Computing Partial Eigenvalue Sum in Electronic Structure Calculations

Z. Bai* M. Fahey† G. Golub‡ M. Menon§ E. Richter¶

September 25, 1997

Abstract

In this paper, we present an algorithm for computing a partial sum of eigenvalues of a large symmetric positive definite matrix pair. We show that this computational task is intimately connected to compute a bilinear form $u^T f(A)u$ for a properly defined matrix A, a vector u and a function $f(\cdot)$. Compared to existing techniques which compute individual eigenvalues and then sum them up, the new algorithm is generally less accurate, but requires significantly less memory and CPU time. In the application of electronic structure calculations in molecular dynamics, the new algorithm has achieved a speedup factor of 2 for small size problems to 20 for large size problems. Relative accuracy within 0.1% to 2% is satisfactory. Previously intractable large size problems have been solved.

Key words: partial eigenvalue sum, bilinear form, Gauss quadrature, Lanczos method, generalized eigenvalue problem, tight-binding molecular dynamics, Monte Carlo simulation
AMS subject classifications: 65F15, 65F50, 65F30, 65D30

1 Introduction

The central numerical computational problem studied in this paper is to compute the sum of all eigenvalues less than a prescribed value μ of the generalized eigenvalue problem

$$H\Psi = \lambda S\Psi, \tag{1}$$

where H and S are real $n \times n$ symmetric matrices with S positive definite. (H, S) is called a symmetric positive definite matrix pair. λ and Ψ are the eigenvalue and eigenvector, respectively. Specifically, let the eigenvalues $\{\lambda_i\}$ of the matrix pair (H, S) be ordered such that

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_m < \mu < \lambda_{m+1} \leq \cdots \leq \lambda_n,$$

then one wants to compute the sum

$$\tau_\mu = \lambda_1 + \lambda_2 + \cdots + \lambda_m. \tag{2}$$

*Department of Mathematics, University of Kentucky, Lexington, KY 40506, U.S.A. (bai@ms.uky.edu).
†Department of Mathematics, University of Kentucky, Lexington, KY 40506, U.S.A. (mrfahey@ms.uky.edu).
‡Scientific Computing and Computational Mathematics Program, Computer Science Department, Stanford University, Stanford, CA 94305, U.S.A. (golub@sccm.stanford.edu).
§Department of Physics and Astronomy and Center for Computational Sciences, University of Kentucky, Lexington, KY 40506, U.S.A. (madhu@ccs.uky.edu).
¶Department of Physics and Astronomy, University of Kentucky, Lexington, KY 40506, U.S.A. (ernst@ccs.uky.edu).

We are interested in the case where the matrices H and S are large and sparse or structured. The number m of eigenvalues less than μ is unknown, which generally could be very large, say about $n/2$.

The motivation for studying this problem comes from solid state physics where computation of the total energy of an electronic structure requires the evaluation of partial eigenvalue summations. Total energy calculations of solid state systems modeled by tight-binding methods are important in simulating real materials of technological importance. In tight-binding methods, a Hamiltonian matrix H is constructed from parameters that contain all electronic structure information pertaining to the material of interest. In conventional tight-binding schemes, the partial sum of eigenvalues obtained by solving the standard eigenvalue problem $H\Psi = \lambda\Psi$ is used in computing the total energy of the system. This approach, however, has been found to be inadequate in many cases when a higher accurate treatment is required. A generalized tight-binding method [28] has been found to be useful. Here, the evaluation of the total energy involves a partial eigenvalue sum of the generalized eigenvalue problem $H\Psi = \lambda S\Psi$. The overlap matrix S contains additional information pertaining to the system. Specifically, the summations are performed over an ordered set of eigenvalues from the lowest upto a maximum value that is determined by the electron occupancy of the system. We should note that the total energy for a given configuration of electrons and ions, however, is not sufficient to determine the stability of the system which requires that the total energy obtained must also be a minimum. Molecular dynamics simulations, therefore, become necessary to obtain a stable configuration where all the forces acting on individual atoms are zero. The forces can be obtained by either direct calculation from analytic expressions or numerical differentiation of the total energy by making small displacements in the configuration. An iteration process is then used to solve the equations of motion. In many cases, however, realistic simulations require a large number of iterations (usually several hundreds of time steps). Recent interest in properties of large atomic clusters necessitates obtaining partial eigenvalue sums for very large matrices.

