JOURNAL OF COMPUTATIONAL PHYSICS 113, 201–214 (1994)

# Scalable Algorithms for Three-Dimensional Reactive Scattering: Evaluation of a New Algorithm for Obtaining Surface Functions

PHIL PENDERGAST, ZAREH DARAKJIAN, AND EDWARD F. HAYES

*Department of Chemistry, Ohio State University, Columbus, Ohio 43210*

AND

DANNY C. SORENSEN

*Department of Mathematical Sciences, Rice University, Houston, Texas 77251*

Received August 4, 1993; revised December 10, 1993

Implementation of the adiabatically adjusting, principal axis hyperspherical coordinate (APH) approach of Parker and Pack for three-dimensional reactive scattering requires solution of a series of two-dimensional (2D) surface eigenproblems. A new algorithm is presented that takes the discrete variable representation (DVR) of the surface Hamiltonian and transforms it implicitly to the sequential diagonalization truncation (SDT) representation of Light and coworkers. This implicit transformation step, when combined with the implicit restarted Lanczos method of Sorensen with Chebyshev preconditioning, can be used to obtain accurate solutions to the large-dimensionality surface eigenproblems encountered in three-dimensional reactive scattering. Timing results are presented and comparisons made with the previously employed SDT-DVR approach for these 2D eigenproblems. The new algorithm is faster than the SDT-DVR algorithm currently in use by about a factor ranging from 2.6 to 4.5 for both scalar and vector implementations. This algorithm also requires much less memory for the same order DVR Hamiltonian than previous approaches. This permits solution of larger eigenproblems without resorting to external storage. Strategies for implementing this algorithm on parallel architecture machines are presented. © 1994 Academic Press, Inc.

## I. INTRODUCTION

Over the past few years it has become possible to determine accurate state-to-state reaction cross sections for three-atom systems with modest numbers of energetically open initial-and-final quantum states [1–11]. While such efforts have achieved a high level of accuracy for a few simple systems, there remain major impediments to the extension of these methods to the vast majority of interesting chemical reactions. On a qualitative level, the basic theoretical and computational issues are easily characterized. First, the complexity and computational intensity of current methods, even for simple three-atom systems, challenge current generations of high-performance computers. Second, as one moves beyond these simple chemical reactions, the complexity of the calculations increases significantly because the number of open initial-and-final quantum states increases by one to two orders of magnitude. For this reason, such problems are, at this moment, beyond the realm of consideration.

Within the last few years, the advances registered in high performance computing technology have revolutionized the approaches adopted for the treatment of massively compute-intensive problems. These advances in supercomputing are being accompanied by significant breakthroughs in numerical methods, holding the promise of realizing substantial savings in compute times for the problems that we can solve today and importantly, extending the range of chemical reactions for which one can obtain accurate quantum cross sections and angular distributions.

The scattering approach that we are seeking to enhance—and with which we have had previous experience [12]—is the adiabatically adjusting, principal axis hyperspherical (APH) coordinate approach of Parker and Pack [1b] in conjunction with the log-derivative propagation method of Johnson [13].

Below, we present a brief description of the computational steps necessary to obtain accurate solutions to the three-dimensional (3D) quantum mechanical scattering problem within the context of the APH formalism. The idea is to include enough detail to explain the context for the present study while relying on earlier descriptions [1] to supply complete details. The goal of this paper is to establish the viability of a new approach for determining the 2D surface functions that are needed in the APH formulation. An additional motivation is to develop scalable parallel algorithms for three- and more-atom reactive scattering codes. In this effort, we are guided by the pioneering work of Aron Kuppermann and coworkers [14] who have been developing codes for distributed-memory parallel architecture machines for several years. Due to differences in the hyperspherical formulations developed by Parker and Pack, the surface Hamiltonian used here is not identical to that generated by Kuppermann *et al.* in their studies. Moreover, the strategy for obtaining the solutions to the 2D eigenproblem (surface functions) is fundamentally different as will be evident from the discussion below.

In the APH coordinate approach the total wave function is expanded in a basis of sector adiabatic surface functions, which in the notation of Ref. [1b, see Eq. (123)] is

$$\Psi^{JMpn} = 4 \sum_{t,\Lambda} \rho^{-5/2} \psi^{Jpn}_{t,\Lambda}(\rho)\, \Phi^{Jp}_{t,\Lambda}(\theta, \chi; \rho_\zeta)\, \hat{D}^{Jp}_{\Lambda M}(\alpha_Q, \beta_Q, \gamma_Q), \tag{1}$$

wherein the surface functions, $\Phi$, are bound state eigenfunctions of the 2D Hamiltonian,

$$\mathscr{H}(\theta, \chi; \rho_\zeta)\, \Phi_t(\theta, \chi; \rho_\zeta) = \mathscr{E}_t(\rho_\zeta)\, \Phi_t(\theta, \chi; \rho_\zeta), \tag{2}$$

in which $\theta$ and $\chi$ are the two hyperspherical angles and $\rho$ is the scattering coordinate. Again the notation of Ref. [1b, see Eq. (164)] has been followed, but the $J$, $p$, and $\Lambda$ indices have been suppressed in Eq. (2) and below in order to simplify the notation.

The 2D surface Hamiltonian (Eq. (2)) is the sum of three kinetic energy operators and a potential energy term with the definition $\vartheta = 2\theta$, the kinetic energy terms are

$$\hat{T}^\chi = -\frac{\hbar^2}{2\mu} \frac{1}{\rho^2 \sin^2(\vartheta/2)} \frac{\partial^2}{\partial \chi^2}, \tag{3}$$

$$\hat{T}^\theta = -\frac{\hbar^2}{2\mu} \frac{16}{\rho^2 \sin\vartheta} \frac{\partial}{\partial \vartheta} \sin\vartheta \frac{\partial}{\partial \vartheta}, \tag{4}$$

and a diagonal centrifugal term,

$$\hat{T}^c = \frac{\hbar^2 \Lambda^2}{\mu \rho^2 (1 - \sin\vartheta)}, \tag{5}$$

where $\Lambda$ is the projection of the angular momentum on the body fixed axis $z$. The potential energy operator is

$$\tilde{V}(\rho_\zeta, \vartheta, \chi) = V(\rho_\zeta, \vartheta, \chi) + \frac{15\hbar^2}{8\mu\rho_\zeta^2}, \tag{6}$$

where $\mu$ is the reduced mass of the three-particle system and $V(\rho_\zeta, \vartheta, \chi)$ is the interparticle potential. As in Ref. [16], we have followed the lead of Launay and Le Dourneuf [11c, 11d] and defined the 2D surface Hamiltonian so that it is independent of the total angular momentum, $J$.

In some earlier work the complexity of the chemical reaction that could be studied was ultimately determined by the efficiency and accuracy of the method for obtaining the surface functions, $\Phi$. The efficiency of these calculations is important both for production scattering calculations and for the setup stages that are needed to make certain that the number of surface functions and their accuracy are sufficient for production studies. Consequently, this particular aspect has been the focus of considerable effort. For the Parker–Pack APH formulation, three basic approaches have been developed: the finite element method (FEM) [15]; the analytic basis function method (ABM) [16]; and the DVR approach [17]. The DVR has been shown to be both accurate and efficient in the study of three-dimensional quantum reactive scattering problems as well as in the vibrational analysis of floppy molecules and molecules in highly-excited vibrational states. The results from these studies indicate that the DVR approach is particularly efficient for small and intermediate values of the hyperradius, $\rho$. In this paper, we present results from a new adaption and implementation of the DVR approach in conjunction with an efficient diagonalization scheme developed by Sorensen [18]. This new approach forms the basis for a scalable parallel computational algorithm both at the highest sector-level of granularity and the intermediate eigenvalue/eigenvector-level of granularity.

In focusing on the DVR approach, it is not our intention to suggest that the DVR is the most appropriate method to be used for all $\rho$ values. In fact, the ABM method seems to have greater utility for large $\rho$ values [16], and it is anticipated that for most problems there will be some turnover value of $\rho$, where the bases should change from the DVR to the ABM.

In the *first stage* of the calculations, the two-dimensional Schrödinger equation, Eq. (2), is expressed in a preselected representation, and, subsequently, the eigenvalues of the Hamiltonian and their associated eigenvectors are found. These are needed in order to generate and solve the APH coupled-channel (CC) equations that arise upon substituting the surface expansion of the total wave function into the full Schrödinger equation. The large-grain parallelism of the computations for this stage arises from the fact that the computations for each sector, $\zeta$, may be carried out independently. By assigning each sector to a different processor, or to a set of processors, one may achieve computational speedups that scale linearly with the number of processors. Moreover, the power of the new algorithm that we present here is that it is also scalable at the intermediate eigenvalue/eigenvector-level of granularity, where the use of clusters of processors for each sector can also privide significant speedups.

In the *second stage*, the potential coupling matrix elements are determined for each sector using the sector surface functions determined in the *first stage*. While there are several coupling terms that must be calculated, the term containing the interaction potential has the form

$$\mathscr{V}_{t,t'} = \langle \Phi_t(\theta, \chi; \rho_\zeta) | \Delta V(\rho_\zeta, \rho, \vartheta, \chi) | \Phi_{t'}(\theta, \chi; \rho_\zeta) \rangle, \quad (7)$$

where

$$\Delta V(\rho_\zeta, \rho, \vartheta, \chi) = V(\rho, \vartheta, \chi) - \frac{\rho_\zeta^2}{\rho^2} V(\rho_\zeta, \vartheta, \chi) \quad (8)$$

(see Eq. (168) in Ref. [1b]). At this stage we also determine the sector-to-sector overlap matrix elements,

$$O_{tt'} = \langle \Phi_t(\rho_\zeta) | \Phi_{t'}(\rho_{\zeta+1}) \rangle. \quad (9)$$

As in the *first stage*, one may achieve significant enhancements by assigning each sector and the adjacent sector-to-sector overlap matrices to a different processor or set of processors. Once again the large-grain sector parallelism leads to algorithms that are inherently scalable as the number of processors or clusters of processors is increased.

