor. Chim. Acta 72 (1987) 265.
[35] G. Fann, R.J. Littlefield, Parallel inverse iteration with reorthogonalisation, in: Proc. 6th SIAM conference on Parallel Processing for Scientific Computing (SIAM, Philadelphia, PA, 1993) pp. 409–413.
[36] A.J.C. Varandas, J. Chem. Phys. 105 (1996) 9.
[37] T.-S. Ho, T. Hollebeck, H. Rabitz, L.B. Harding, G. Schatz, J. Chem. Phys. 105 (1996) 10472.
[38] A. Dobbyn, P.J. Knowles, Parallel MOLPRO version 1.0, in preparation.