Let us now briefly review the existing classes of methods for computing the partial eigenvalue sum τ_μ. The first class of methods can be summarized as *dense and band methods* for explicitly computing all or some of the eigenvalues of a pair of dense or banded matrices. As presented in LAPACK [1], one first computes the Cholesky factorization of $S = LL^T$ and then explicitly transfers the generalized eigenvalue problem (1) to the standard eigenvalue problem $A\Phi = \lambda\Phi$, where $A = L^{-1}HL^{-T}$, and $\Phi = L^T\Psi$. All the eigenvalues of A can be computed by the QR algorithm followed by the summation of the desired eigenvalues. One could also use the bisection method to compute only those eigenvalues less than μ. The LAPACK routine `DSYGV` is an implementation of the QR algorithm (which is the fastest method for computing all eigenvalues), while `DSYGVX` is the LAPACK implementation of the bisection method. The computational complexity of dense methods is $\mathcal{O}(n^3)$ with $\mathcal{O}(n^2)$ storage requirement. When n is large, both computational complexity and memory requirements forbid the use of these algorithms. An improvement of these dense methods is to exploit band structures in the matrices H and S by utilizing a proper sparse matrix reordering scheme. The computational complexity is then reduced to $\mathcal{O}(n^2 b_*)$ with storage requirement $\mathcal{O}(nb_{_H} + nb_{_S})$, where $b_* = \max(b_{_H}, b_{_S})$, $b_{_H}$ is the half-bandwidth of H and $b_{_S}$ is the half-bandwidth of S. The LAPACK routines `DSBGV` and `DSBGVX` are implementations of the QR and bisection methods for banded matrix pairs. The sparse matrix reordering step has to be implemented separately by the user. Parallel implementation of these methods can be found in ScaLAPACK [6]. A novel parallel algorithm which uses high performance BLAS kernels is developed in PRISM project [5].

The second class of methods is based on *sparse matrix techniques*. For example, one can use the Lanczos algorithm with shift-and-invert spectral transformation [12, 20]. This approach applies

the Lanczos algorithm to a sequence of selected shifts to successively compute a few eigenvalues near the shift. However, for each shift ν, a symmetric LDL^T factorization of the shifted system $H - \nu S$ has to be computed along with reorthogonalizing the new initial vector to the previously converged eigenvectors. The storage required is $\mathcal{O}(\xi + nm)$ where ξ is the required storage for the symmetric LDL^T factorization and nm for storing converged eigenvectors. When n and m are large, as well as ξ, memory and CPU time requirements prohibit the use of these methods.

The third class of methods is based upon a *nonlinear optimization approach* to directly compute the partial eigenvalue sum. The following well-known result shows that the partial sum of eigenvalues of a symmetric positive definite pair (H, S) can be characterized by a trace minimization problem:

$$\lambda_1 + \lambda_2 + \cdots + \lambda_m = \min_{U \in \mathcal{U}} \mathrm{tr}(U^T H U),$$

where $\mathcal{U}$ is the set of all $n \times m$ matrices U such that $U^T S U = I_m$ [21, p.191]. A numerical procedure for solving such a trace minimization problem is developed in [31, 11]. However, this method suffers from the fact that a large subspace iteration may be required if one computes a sum of a large number of eigenvalues, which again becomes intractable with large n and m. Furthermore, the number of eigenvalues less than the prescribed value μ is generally unknown in our application.

In this paper, we present a new algorithm which directly computes the partial eigenvalue sum τ_μ and significantly reduces memory and arithemetic costs. We shall show that the computation of the partial eigenvalue sum is related to the computation of the bilinear form $u^T f(A) u$ for a properly defined matrix A, a vector u, and a function $f(\cdot)$, which is defined on the eigenvalues of A. In [16, 17, 2], an efficient algorithm is developed for computing the bilinear form. In this algorithm, the matrix A is only referenced through matrix-vector multiplication and is therefore suited for large sparse or structured problems. Comparing with the techniques reviewed earlier, this new approach is generally less accurate, but requires much less memory and is faster. In the aforementioned application in tight-binding molecular dynamics involving very large systems, the accuracy is acceptable. The computational cost of the new method scales with the cost of a matrix-vector multiplication. For instance, comparing with the QR and bisection methods, speedup factors of 2 for small system sizes ($n = 480$) to 20 for large sizes ($n = 4244$) have been achieved. In practice, we are able to tackle problem sizes previously intractable.