In the *third stage*, the log-derivative matrix, which is directly related to the wave-function $\psi(\rho)$ in Eq. (1), is propagated starting in the exchange region (small $\rho$) and terminating in the asymptotic region (large $\rho$). The propagation phase of the calculations is implemented using the LOGDER/VIVAS program [13, 19]. One simple scheme for achieving high levels of parallelism for this stage is to assign different scattering energies to different processors or groups of processors. For this approach one needs multiple access to all the coupling-matrix elements for all sectors. Another approach is to assign a sector to a processor or cluster of processors, as in stages 1 and 2, and then run the scattering energies through the sector sequence in a pipeline fashion. If the number of energies and total angular momenta, $J$, greatly exceed the number of sectors, this may be the method of choice because there will be less interprocessor communication between stages 2 and 3 and, importantly, multiple access to all the coupling matrix elements is not required. Parker has worked with this stage in a distributed processor environment [20].

In the *fourth stage*, the asymptotic boundary conditions are applied in the usual way, and the elements of the scattering matrix, S, are determined. Kuppermann [14b] has pointed out previously that this stage is so rapid that a single processor is probably all that will be needed to efficiently determine the scattering matrix elements.

The four computational stages described above present varying degrees of computational demands. The *first stage* (calculation of the surface functions and their associated eigenvalues) and the *third stage* (the propagation of the logarithmic derivative of the wave-function) both involve substantial computer resources. In this paper we will be reporting a scalable algorithm for the first stage that also turns out to be more efficient on normal scalar and vector computers than the current method of choice. While the application and numerical results presented here all deal with three-atom scattering, the approach presented can be extended to four- and higher-atom scattering or to bound-state quantum problems. For these higher dimensional problems the efficiency of this general approach may be even more significant.

## II. THE SEQUENTIAL DIAGONALIZATION TRUNCATION METHOD

In the following we discuss the solution of Eq. (2) as it has been implemented in the Parker–Pack 3D reactive scattering system. The method utilizes the sequential diagonalization truncation (SDT) scheme of Light and coworkers [21]. This method results in the reduction of the order of the 2D surface Hamiltonian $\mathscr{H}^{\mathrm{DVR}}$ through a series of steps involving diagonalization of relatively small matrices, the truncation of unwanted solutions, and the eventual diagonalization of a Hamiltonian matrix with significantly reduced order over the original DVR Hamiltonian [21, 22].

The finite basis representation upon which the DVR is based consists of a product of normalized Legendre polynomials in $\cos\vartheta$ and symmetry-adapted normalized trigonometric functions ($\Pi_m(\chi) = \cos 2(m-1)\chi$, $m = 1, 2, \ldots, m_{\max}$) for the $\theta$ and $\chi$ degrees of freedom, respectively. The surface function expression in the direct product FBR is [21, 22]

$$\Phi_t = \sum_{l=0}^{l_{\max}} \sum_{m=1}^{m_{\max}} k_{lm}^t \hat{P}_l(\cos\vartheta)\, \Pi_m(\chi). \quad (10)$$

Within the DVR the coupled equations are designated by quadrature points of the FBR and not by the FBR basis. As noted by Light and coworkers [21], both bases, the $N$-point quadrature and the set of $N$ FBR basis functions, are isomorphic. The DVR and FBR Hamiltonians are related through a unitary transformation,

$$\mathscr{H}^{\mathrm{DVR}} = \mathbf{U}^{\mathrm{T}} \mathbf{H}^{\mathrm{FBR}} \mathbf{U}, \quad (11)$$

where U is the FBR-to-DVR transformation matrix. Defining a diagonal $\theta$ factor as

$$f_{\theta,ij}^{\mathrm{DVR}} = \left[\sin\left(\frac{\vartheta_{ij}}{2}\right)\right]^{-2} \delta_{ij}, \quad (12)$$

the surface Hamilton in the DVR can be written in tensor-product form [22] as

$$\mathscr{H}^{\mathrm{DVR}} = \mathbf{h}_\theta^{\mathrm{DVR}} \otimes \mathbf{I}_\chi + \mathbf{f}_\theta^{\mathrm{DVR}} \otimes \mathbf{h}_\chi^{\mathrm{DVR}} + \tilde{\mathbf{V}}^{\mathrm{DVR}}, \quad (13)$$

where $\mathbf{h}_\theta$ and $\mathbf{h}_\chi$ are the kinetic energy operators for the $\theta$ and $\chi$ degrees of freedom associated with $\hat{T}^\theta$ and $\hat{T}^\chi \sin^2(\vartheta/2)$, respectively (see Eqs. (3), (4) and where $\tilde{\mathbf{V}}^{\mathrm{DVR}}$ is diagonal. Expressions for $\mathbf{h}_\theta^{\mathrm{DVR}}$ and $\mathbf{h}_\chi^{\mathrm{DVR}}$ are given in Appendix A.

Choosing a DVR for the matrix representation of the Hamiltonian significantly reduces the computational labor associated with the calculation of matrix elements since the DVR transfers the coupling from the potential operator to the kinetic energy operator. In the FBR the number of potential matrix elements that need to be determined is equal to the square of $n_\chi^{\mathrm{FBR}} \times n_\theta^{\mathrm{FBR}}$ for each energy of interest, where $n_\chi^{\mathrm{FBR}}$ and $n_\theta^{\mathrm{FBR}}$ are the FBR expansion sizes for the $\chi$ and $\theta$ bases, respectively. The potential matrix is diagonal in the DVR and the elements are equal to the value of the potential energy at the points labled by specific values for a $\theta$ and $\chi$ pair and the value of $\rho_\zeta$.

The matrix of the Hamiltonian (Eq. (13) has a regular and sparse structure. The matrix may be written as

$$\mathscr{H}^{\mathrm{DVR}} = \mathscr{B} + \mathscr{K} \tag{14}$$

with

$$\mathscr{B} = \mathscr{B}_{ii} (i = 1, ..., n_\theta), \tag{15}$$

where each $\mathscr{B}_{ii}$ is an $n_\chi$ by $n_\chi$ block diagonal submatrix,

$$\mathscr{B} = \begin{pmatrix} \mathscr{B}_{11} & 0 & \cdots & 0 \\ 0 & \mathscr{B}_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathscr{B}_{n_\theta n_\theta} \end{pmatrix},$$

the elements of which are the second and third terms in Eq. (13), where the potential energy term is augmented by the total angular momentum term if $\Lambda$ is different from zero, viz.,

$$\tilde{V}^\Lambda(\rho_\zeta, \vartheta, \chi) = V(\rho_\zeta, \vartheta, \chi) + \frac{15\hbar^2}{8\mu\rho_\zeta^2} + \frac{\hbar^2\Lambda^2}{\mu\rho_\zeta^2(1 - \sin\vartheta)}. \tag{16}$$

The first term in Eq. (13) is the $\theta - \chi$ coupling portion of $\mathscr{H}^{\mathrm{DVR}}$ and may be expressed as

$$\mathscr{K} = \mathscr{K}_{mn} \; (m, n = 1, ..., n_\theta), \tag{17}$$

where $\mathscr{K}_{mn}$ is an $n_\chi$ by $n_\chi$ submatrix with diagonal elements only. Pictorially, $\mathscr{K}$ may be represented as

$$\mathscr{K} = \begin{pmatrix} \mathscr{K}_{11} & \mathscr{K}_{12} & \mathscr{K}_{13} & \cdots & \mathscr{K}_{1n_\theta} \\ \mathscr{K}_{21} & \mathscr{K}_{22} & \mathscr{K}_{23} & \cdots & \mathscr{K}_{2n_\theta} \\ \mathscr{K}_{31} & \mathscr{K}_{32} & \mathscr{K}_{33} & \cdots & \mathscr{K}_{3n_\theta} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathscr{K}_{n_\theta 1} & \mathscr{K}_{n_\theta 2} & \mathscr{K}_{n_\theta 3} & \cdots & \mathscr{K}_{n_\theta n_\theta} \end{pmatrix}.$$

A unique feature of the $\mathscr{K}$ matrix is that for each $\mathscr{K}_{ij}$, all the diagonal elements are identical for the entire subblock $ij$, or $\mathscr{K}_{ij} = \alpha_{ij}\mathbf{1}$, where $\alpha_{ij}$ is a scalar and $\mathbf{1}$ is the unit matrix. The elements of these matrices account for the coupling between the $\chi$ and $\theta$ degrees of freedom.