The organization of the rest of the paper is as follows: In Section 2, the electronic structure calculation by tight-binding method is introduced in which one requires the calculation of the partial eigenvalue sum. In Section 3, we describe how the problem of computing a partial eigenvalue sum is connected to that of computing a bilinear form. In Section 4, we review the basic idea of the method for computing the bilinear form $u^T f(A) u$. Section 5 describes a Monte Carlo simulation technique. Section 6 presents the whole algorithm and discusses some computational issues. The applications and performance of the new methods for computing the total energy in condensed matter systems are presented in Section 7. Concluding remarks are in Section 8.

2 Electronic Structure Calculation by tight-binding methods

In this section we briefly review molecular dynamics methods involving electronic structure calculation to which the present method for calculating the partial eigenvalue sum is applied.

We consider a system of N carbon atoms with each atom having a nucleus and six electrons. When bonding with each other, only four electrons from each atom participate in the process and are called valence electrons. In the non-interacting atoms, these valence electrons occupy the free

atom eigenstates called s and p orbitals because of their angular momentum. Furthermore, the electronic eigenstates are characterized according to their spin quantum number (spin up and spin down). The number of eigenstates determines the dimension of the matrices for our computational problem while the partial eigenvalue sum is determined by the number of electrons in the system.

For the carbon system it is a good approximation to neglect the spin dependency of ground state properties of the system. Then, spin up and spin down states are equally occupied and energetically degenerate. This allows us to deal only with half of the possible electronic states (spin up or spin down) and half of the available electrons to occupy them. Spin degeneracy requires only to multiply quantities depending on the number of electrons by a factor of two. Only one s and three p (p_x, p_y, p_z) valence orbitals on each atom have to be considered within this approximation. The dimension of our problem reduces to $4N$ (=number of atomic orbitals) with $2N$ electrons to occupy them. If the atoms are brought together closely enough to form clusters or molecules, the interactions between the atoms results in the formation of new set of states called the molecular orbitals. Within the tight-binding method, these molecular orbitals are approximated by a linear combination of s and p atomic orbitals. In the molecular system there are again a total of $4N$ molecular orbitals (eigenstates of the problem) and $2N$ valence electrons.

The quantum mechanical problem is modeled within a non-orthogonal tight-binding approximation. The electronic part can then be represented by a Hamiltonian matrix H and an overlap matrix S, each of dimension $4N \times 4N$. Both H and S are functions of the atomic coordinates, i.e., $H = H(\{\mathcal{R}\})$ and $S = S(\{\mathcal{R}\})$, where $\{\mathcal{R}\}$ represents the set of all atomic positions and satisfy the eigenvalue equation,

$$H\Psi_k = S\lambda_k\Psi_k, \tag{3}$$

where λ_k is the energy of a single particle state.

The total energy of the system is given by the sum [27]:

$$U = U_{\mathrm{el}} + U_{\mathrm{rep}}, \tag{4}$$

where U_{el} is the electronic contribution to the total energy, obtained by performing a partial sum over the eigenvalues, i.e., sum over the eigen-energies of the occupied electronic states:

$$U_{\mathrm{el}} = 2\sum_{k=1}^{2N} \lambda_k, \tag{5}$$

and U_{rep} is given by a sum over pair potentials

$$U_{\mathrm{rep}} = \sum_{i=1}^{N} \sum_{j>i} \phi(r_{ij}), \tag{6}$$

where ϕ is a simple pair potential term with r_{ij} being the distance between atoms i and j. While the evaluation of the sum in equation (6) is straightforward, computing the sum in equation (5) poses all the computational challenge.

We use molecular dynamics simulations to find the configuration $\{\mathcal{R}\}$ which minimizes the total energy U, i.e. to find the equilibrium state of the system. For this purpose one follows the paths of the atoms in configuaration space which are determined by the classical (coupled) equations of motion. Therefore, one also needs to obtain the forces acting on each atom. They can be derived by taking the partial derivatives of the total energy U with respect to the positions, $x \in \{\mathcal{R}\}$, of each atom. If this is not feasible, numerical differentiation of U_{el} becomes necessary

to obtain the forces. This can be accomplished by making small displacements δx in each of the atomic positions and evaluating the expression

$$\frac{U(x+\delta x)-U(x)}{\delta x}. \tag{7}$$

Molecular dynamics can now be performed by solving the equations of motion from Newton's second law,

$$M\frac{d^2x}{dt^2} = -\frac{\partial U}{\partial x}, \tag{8}$$

where M is the mass of the atom. A small damping term needs to be added to the above equation to drive the system to equilibrium (minimum energy) configuration. The equation can be numerically integrated using predictor-corrector methods [30, 14, 3, 4]. Thus, the molecular dynamics simulations consist of the following steps:

1. Initialize coordinates x;
2. Predictor step;
3. Compute total energy $U = U_{\rm el} + U_{\rm rep}$;
4. Compute the forces;
5. Corrector step.

Steps 2 through 5 are repeated (usually several hundred times) until all the forces are zero.

In this paper, we have applied the new method to systems of carbon atoms due to the current interest in fullerenes and nanotubes. These two systems have, as constituents, carbon atoms with three-fold coordination. The versatility of carbon atoms in forming materials with widely differing properties provides an interesting challenge in formulating theoretical and numerical methods capable of accurate structural determinations in various configurations. Figure 1 shows some typical configurations for carbon systems: (a) C_{60} fullerene, (b) carbon nanotorus and (c) carbon nanotube.

3 Partial Eigenvalue Sum

Given a symmetric matrix $A \in \mathbb{R}^{n\times n}$ and a scalar μ, the crux of the new method for computing the partial eigenvalue sum

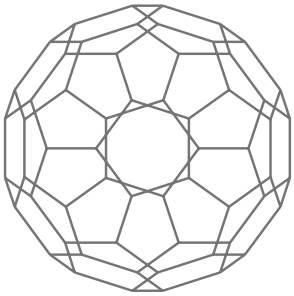
$$\tau_{\mu,A} = \sum_{\lambda_i<\mu} \lambda_i$$

is to construct a function f such that the trace of $f(A)$ approximates the sum $\tau_{\mu,A}$. Specifically, one wants to construct a function f such that

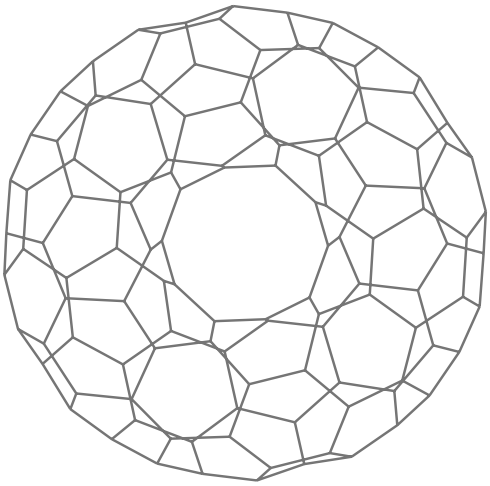
$$f(\lambda_i) = \begin{cases} \lambda_i, & \text{if } \lambda_i < \mu \\ 0, & \text{if } \lambda_i > \mu, \end{cases} \tag{9}$$

for $i = 1, 2, \ldots, n$. Then, we have $\mathrm{tr}(f(A)) = \lambda_1 + \cdots + \lambda_m$. There are many choices of such functions. The simplest choice is $f(\zeta) = \zeta h(\zeta)$, where $h(\zeta)$ is a shifted step function:

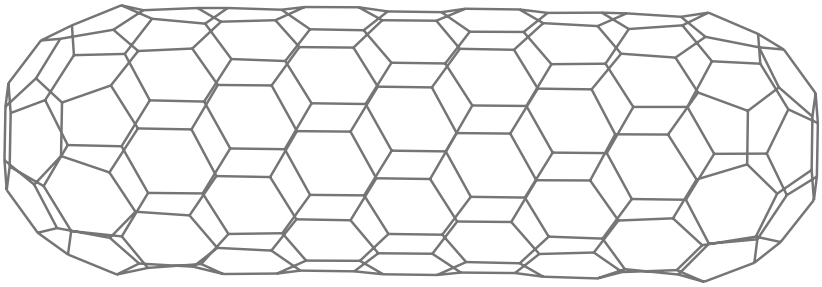
$$h(\zeta) = \begin{cases} 1 & \text{if } \zeta < \mu \\ 0 & \text{if } \zeta > \mu. \end{cases}$$



(a) C60 Fullerene



(b) Nanotorus



(c) Nanotube

Figure 1: Three configurations of carbon systems.