The first step in the implementation of the SDT method defines a suitable contracted representation to which the 2D surface Hamiltonian can be transformed. This representation is chosen as that in which the block diagonal matrices $\mathscr{B}_{ii}$ have a diagonal form. Prior to their diagonalization in the SDT approach, the submatrices $\mathscr{B}_{ii}$ have a diagonal form. Prior to their diagonalization in the SDT approach, the submatrices $\mathscr{B}_{ii}$ are reduced in order by discarding those rows and columns corresponding to the potential values (Eq. 16) that have a magnitude above a preset cutoff potential. This "filtering" has the effect of excluding those regions of the potential surface that are inaccessible at the scattering energies of interest. The matrix $\mathbf{Q}$ is defined such that

$$\mathbf{Q}^{\mathrm{T}}\mathscr{B}\mathbf{Q} = \mathbf{\Upsilon}, \tag{18}$$

where $\mathbf{\Upsilon}$ is a diagonal matrix composed of the series of submatrices $\Upsilon_{ii}$,

$$\mathbf{\Upsilon} = \Upsilon_{ii} \; (i = 1, ..., n_\theta), \tag{19}$$

$$\mathbf{\Upsilon} = \begin{pmatrix} \Upsilon_{11} & 0 & 0 & \cdots & 0 \\ 0 & \Upsilon_{22} & 0 & \cdots & 0 \\ 0 & 0 & \Upsilon_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \Upsilon_{n_\theta n_\theta} \end{pmatrix}.$$

$\mathbf{Q}$ is actually composed of a series of diagonal blocks, again with order $\leqslant n_\chi$, viz.,

$$\mathbf{Q} = Q_{ii} \quad (i = 1, ..., n_\theta), \tag{20}$$

with $Q_{ij} \equiv 0$ for $i \neq j$ and, pictorially, $\mathbf{Q}$ looks exactly like $\mathbf{B}$. It follows that

$$Q_{ii}^{\mathrm{T}} B_{ii} Q_{ii} = \Upsilon_{ii}. \tag{21}$$

The truncated representation is obtained by retaining only those parts of the coefficient matrices, $Q_{ii}$, the associated eigenvalues of which are less than a predetermined cutoff value, *ecut*. This defines a set of rectangular matrices $\tilde{Q}_{ii}$.

In the third step, the Hamiltonian matrix $\mathscr{H}^{\mathrm{DVR}}$ is transformed to the representation defined by the truncated $\tilde{\mathbf{Q}}$. It is this contraction that is responsible for the substantial reduction in the dimension of the Hamiltonian matrix,

$$\tilde{\mathscr{H}}^{\mathrm{DVR}} = \tilde{\mathbf{Q}}^{\mathrm{T}}\mathscr{H}^{\mathrm{DVR}}\tilde{\mathbf{Q}}. \tag{22}$$

The resulting matrix, $\tilde{\mathscr{H}}^{\mathrm{DVR}}$, which is explicitly calculated, is no longer sparse and may contain more nonzero elements than the original unreduced $\mathscr{H}^{\mathrm{DVR}}$. Thus, in the final step, the matrix $\tilde{\mathscr{H}}^{\mathrm{DVR}}$ is diagonalized using standard dense matrix techniques. This contrasts with the method to be discussed in the next section.

## III. SCALABLE IMPLEMENTATION OF THE SDT METHOD

This method is based upon an analysis of the matrix formed as a result of the tensor operations in Eq. (13). The resulting 2D Hamiltonain consists of a series of diagonal blocks of order $n_\chi$ and a series of offdiagonal submatrices (also of order $n_\chi$) that contain only diagonal elements (see Fig. 1). Such a form allows for efficient formation of the matrix elements, low storage requirements for the matrix, and specialized diagonalization methods for the matrix that preserve the original sparse matrix $\mathscr{H}^{\mathrm{DVR}}$.

As in Eq. (14), the 2D Hamiltonian matrix $\mathscr{H}^{\mathrm{DVR}}$ is expressed as the sum of two matrices, viz.,

$$\mathscr{H}^{\mathrm{DVR}} = \mathbf{B} + \mathbf{K}; \tag{23}$$

however, here we have made a slightly different separation of terms. The matrix **B** still consists of diagonal blocks each of which has order $n_\chi$ and the matrix **K** is a series of off-diagonal rays whose submatrices have order $n_\chi$. One may express the matrix **B** as a series of submatrices $B_{ij}$ with

$$\mathbf{B} = B_{ij}\ (i, j = 1, ..., n_\theta), \tag{24}$$

where $B_{ij} \equiv 0$ for $i \neq j$ (compare Eq. 14), and

$$B_{ii} = \mathscr{B}_{ii} + \mathscr{K}_{ii}, \tag{25}$$

Similarly, the **K** matrix may be expressed as a series of submatrices $K_{ij}$,

$$\mathbf{K} = K_{ij}\ (i, j = 1, ..., n_\theta), \tag{26}$$

where $K_{ij} = \mathscr{K}_{ij}$, for $i \neq j$ and all $K_{ii} = 0$. As a result, one can then store **K** as an $n_\theta$ by $n_\theta$ array.

The matrix **Q** is again defined such that

$$\mathbf{Q}^{\mathrm{T}}\mathbf{B}\mathbf{Q} = \mathbf{\Upsilon}, \tag{27}$$

where $\mathbf{\Upsilon}$ is a diagonal matrix composed of the series of submatrices $\Upsilon_{ii}$,

$$\mathbf{\Upsilon} = \Upsilon_{ii}\ (i = 1, ..., n_\theta). \tag{28}$$

(see Eq. (19)). The matrix **Q** generated here is the same matrix as that generated in the previous SDT implementation. This follows from the fact that each $\mathscr{B}_{ii}$ differs from the corresponding $B_{ii}$ by a constant $(\alpha_{ii})$ along the diagonal (see Eq. (25)). While the eigenvalues will shift by a constant amount, the eigenvectors do not change. Each individual $\Upsilon_{ii}$ now contains the eigenvalues of the corresponding $\mathscr{B}_{ii}$ shifted by $\alpha_{ii}$.

As in the previous discussion of the SDT method, the matrix to be diagonalized is reduced by the transformation

$$\begin{aligned}\tilde{\mathbf{Q}}^{\mathrm{T}}\mathscr{H}^{\mathrm{DVR}}\tilde{\mathbf{Q}} &= \tilde{\mathbf{Q}}^{\mathrm{T}}\mathbf{B}\tilde{\mathbf{Q}} + \tilde{\mathbf{Q}}^{\mathrm{T}}\mathbf{K}\tilde{\mathbf{Q}} \\ &= \tilde{\mathbf{\Upsilon}} + \tilde{\mathbf{Q}}^{\mathrm{T}}\mathbf{K}\tilde{\mathbf{Q}},\end{aligned} \tag{29}$$

where again a specified energy cutoff, *ecut*, is used to cast out those parts of $Q_{ii}$ corresponding to eigen solutions above the cutoff. The truncated $\tilde{\mathbf{\Upsilon}}$ is, in general, no longer the regular matrix given in Eq. (28), but it is rather a series of $\tilde{\Upsilon}_{ii}$, where each submatrix can now have an order $0 \leqslant n'_\chi \leqslant n_\chi$. The transformation $\tilde{\mathbf{Q}}^{\mathrm{T}}\mathbf{K}\tilde{\mathbf{Q}}$ will proceed over the appropriate order of $\tilde{Q}_{ii}$ (corresponding to the order of $\tilde{\Upsilon}$). However, the diagonalization of

$$\mathbf{Q}^{\mathrm{T}}\mathscr{H}^{\mathrm{DVR}}\mathbf{Q} \approx \tilde{\mathbf{\Upsilon}} + \tilde{\mathbf{Q}}^{\mathrm{T}}\mathbf{K}\tilde{\mathbf{Q}} \tag{30}$$

can still be accomplished without explicitly forming the full matrix. In the previous formulation of the SDT method the

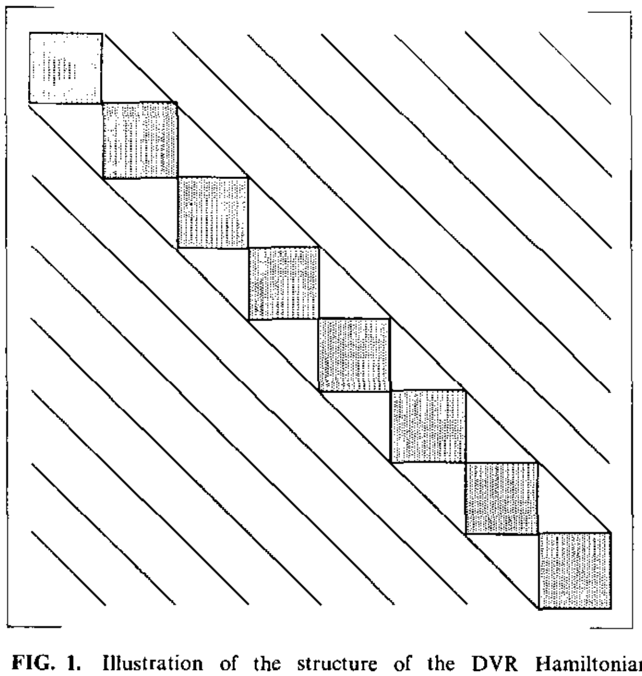

FIG. 1. Illustration of the structure of the DVR Hamiltonian, Eq. (13). The inner blocks, which are the dense $\chi$-space matrix elements, are depicted by shaded blocks of dimension $n_\chi \times n_\chi$. The off-diagonal blocks corresponding to different $\theta$-space DVR points are equal to a constant along the diagonal of each offdiagonal block and zero elsewhere. This is indicated by the connected diagonal lines. Appendix A provides an explanation for this simple and sparse structure of the DVR Hamiltonian.

transformation $\tilde{\mathbf{Q}}^{\mathrm{T}}\mathbf{K}\tilde{\mathbf{Q}}$ is completed explicitly, producing a dense matrix that must be diagonalized.

The implied transformation can be accomplished efficiently using the implicitly restarted Lanczos algorithm (IRLM) [18] to calculate the needed eigenvalues and eigenvectors. IRLM is a subset of a Fortran software package ARPACK, developed by P. A. Vu and D. C. Sorensen that handles a variety of standard and generalized eigenproblems. The IRLM requires a user-supplied routine that accomplishes the matrix-vector product $[\tilde{\mathbf{Y}} + \tilde{\mathbf{Q}}^{\mathrm{T}}\mathbf{K}\tilde{\mathbf{Q}}]\mathbf{x}$ but does not require explicit formation of the matrix to be diagonalized. Since the matrix-vector product represents the majority of the work done in the scheme, the efficiency of the matrix-vector product routine is central to the overall efficiency of the *Stage* 1 calculation.

## IV. PARALLELIZATION

As noted previously, the conceptual framework for the design of a parallel algorithm for *Stage* 1 of the overall 3D scattering calculation emanates from the potential for large-grain parallelization inherent in the division of the $\rho$-space into many sectors and the potential for medium-grain parallel algorithms associated with the eigenvalue problem that must be solved for each sector. Here there is the obvious trade-off between the number of processors assigned to each sector and the number of sectors assigned to each cluster. Load balancing among the sectors needs to be addressed because the order of the 2D Hamiltonians may change dramatically from sector to sector. However, within the SDT method the ultimate order of the matrix to be diagonalized after the truncation procedure does not vary much across the various sector calculations. Major differences may be addressed by judiciously adjusting the number of processors per cluster at the intermediate levels in order to have convenient access to the necessary memory and to achieve optimal load balancing.

Since the steps necessary to achieve efficient large-grain parallelization at the sector level are rather obvious, we will focus our discussion on the intermediate level (cluster level) of parallelization that is possible for each sector. In the context of this application, the important contribution of the IRLM algorithm is that it facilitates and makes possible the efficient mapping of the eigenvalue/eigenvector computations onto a cluster of processors [18b], provided the necessary matrix-vector product can also be efficiently mapped onto the same cluster. To demonstrate that this is the case, we begin by noting that there are two major parts of the code. The first consists of the creation of the $B_{ii}$ and the $Q_{ii}$ submatrices. For this step it is not necessary to have direct access to the entire 2D Hamiltonian matrix. A single processor can generate and diagonalize the submatrices $B_{ii}$, producing the submatrices $Q_{ii}$.

The next step is the formation of the matrix-vector product. If one splits the matrix-vector multiply into its component parts, viz.,

$$\mathbf{y} = [\tilde{\mathbf{Y}} + \tilde{\mathbf{Q}}^{\mathrm{T}}\mathbf{K}\tilde{\mathbf{Q}}]\mathbf{x} \tag{31}$$

becomes

$$\mathbf{u} = \tilde{\mathbf{Q}}\mathbf{x} \tag{32a}$$

$$\mathbf{v} = \mathbf{K}\mathbf{u} \tag{32b}$$

$$\mathbf{w} = \tilde{\mathbf{Q}}^{\mathrm{T}}\mathbf{v} \tag{32c}$$

and

$$\mathbf{y} = \tilde{\mathbf{Y}}\mathbf{x} + \mathbf{w}, \tag{32d}$$

where

$$w_i = \sum_{j \neq i}^{n_\theta} \tilde{Q}_{ii}^{\mathrm{T}} \alpha_{ij} \mathbf{1} \tilde{Q}_{jj} x_j. \tag{32e}$$

Appendix B contains the derivation of Eq. (32e). In order to form the matrix-vector multiply each processor or cluster of processors requires the full set of $Q_{ii}$ and the entire vector $\mathbf{x}$. A version of the code employing the matrix-vector multiply scheme of Eq. (32e) has been implemented and parallelized on a Cray YMP.

For those machines with BLAS (basic linear algebra subroutines) routines, each $w_i$ can be calculated using $2n_\theta$ gemv calls and $(n_\theta^2 - n_\theta)$ axpy calls. If each processor in the cluster has access to all the $Q_{ii}$'s and initially to the $x_i$ component of the full $\mathbf{x}$, the matrix-vector multiply can be accomplished by establishing a loop of nearest neighbor processors that each contains one $x_i$ component of $\mathbf{x}$ initially. Following computation of the partial sums for each $y_i$, the $x_i$ are then passed around this loop of processors so that the full sums can be obtained. The resulting $y_i$ can be passed directly to the IRLM algorithm without interprocessor communication, provided the multiprocessor mapping is arranged to correspond to IRLM's matrix-vector partitioning [18c]. As a result, the interprocessor communication associated with the matrix-vector product is relatively small and efficient.

## V. POLYNOMIAL PRECONDITIONING

There are two fundamental computational costs associated with IRLM. One is the cost of the internal numerical operations required to perform the IRLM. These include both the orthogonalization and the updating operations applied to the basis vectors. Second, there is the cost associated with performing a matrix-vector product with the discrete Hamiltonian $\mathscr{H}^{\mathrm{DVR}}$. Typically, each iteration of the IRLM requires a number of matrix vector products with $\mathscr{H}^{\mathrm{DVR}}$ proportional to the number of eigenvalues sought.

The IRLM takes fewer iterations when the desired eigenvalues are well separated and near one end of the spectrum. The eigenproblems arising in this application present a distribution of eigenvalues that are well separated at the right end and tightly clustered at the left end of the spectrum. This is opposite to the desired distribution since we seek the eigenvalues at the left end.

A standard way to accelerate convergence of Lanczos algorithms in general is to find the eigenvalues and eigenvectors of a function of the matrix and then use the eigenvectors to compute eigenvalues of the original matrix, **A**, via Raleigh quotients. For example, if

$$\phi(\mathbf{A})\mathbf{q} = \mathbf{q}\mu \qquad \text{then} \quad \mathbf{A}\mathbf{q} = \mathbf{q}\boldsymbol{\lambda} \tag{33}$$

with

$$\boldsymbol{\lambda} = \mathbf{q}^{\mathrm{T}}\mathbf{A}\mathbf{q}, \qquad \mu = \phi(\lambda). \tag{34}$$

This function $\phi$ should, of course, be constructed so that the transformed spectrum $\{\mu = \phi(\lambda) : \lambda \in \text{spectrum}(\mathbf{A})\}$ has a desirable distribution.

The eigenvalues of interest are in an interval $(0, a_0)$ and the entire spectrum of the matrix lies in an interval $(0, a_1)$ with $a_0 < a_1$. To enhance convergence in this case, one should construct $\phi$ to be monotone decreasing rapidly on the interval $(0, a_1)$ from a large positive value at zero to a relatively small value at $a_1$ and then to be as small as possible in absolute value over the remaining interval $[a_0, a_1)$. One of the more commonly used transformations is the shift and invert method which sets $\phi$ as $\phi(\lambda) = 1/(\lambda + \sigma)$, where $\sigma$ is some fixed non-negative number. The corresponding matrix function is

$$\phi(\mathbf{A}) = (\mathbf{A} + \sigma\mathbf{I})^{-1}. \tag{35}$$

The Lanczos process for this transformed matrix would require matrix vector products of the form

$$w = (\mathbf{A} + \sigma\mathbf{I})^{-1} v \tag{36}$$

at each step. Since the inverse of a sparse matrix is often dense (and also for numerical reasons), this operation is generally accomplished by solving the linear system

$$(\mathbf{A} + \sigma\mathbf{I}) w = v. \tag{37}$$

This may be done by computing the Cholesky factorization of the matrix and then using the triangular factorization to solve linear systems of this form at each step of the Lanczos iteration. Unfortunately, this approach is not attractive for our application since the Cholesky factors of the $\mathscr{H}^{\mathrm{DVR}}$ matrix will be nearly dense.

Another approach is to use an iterative method to solve these linear systems. With an appropriate pre-conditioner [23] this might be viable. However, at present we have not found a pre-conditioner that will make the iterative solver for the linear system converge fast enough to produce a saving in time with respect to just using the IRLM on the original matrix. This may be explained to some extent by a heuristic argument concerning the amount of information present in the underlying Krylov subspace associated with a fixed number of matrix vector products with the matrix **A** and a given starting vector $v_1$. Further investigation is needed for a complete analysis.

An alternative is to construct a function $\phi(\mathbf{A})$ that is easy to compute. One possibility is to construct a low degree polynomial $\phi$. There are many options. A standard choice is to use a Chebychev polynomial associated with the interval $[a_0, a_1)$. This polynomial may be specified in several ways. In terms of its roots, the polynomial is

$$\prod_{j=1}^{m} (\lambda - \mu_j), \tag{38}$$

where

$$\mu_j = c + r\cos(\eta_j), \qquad \eta_j = \pi(2j-1)/2m, \qquad j = 1, 2, \ldots, m, \tag{39}$$

with $c$ being the center and $r$ being half the length of the interval $[a_0, a_1)$. These polynomials are usually specified by a three-term recursion and when normalized to be monic, the $m$th polynomial has the well-known property of having the smallest maximum absolute value on the interval $[a_0, a_1)$ over all monic polynomials of degree $m$. See Ref. [24] for further details.

This can be very effective but there are again trade-offs concerning the degree $m$ and the overall effect on the total number of matrix vector products required. acceleration has been used in a different form by several authors including Saad [25], Chatelin and Ho [26], and Sadkane [27]. Here, we merely hope to obtain a transformation of the spectrum that will induce favorable convergence properties within the IRLM.

## V. RESULTS AND DISCUSSION

To evaluate the proposed approach we have subjected our new algorithm combined with the IRLM algorithm to a wide variety of tests in order to determine how robust the method is for obtaining the eigensolutions of the surface Hamiltonian. We have also carried out detailed comparisons of this new approach with the SDT algorithm that is currently used in the APH scattering code. To evaluate both the relative computational times and the memory requirements, representative diagonalization problems were selected spanning the typical range of representation-dimensionalities employed in the generation of surface functions for relatively light, atom–diatom reactive scattering problems. The potential and other parameters from the recent study [12] of the reaction $He + H_2^+ \rightarrow HeH^+ + H$ were used throughout. The performance of the codes on both workstation-type machines, as well as on a supercomputer that allows for efficient vectorization, has been examined. The representative results for different types of machine architectures are reported as a way of gauging the expected enhancements on machines widely utilized in scientific research and production runs. The relative storage requirements of the DVR codes are important because more complex systems will involve more storage, and, for distributed memory systems, the memory requirements can be a significant issue.

Table I contains a summary of timing tests on a Cray-YMP for a variety of orders of Chebychev polynomial preconditioners. As a result of these and other studies of various Chebychev preconditioning schemes we have decided that a six-term polynomial preconditioner is sufficient to transform the eigenvalue spectrum so that the desired eigenvectors and eigenvalues can be determined efficiently using the IRLM for this particular application. For this reason, all the results reported below for the new algorithm (i.e., present results) have utilized a six-term Chebychev preconditioner. Figure 2 shows the normal eigenvalue spectrum for $\mathcal{H}^{DVR}$, demonstrating the dense pattern of eigenvalues at the left end of the spectrum and the less dense pattern at the right. Figure 3 shows the eigenvalue spectrum after preconditioning with a six-term polynomial. Note the difference in the eigenvalue spectrum as compared with Fig. 2.

In Table II we report the performance of both versions of the SDT-DVR code (the new and the conventional) in addition to their respective memory requirements over a range of dimensions representative of those that are employed in the investigation of small-to-medium size systems.

The results for the SPARC-2 test runs indicate clearly the superior performance of the new approach over the conven-

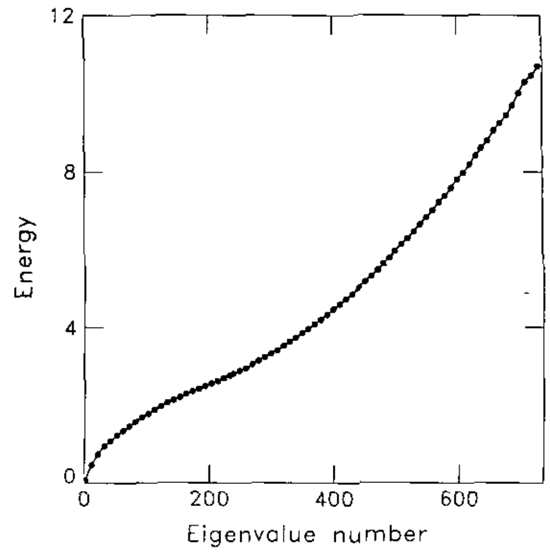

FIG. 2. Plot of the eigenvalue spectrum of $\mathcal{H}^{DVR}$ following SDT truncation. Full DVR dimension = 3200; truncated dimension = 735; $\rho = 5.0$ bohr. The abscissa denotes the eigenvalue number and the ordinate gives the eigenvalue. Note the density of the eigenvalues on the left end of the spectrum. The dots designate every tenth eigenvalue.

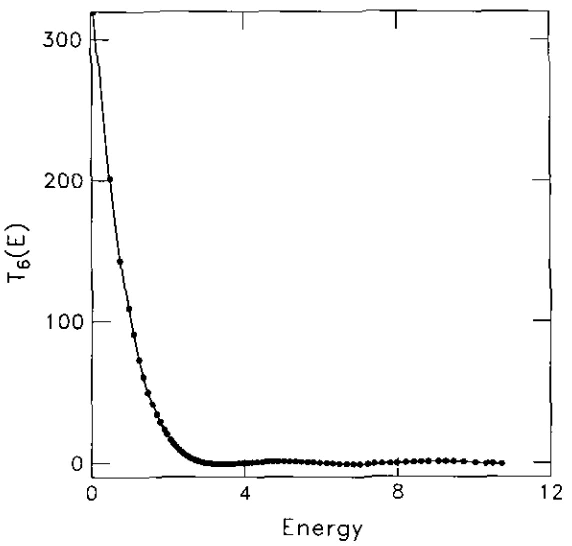

FIG. 3. Plot of the eigenvalue spectrum for the same case as in Fig. 2. The abscissa denotes the eigenvalue and the ordinate give the value of the sixth-order Chebyshev polynomial at the ordinate value. Note the change in density of the eigenvalue spectrum at the left end of the plot due to the Chebyshev transformation Eqs. (38) and (39). The dots designate every tenth eigenvalue.

TABLE I

Timing Results Comparing Order of Chebychev Polynomial

| Order of polynomial | Case 1[a] | Case 2[b] | Case 3[c] |
|---|---|---|---|
| 2 | 14.1 | 23.0 | 27.4 |
| 3 | 13.4 | 22.3 | 25.7 |
| 4 | 13.4 | 23.0 | 26.1 |
| 5 | 15.0 | 22.4 | 26.5 |
| 6 | 14.2 | 25.4 | 25.7 |

*Note.* Times in seconds on a single processor of a Cray YMP.
[a] Full dimension = 5000; truncated dimension = 772; $\rho = 8.05$ bohr.
[b] Full dimension = 7200; truncated dimension = 928; $\rho = 8.05$ bohr.
[c] Full dimension = 7200; truncated dimension = 1099; $\rho = 5.0$ bohr.

TABLE II

Timing Results and Memory Requirements for the Present and Conventional Sequential Diagonalization Truncation Runs on a SPARC-2

|  |  |  |  |  | Time |  | Memory |  |
|---|---|---|---|---|---|---|---|---|
| $N^a$ | $N_\theta$ | $N_\chi$ | $N'^b$ | $M^c$ | $PM^d$ | $CM^e$ | $PM^f$ | $CM^g$ |
| 3200 | 40 | 80 | 735 | 100 | 306 | 1041 | 3.8 | 12.6 |
| 7200 | 60 | 120 | 1099 | 100 | 943 | 3470 | 9.6 | 35.2 |
| 9750 | 65 | 150 | 1193 | 100 | 1169 | 4393 | 14.7 | 52.2 |
| 11250 | 75 | 150 | 1376 | 100 | 2960 | 6717 | 16.9 | 63.2 |
| 12800 | 80 | 160 | 1467 | 100 | 3521 | 8182 | 20.0 | 75.0 |

[a] The original order of the Hamiltonian matrix.
[b] The order of the Hamiltonian matrix subsequent to truncation due to the application of *ecut*.
[c] The number of eigenvalues requested.
[d] The execution time in seconds on the SPARC-2 employing the present SDT method.
[e] The execution time in seconds on the SPARC-2 employing the conventional SDT method.
[f] The memory requirement on the SPARC-2 for the present code in megabytes.
[g] The memory requirement on the SPARC-2 for the conventional SDT code in megabytes.

tional SDT-DVR implementation. For dimensions equal to or greater than 3200 our present DVR code is, on the average, more efficient than the conventional implementation by a factor of 3 to 4. Since a typical production run for a single value of the total angular momentum might require of the order of 1000 such diagonalizations, the extent of savings in total computational time to be secured through the implementation of this new approach is considerable.

There are several reasons for this enhanced performance. The most important factor is our ability to convert a full diagonalization problem into a much smaller eigenvalue problem. Computational effort is thus focused on the portion of the eigenvalue problem that is actually of interest. Such a strategy results in speedups in both execution time and wall clock time for production runs. Moreover, use of the IRLM preserves the sparse structure of the 2D Hamiltonian through the matrix-vector product phase of the computations.

A third significant advantage of the new algorithm is the particular implementation of the Chebyshev preconditioning process developed for this application. For each case the range of unwanted eigenvalues (i.e., the region $[a_0, a_1)$) must be specified. Since IRLM is very efficient in finding the largest eigenvalue of the SDT-DVR Hamiltonian, operationally one can specify $a_1$ by setting it equal to the highest calculated eigenvalue. This is a very rapid calculation even without the Chebychev preconditioning because the highest eigenvalue is well separated from the rest of the spectrum (see Fig. 2). Selecting a value for $a_0$ is not as straightforward because the nature of the eigenvalue spectrum is not known in advance. After numerous empirical tests to determine the optimal values of $a_0$ for various problems as well as the sensitivity of the computation times to the variation of $a_0$ from the optimal value, we discovered that setting $a_0$ equal to the value of SDT cutoff energy, *ecut*, produced a very effective preconditioning polynomial for this application. Moreover, this value is only slightly less effective than the optimal value of $a_0$.

The last two columns of Table II summarize the memory requirements of the two SDT-DVR codes. Here the new code also has the advantage. Its memory demands are on the average much less than those required by the conventional approach—about a factor of four less. The difference in the memory requirements for the two SDT-DVR implementations arise from two factors: (1) the largest memory requirement in the previous SDT-DVR code comes from the choice of eigenvalue/eigenvector routine used for the truncated 2D Hamiltonian. The routine calculates *all* of the eigenvalues and eigenvectors, thus requiring storage capacity for these values. IRLM allows one to choose the number of eigenpairs calculated and only requires storage proportional to twice the number of eigenvectors requested, thus providing for significant savings. (2) The previously implemented code actually completes the transformation of Eq. (22) and stores the entire truncated 2D Hamiltonian. The final order of this matrix is approximately $\sum_{i=1}^{n\theta} n'_\chi(i)$. Under the best of circumstances the previous code requires storage for two matrices of this order: one for the matrix and one for the eigenvectors. The newly implemented code requires only the rectangular $\tilde{Q}_{ii}$ arrays (at most $n_\theta n_\chi^2$ words) plus the eigenvector storage mentioned above.

Tables III through VI provide an assessment of the performances of both computer codes on three different types of machines, each with its own unique architecture. Each table presents results for a particular case characterized by its full representation dimension, the truncated dimension, and the value of the hyperspherical scattering coordinate $\rho$. The four cases were chosen as being representative of systems that have been or are being currently investigated, or of those into which we want to expand our research efforts in the near future. We have also included the amount of the total time of the calculation spent in the matrix-vector product routine. This is about two-thirds of the total time involved in the calculation because the routine is called a large number of times from the IRLM routines.

Results listed in Table III for $\rho = 5.0$ bohr and a truncated dimension of 735 are representative of dimensions employed in the study of system such as $HeH_2^+$ in regions where the propagation is less than half-way complete. The performance ratio—the old time divided by the new time—

**TABLE III**

Timing Results for the Present and Conventional Sequential Diagonalization Truncation Runs

| Machine | $MV$[a] time | Total[b] time | Performance[c] ratio |
|---|---|---|---|
| SPARC-2[d] | 192 | 306 | 3.6 |
| SPARC-2 | — | 1118 | |
| RS-6000/530 | 49 | 81 | 4.0 |
| RS-6000/530 | — | 320 | |
| CRAY-YMP | 7 | 12 | 2.6 |
| CRAY-YMP | — | 31 | |

*Note.* Full dimension = 3200; truncated dimension = 735; $\rho = 5.0$ bohr
[a] The CPU time spent in the matrix-vector multiplication routine in seconds.
[b] The CPU time in seconds for setup plus diagonalization.
[c] The performance ratio = CPU time (conventional code)/CPU time (present code).
[d] The first line of each entry for each machine is for the current method while the second line is for the conventional method.

**TABLE V**

Timing Results for the Present and Conventional Sequential Diagonalization Truncation Runs

| Machine | $MV$[a] time | Total[b] time | Performance[c] ratio |
|---|---|---|---|
| SPARC-2[d] | 320 | 460 | 3.0 |
| SPARC-2 | — | 1379 | |
| RS-6000/530 | 80 | 125 | 3.1 |
| RS-6000/530 | — | 393 | |
| CRAY-YMP | 8 | 14 | 3.0 |
| CRAY-YMP | — | 42 | |

*Note.* Full dimension = 5000; truncated dimension = 772; $\rho = 8.05$ bohr
[a] The CPU time spent in the matrix-vector multiplication routine in seconds.
[b] The CPU time in seconds for setup plus diagonalization.
[c] The performance ratio = CPU time (conventional code)/CPU time (present code).
[d] The first line of each entry is for the current method while the second line is for the conventional method

increases in going from the CRAY-YMP to the SPARC-2 and the RS6000 with a range of [2.6, 4].

On the Cray YMP the range of megaflops (MFLOPS) for the present code was 104 ($n_\chi = 80$; $n_\theta = 40$) to 127 ($n_\chi = 120$; $n_\theta = 60$) while the range for the conventional code was 127 to 138. Both codes vectorize about the same amount with the conventional code working with a full truncated matrix while the present code's performance is dependent upon the length of the $n_\chi$ and $n_\theta$ variables. As these numbers increase, better use is made of the Cray's vector processors, resulting in an increased MFLOPS performance.

Table IV contains results from a case at the same $\rho$ value of 5.0 bohr but with a higher truncated dimensionality of 1099. There is an overall increase in the performance ratios across the various machines. This is an important factor to be taken into consideration in conjunction with the feasibility of studies focusing on systems characterized by a

**TABLE IV**

Timing Results for the Present and Conventional Sequential Diagonalization Truncation Runs

| Machine | $MV$[a] time | Total[b] time | Performance[c] ratio |
|---|---|---|---|
| SPARC-2[d] | 672 | 943 | 4.1 |
| SPARC-2 | — | 3833 | |
| RS-6000/530 | 174 | 252 | 4.5 |
| RS-6000/530 | — | 1141 | |
| CRAY-YMP | 16 | 26 | 3.7 |
| CRAY-YMP | — | 96 | |

*Note.* Full dimension = 7200; truncated dimension = 1099; $\rho = 5.0$ bohr
[a] Matrix vector multiplication time.
[b] Setup plus diagonalization.
[c] Performance ratio = conventional CPU time/current CPU time.
[d] The first line of each entry is for the current method while the second line is for the conventional method

**TABLE VI**

Timing Results for the Present and Conventional Sequential Diagonalization Truncation Runs

| Machine | $MV$[a] time | Total[b] time | Performance[c] ratio |
|---|---|---|---|
| SPARC-2[d] | 515 | 743 | 3.2 |
| SPARC-2 | — | 2416 | |
| RS-6000/530 | 129 | 194 | 3.6 |
| RS-6000/530 | — | 705 | |
| CRAY-YMP | 15 | 25 | 2.8 |
| CRAY-YMP | — | 70 | |

*Note.* Full dimension = 7200; truncated dimension = 928; $\rho = 8.05$ bohr
[a] The CPU time spent in the matrix-vector multiplication routine in seconds.
[b] The CPU time in seconds for setup plus diagonalization.
[c] The performance ratio defined as CPU time (conventional code)/CPU time (present code).
[d] The first line of each entry is for the current method while the second line is for the conventional method

higher density of states than the $HeH_2^+$ results presented here. Performance ratios ranging from 3.7 to 4.5 are found for this case.

Table V summarizes the results from a case representative of regions lying in the second half of the $HeH_2^+$ propagation at a $\rho$ value of 8.05 bohr and a reduced dimension of 772. The performance ratios for this case are about 3.

Finally in Table VI results are presented for a case similar to the previous one but with a somewhat larger DVR representation as might be characteristic of a more state-intensive system. Here too the performance ratios range from 2.8 to 3.6.

## VI. CONCLUSIONS

The results presented in this paper indicate that our new algorithm, which combines the SDT-DVR method [21] and the IRLM algorithm [18] with several special features, provides the basis for a robust methodology for obtaining the 2D surface functions needed in the APH 3D reactive scattering method of Parker and Pack [1]. The new code has been developed and tested. It is about 3 to 4 times faster and requires only about one-fourth the memory needed for the current implementation of the SDT-DVR code. These initial results are sufficiently encouraging for mainline scalar and vector machines that we plan to incorporate this new option into the full 3D scattering code in the near future. This new approach to a demanding phase of the overall scattering problem will facilitate both setup and production studies of systems with more energetically-open initial and final quantum states. Beyond the applications possible with traditional scalar and vector processors, it is significant that this new algorithm is, in principle, scalable with the total number of processors on a massively parallel architecture computer. Work to implement this new code on massively parallel systems such as the Intel Delta and the CM-5 is currently underway.

## APPENDIX A

The derivation of the matrix elements needed to assemble the 2D Hamiltonian matrix $\mathscr{H}^{\text{DVR}}$ begins with Eq. (16) of Ref. [22],

$$\mathbf{h}_x^{\text{DVR}} = \mathbf{U}^{\text{T}} \mathbf{h}_x^{\text{FBR}} \mathbf{U}, \tag{A.1}$$

where the operators for $\mathbf{h}_\theta^{\text{FBR}}$ and $\mathbf{h}_\chi^{\text{FBR}}$ matrix elements are

$$h_\theta = -\frac{\hbar^2}{2\mu\rho_\zeta^2} \frac{16}{\sin\vartheta} \frac{\partial}{\partial\vartheta} \sin\vartheta \frac{\partial}{\partial\vartheta} \tag{A.2}$$

and

$$h_\chi = -\frac{\hbar^2}{2\mu\rho_\zeta^2} \frac{\partial^2}{\partial\chi^2}. \tag{A.3}$$

The finite basis representation (FBR) for the $\theta$-space is formed from the normalized associated legendre polynomials $P_l^0(\cos\vartheta)$. The transformation between the FBR and the DVR basis functions is

$$D_j^{\text{DVR}}(\cos\vartheta) = \sum_{l=0}^{l_{\max}} U_{lj}^\theta P_l^0(\cos\vartheta), \tag{A.4}$$

while, according to the Christoffel–Darboux theorem, $\mathbf{U}^\theta$ is given by

$$U_{lj}^\theta = P_l^0(\cos\vartheta_j)\, w_j^{1/2}, \tag{A.5}$$

wherein $\vartheta_j$ is the value of $\vartheta$ at the quadrature point indexed by $j$ and $w_j$ is the corresponding weight for this Gauss–Legendre quadrature point. Thus $\vartheta_j$ is the $j$th zero of $P_{l_{\max}}^0(\cos\vartheta)$.

From Eq. (A.1) it follows that

$$\begin{aligned} h_\theta^{\text{DVR}}(jj') &= \langle D_{j'} | h_\theta^{\text{FBR}} | D_j \rangle \\ &= \sum_{l=0}^{l_{\max}} \sum_{l'=0}^{l_{\max}} U_{lj}^\theta U_{l'j'}^\theta \langle P_l^0(\cos\vartheta_j) | h_\theta | P_{l'}^0(\cos\vartheta_{j'}) \rangle. \end{aligned} \tag{A.6}$$

Using Eq. (A.2) and the orthonormality of the $P_l^0(\cos\theta)$ one obtains

$$h_\theta^{\text{FBR}}(ll') = \frac{8\hbar^2}{\mu\rho_\zeta^2} l(l+1)\, \delta_{ll'}, \tag{A.7}$$

which, upon substitution into Eq. (A.6), results in

$$h_\theta^{\text{DVR}}(jj') = \frac{8\hbar^2}{\mu\rho_\zeta^2} \sum_{l=0}^{l_{\max}} l(l+1)\, P_l^0(\cos\vartheta_j)\, P_l^0(\cos\vartheta_{j'}) (w_j w_{j'})^{1/2}. \tag{A.8}$$

From Eq. (A.8) and the expression for the 2D Hamiltonain matrix $\mathscr{H}^{\text{DVR}}$, Eq. (13), it can be seen that the contribution to $\mathscr{H}^{\text{DVR}}$ from the $\mathbf{h}_\theta^{\text{DVR}} \otimes \mathbf{I}_\chi$ term can be written simply as $\alpha_{jj'} \mathbf{I}_\chi$. It is this feature of the DVR tensor product when combined with the properties of the $\mathbf{f}_\theta^{\text{DVR}} \otimes \mathbf{h}_\chi^{\text{DVR}}$ term, discussed below, that leads to the offdiagonal striping illustrated in Fig. 1.

For the $\chi$-space and the range of integration zero to $\pi/2$, the normalized basis set $(\Pi_m = N_m \cos(2(m-1)\chi))$, $m = 1, 2, \ldots, m_{\max}$, is the finite basis representation, where $N_m$ is the normalization factor. The transformation between the FBR and the DVR is

$$D_k^{\mathrm{DVR}}(\chi) = \sum_{m=1}^{m_{\max}} U_{mk}^{\chi} \Pi_m(\chi), \tag{A.9}$$

where, according to the Christoffel–Darboux theorem, $\mathrm{U}^{\chi}$ is given by

$$U_{mk}^{\chi} = \Pi_m(\chi_k)\, w_k^{1/2}, \tag{A.10}$$

wherein $\chi_k$ is the value of $\chi$ at the Gauss–Chebychev quadrature point indexed by $k$ and $w_k$ is the constant weight, $\pi/2m_{\max}$.

From Eq. (A.1) it follows that

$$\begin{aligned} h_\chi^{\mathrm{DVR}}(kk') &= \langle D_{k'}\, |h_\chi^{\mathrm{FBR}}|\, D_k\rangle \\ &= \sum_{m=1}^{m_{\max}} \sum_{m'=1}^{m_{\max}} U_{mk}^{\chi} U_{m'k'}^{\chi} \langle \Pi_m(\chi_k)\, |h_\chi|\, \Pi_{m'}(\chi_{k'})\rangle. \end{aligned} \tag{A.11}$$

Using Eq. (A.3) and the orthogonality of the $\Pi_m(\chi_k)$, one obtains

$$h_\chi^{\mathrm{FBR}}(mm') = \frac{2\hbar}{\mu\rho_\zeta^2}\, m^2 \delta_{mm'} \tag{A.12}$$

which, upon substitution into (A.11), leads to

$$\mathbf{h}_\chi^{\mathrm{DVR}}(kk') = \frac{2\hbar}{\mu\rho_\zeta^2} \sum_{m=1}^{m_{\max}} m^2 U_{mk}^{\chi} U_{mk'}^{\chi}. \tag{A.13}$$

From Eq. (A.13) and the expression for the 2D Hamiltonian matrix $\mathscr{H}^{\mathrm{DVR}}$, Eq. (13), it can be seen that the contribution to $\mathscr{H}^{\mathrm{DVR}}$ from the $\mathbf{f}_\theta^{\mathrm{DVR}} \otimes \mathbf{h}_\chi^{\mathrm{DVR}}$ corresponds to the $\mathscr{B}$ matrix of Eq. (15). Since the matrix for $\mathbf{f}_\theta^{\mathrm{DVR}}$ is diagonal, it follows that the offdiagonal $\mathscr{B}_{ij}$ blocks are zero.

## APPENDIX B

The 2D Hamiltonian matrix $\mathscr{H}^{\mathrm{DVR}}$ is expressed as

$$\mathscr{H}^{\mathrm{DVR}} = \mathbf{B} + \mathbf{K}, \tag{B.1}$$

where $\mathbf{B}$ consists of a series of block diagonal submatrices of order $n_\chi$ and $\mathbf{K}$ is the matrix of off-diagonal rays whose submatrices have order $n_\chi$. The form of $\mathscr{H}^{\mathrm{DVR}}$ is shown in Fig. 1.

A matrix $\mathbf{Q}$ is defined such that

$$\mathbf{Q}^{\mathrm{T}}\mathbf{B}\mathbf{Q} = \mathbf{\Upsilon}, \tag{B.2}$$

where $\mathbf{\Upsilon}$ is diagonal. The matrix to be diagonalized is

$$\begin{aligned} \mathbf{Q}^{\mathrm{T}}\mathscr{H}^{\mathrm{DVR}}\mathbf{Q} &= \mathbf{Q}^{\mathrm{T}}\mathbf{B}\mathbf{Q} + \mathbf{Q}^{\mathrm{T}}\mathbf{K}\mathbf{Q} \\ &= \mathbf{\Upsilon} + \mathbf{Q}^{\mathrm{T}}\mathbf{K}\mathbf{Q}. \end{aligned} \tag{B.3}$$

Recall that

$$\mathbf{Q} = Q_{ii} \quad (i = 1, \ldots, n_\theta), \tag{B.4}$$

with ${}_{ij} \equiv 0$ for $i \neq j$ and, pictorially, $\mathbf{Q}$ looks exactly like $\mathbf{B}$. Also, the $\mathbf{K}$ matrix may be expressed as a series of submatrices $K_{ij}$,

$$\mathbf{K} = K_{ij} \quad (i, j = 1, \ldots, n_\theta), \tag{B.5}$$

where each $K_{ij}$ is of order $n_\chi$ and all $K_{ii} = 0$. $K_{ij} = \alpha_{ij}\mathbf{1}$, where $\alpha_{ij}$ is a scalar and $\mathbf{1}$ is the unit matrix.

The matrix-vector multiply is split into parts, viz.,

$$\mathbf{y} = [\mathbf{\Upsilon} + \mathbf{Q}^{\mathrm{T}}\mathbf{K}\mathbf{Q}]\mathbf{x} \tag{B.6}$$

becomes

$$\mathbf{u} = \mathbf{Q}\mathbf{x} \tag{B.7}$$

$$\mathbf{v} = \mathbf{K}\mathbf{u} \tag{B.8}$$

$$\mathbf{w} = \mathbf{Q}^{\mathrm{T}}\mathbf{v} \tag{B.9}$$

and

$$\mathbf{y} = \mathbf{\Upsilon}\mathbf{x} + \mathbf{w}. \tag{B.10}$$

Ignoring the effects of the truncation scheme (which only alter the orders of the matrices and not the specific details of the matrix-vector multiplication), define a vector $\mathbf{x}$ which is divided into $n_\theta$ parts each consisting of $n_\chi$ components,

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n_\theta} \end{pmatrix}. \tag{B.11}$$

The same may be done for the vectors $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$, and $\mathbf{y}$.

The operations are

$$\mathbf{u}=\mathbf{Qx}=\begin{pmatrix} Q_{11}\cdot x_1 \\ Q_{22}\cdot x_2 \\ \vdots \\ Q_{n\theta n\theta}\cdot x_{n\theta} \end{pmatrix}, \tag{B.12}$$

$$\mathbf{v}=\mathbf{Ku}=\begin{pmatrix} K_{12}u_2+K_{13}u_3+\cdots+K_{1n\theta}u_{n\theta} \\ K_{21}u_1+K_{23}u_3+\cdots+K_{2n\theta}u_{n\theta} \\ \vdots \\ K_{n\theta 1}u_1+K_{n\theta 3}u_2+\cdots+K_{n\theta n\theta-1}u_{n\theta-1} \end{pmatrix},$$

$$=\begin{pmatrix} K_{12}Q_{22}\cdot x_2+K_{13}Q_{33}\cdot x_3+\cdots+K_{1n\theta}Q_{n\theta n\theta}\cdot x_{n\theta} \\ K_{21}Q_{11}\cdot x_1+K_{23}Q_{33}\cdot x_3+\cdots+K_{2n\theta}Q_{n\theta n\theta}\cdot x_{n\theta} \\ \vdots \\ K_{n\theta 1}Q_{11}\cdot x_1+K_{n\theta 2}Q_{22}\cdot x_2+\cdots+K_{n\theta n\theta-1}Q_{n\theta n\theta-1}\cdot x_{n\theta-1} \end{pmatrix}, \tag{B.13}$$

and

$$\mathbf{w}=\mathbf{Q}^{\mathrm{T}}\mathbf{v}$$

$$=\begin{pmatrix} Q_{11}^{\mathrm{T}}K_{12}Q_{22}\cdot x_2+Q_{11}^{\mathrm{T}}K_{13}Q_{33}\cdot x_3+\cdots+Q_{11}^{\mathrm{T}}K_{1n\theta}Q_{n\theta n\theta}\cdot x_{n\theta} \\ Q_{22}^{\mathrm{T}}K_{21}Q_{11}\cdot x_1+Q_{22}^{\mathrm{T}}K_{23}Q_{33}\cdot x_3+\cdots+Q_{22}^{\mathrm{T}}K_{2n\theta}Q_{n\theta n\theta}\cdot x_{n\theta} \\ \vdots \\ Q_{n\theta n\theta}^{\mathrm{T}}K_{n\theta 1}Q_{11}\cdot x_1+Q_{n\theta n\theta}^{\mathrm{T}}K_{n\theta 2}Q_{22}\cdot x_2+\cdots+Q_{n\theta n\theta}^{\mathrm{T}}K_{n\theta n\theta-1}Q_{n\theta n\theta-1}\cdot x_{n\theta-1} \end{pmatrix}, \tag{B.14}$$

from which it can be shown by induction that

$$w_i=\sum_{j\neq i}^{n\theta} Q_{ii}^{\mathrm{T}}K_{ij}Q_{jj}\cdot x_j \tag{B.15}$$

or

$$w_i=\sum_{j\neq i}^{n\theta} Q_{ii}^{\mathrm{T}}\alpha_{ij}\mathbf{1}Q_{jj}\cdot x_j. \tag{B.16}$$

## ACKNOWLEGMENTS

Partial support for this project came from National Science Foundation project (ASC-9113693); additional support was also provided by the National Science Foundation in the form of a postdoctoral fellowship grant (ACS-9211104). Computer time on the Cray-YMP was provided by a grant from the Ohio Supercomputer Center. The authors also thank J. D. Kress, R. T. Pack, and G. A. Parker for many insightful discussions of earlier 3D quantum scattering studies and various approaches to obtaining the requisite 2D surface functions. Finally, the authors wish to acknowledge the use of AMS-TEX (Version 2.0) from the Americain Mathematical Society in the preparation of a Tex version of this manuscript.

## REFERENCES

1. (a) G. A. Parker, R. T. Pack, and R. B. Walker, *Chem. Phys. Lett.* **137**, 564 (1987); (b) R. T. Pack and G. A. Parker, *J. Chem. Phys.* **87**, 3888 (1987).
2. (a) J. Z. H. Zhang and W. H. Miller, *Chem. Phys. Lett.* **140**, 329 (1987); (b) J. Z. H. Zhang, S. I. Chu, and W. H. Miller, *J. Chem. Phys.* **88**, 4549 (1988).
3. (a) D. W. Schwenke, K. Haug, D. G. Truhlar, Y. Sun, J. Z. H. Zhang, and D. J. Kouri, *J. Phys. Chem.* **91**, 6080 (1987); (b) D. W. Schwenke, K. Haug, M. Zhao, D. G. Truhlar, Y. Sun, J. Z. H. Zhang, and D. J. Kouri, *J. Phys. Chem.* **92**, 3202 (1988).
4. D. E. Manolopoulos and R. E. Wyatt, *Chem. Phys. Lett.* **152**, 23 (1988).
5. (a) K. Haug, D. W. Schwenke, D. G. Truhlar, J. Z. H. Zhang, and D. J. Kouri, *J. Phys. Chem.* **90**, 6757 (1986); (b) J. Z. H. Zhang, D. J. Kouri, K. Haug, D. W. Schwenke, Y. Shima, and D. J. Truhlar, *J. Chem. Phys.* **88**, 2492 (1988).
6. (a) M. Bear and Y. Shima, *Phys. Rev. A* **35**, 5252 (1987); (b) M. Baer, *J. Phys. Chem.* **91**, 5846 (1987); (c) D. Neuhauser and M. Baer, *J. Chem. Phys.* **88**, 2856 (1988); (d) M. Baer, *J. Chem. Phys.* **90**, 3043 (1989).
7. (a) F. Webster and J. C. Light, *J. Chem. Phys.* **90**, 265 (1989); (b) F. Webster and J. C. Light, *J. Chem. Phys.* **90**, 300 (1989).
8. (a) A. Kuppermann and P. G. Hipes, *J. Chem. Phys.* **84**, 5962 (1986); (b) S. A. Cuccaro, P. G. Hipes, and A. Kuppermann, *Chem. Phys. Lett.* **154**, 155 (1989); (c) B. Lepetit, Z. Peng, and A. Kuppermann, *Chem. Phys. Lett.* **166**, 572 (1990); (d) B. Lepetit and A. Kuppermann, *Chem. Phys. Lett.* **166**, 581 (1990); (e) A. Kuppermann and Y.-S. M. Wu, *Chem. Phys. Lett.* **166**, 581 (1990); (e) A. Kuppermann and Y.-S. M. Wu, *Chem. Phys. Lett.* **205**, 577 (1993).
9. (a) J. Linderberg and B. Vessal, *Int. J. Quant. Chem.* **31**, 65 (1987); (b) J. Linderberg and S. B. Padkjaekar, Y. Öhrn, and B. Vessal, *J. Chem. Phys.* **90**, 6254 (1989).
10. G. C. Schatz, *Chem. Phys. Lett.* **150**, 92 (1988).
11. (a) J. M. Launay and B. Lepetit, *Chem. Phys. Lett.* **144**, 346 (1988); (b) B. Lepetit and J. M. Launay, *Chem. Phys. Lett.* **151**, (1988); (c) J. M. Launay and M. LeDourneuf, *Chem. Phys. Lett.* **163**, 178 (1989); (d) J. M. Launay and M. LeDourneuf, *Chem. Phys. Lett.* **169**, 473 (1990); J. M. Launay, *Theor. Chim. Acta* **79**, 183 (1991).
12. (a) J. D. Kress, R. B. Walker, and E. F. Hayes, *J. Chem. Phys.* **93**, 8085 (1990); (b) Z. Darakjian, E. F. Hayes, G. A. Parker, E. A. Butcher, and

J. D. Kress, *J. Chem. Phys.* **95**, 2516 (1991); (c) J. D. Kress and E. F. Hayes, *J. Chem. Phys.* **97**, 4881 (1992); (d) J. D. Kress, R. B. Walker, E. F. Hayes, and P. Pendergast, submitted for publication.

13. B. R. Johnson, *J. Comput. Phys.* **13**, 445 (1973); *J. Chem. Phys.* **67**, 4086 (1977); **69**, 4678 (1978).

14. (a) P. G. Hipes and A. Kuppermann, *Chem. Phys. Lett.* **133**, 1 (1987); (b) Y.-S. M. Wu, S. A. Cuccaro, P. G. Hipes, and A. Kuppermann, *Chem. Phys. Lett.* **168**, 429 (1990); (c) Y.-S. M. Wu, S. A. Cuccaro, P. G. Hipes, and A. Kuppermann, *Theor. Chim. Acta* **79**, 225 (1991).

15. B. J. Archer, G. A. Parker, and R. T. Pack, *Phys. Rev. A* **41**, 1303 (1990).

16. G. A. Parker and R. T. Pack, *J. Chem. Phys.* **98**, 6883 (1993).

17. (a) P. R. Certain and A. S. Dickinson, *J. Chem. Phys.* **49**, 4209 (1968). (b) J. V. Lill, G. A. Parker, and J. C. Light, *J. Chem. Phys.* **89**, 483 (1982); *J. Chem. Phys.* **85**, 899 (1986), (c) J. C. Light, I. P. Hamilton, and J. V. Lill, *J. Chem. Phys.* **82**, 1400 (1984).

18. (a) D. C. Sorensen, "The *k*-step Arnoldi Process," in *Large-scale Numerical Optimization*, edited by T. F. Coleman and Yuying Li (SIAM, Philadelphia, 1990), p. 228; (b) D. C. Sorensen, Z. Tomašić, and P. A. Vu, in *Proceedings, 1993 SCS Simulation Multiconference*, edited by A. Tentner (SCS, San Diego, 1993), p. 149; (c) D. Y. Hu, D. C. Sorensen, and Z. A. Tomašić, in preparation.

19. (a) L. D. Thomas, M. H. Alexander, B. R. Johnson, W. A. Lester, Jr., J. C. Light, K. D. McLenithan, G. A. Parker, M. J. Redmon, T. G. Schmalz, D. Secrest, and R. B. Walker, *J. Comput. Phys.* **41**, 407 (1981), and references therein; (b) G. A. Parker, T. G. Schmalz, and J. C. Light, *J. Chem. Phys.* **73**, 1757 (1980).

20. G. A. Parker, private communication.

21. (a) Z. Bačić, R. M. Whitnell, D. Brown, and J. C. Light, *Comput. Phys. Commun.* **51**, 35 (1988); (b) R. M. Whitnell and J. C. Light, *J. Chem. Phys.* **90**, 1774 (1989); (c) T. J. Park and J. C. Light, *J. Chem. Phys.* **90**, 2593 (1989).

22. Z. Bačić, J. D. Kress, G. A. Parker, and R. T. Pack, *J. Chem. Phys.* **92**, 2344 (1990).

23. J. J. Dongarra, I. S. Duff, D. C. Sorensen, and H. A. van der Vorst, *Solving Linear Systems on Vector and Shared Memory Computers* (SIAM, Philadelphia, 1991).

24. B. N. Parlett, *The Symmetric Eigenvalue Problem* (Prentice–Hall, Englewood Cliffs, NJ, 1980).

25. Y. Saad, *Math. Comput.* **42**, 567 (1984).

26. F. Chatelin and D. Ho, *Math. Modeling Numer. Anal.* **24**, 53 (1990).

27. M. Sadkane, *Numer. Math.* **64**, 181 (1993).