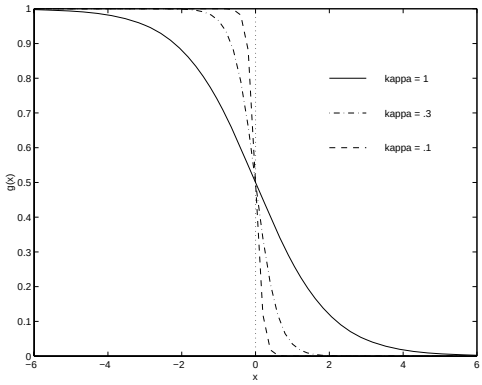


Figure 2: Graphs of $g(\zeta)$ for different values of κ where $\mu = 0$.

Alternatively, we choose f to be of the form

$$f(\zeta) = \zeta g(\zeta) \tag{10}$$

where

$$g(\zeta) = \frac{1}{1 + \exp\left(\frac{\zeta - \mu}{\kappa}\right)},$$

μ and κ are constants. This function, among other names, is known as the Fermi-Dirac distribution function [23, p. 347]. In the context of a physical system, the usage of this distribution function is motivated by thermodynamics. It directly represents thermal occupancy of electronic states. κ is proportional to the temperature of the system, and μ is the chemical potential (the highest energy for occupied states).

It is easily seen that $0 < g(\zeta) < 1$ for all ζ with horizontal asymptotes 0 and 1. $(\mu, \frac{1}{2})$ is the inflection point of g and the sign of κ determines whether g is decreasing ($\kappa > 0$) or increasing ($\kappa < 0$). For our application, we want the sum of all eigenvalues less than μ, so we use $\kappa > 0$. The magnitude of κ determines how "close" the function g maps $\zeta < \mu$ to 1 and $\zeta > \mu$ to 0. As $\kappa \to 0^+$, the function $g(\zeta)$ converges to the step function $h(\zeta)$. The graphs of the function $g(\zeta)$ for $\mu = 0$ and different values of the parameter κ are plotted in Figure 2. Numerically, for example with $\kappa = 0.1$, we have $g(-1) = 9.9995 \times 10^{-1}$, $g(-0.5) = 9.9331 \times 10^{-1}$, $g(0.5) = 6.6929 \times 10^{-3}$ and $g(1) = 4.5398 \times 10^{-5}$.

With this choice of $f(\zeta)$, we have

$$\tau_{\mu,A} = \sum_{\lambda_i < \mu} \lambda_i \approx \mathrm{tr}(f(A)) = \sum_{i=1}^{n} e_i^T f(A) e_i.$$

where e_i is the i-th column of the $n \times n$ identity matrix.

To this end, the problem of computing the partial eigenvalue sum is recast as computing the summation of n bilinear forms $e_i^T f(A) e_i$ for $i = 1, \ldots, n$.

4 Computing the Bilinear Linear Form $u^T f(A) u$

In this section, we review the scheme presented in [7, 16, 17, 2] for computing the bilinear form $u^T f(A) u$. This scheme serves as a basic computational tool for computing the partial eigenvalue

sum. Numerous matrix computation problems can be represented as the problem of computing a bilinear form $u^T f(A)v$. For example, computing an error bound for an approximate solution of a linear systems of equations [7], bounding an element of A^{-1} [16, 17], minimizing a quadratic functional with constraints [18, 17], estimating the parameter in the generalized cross-validation technique [19] and computing $\det(A)$ [2] all can be recast as the problem of computing a bilinear form $u^T f(A)v$.

The main idea of computing the bilinear form $u^T f(A)u$ is to first transform it to a Riemann-Stieltjes integral and then use Gaussian quadrature. This in turn uses orthogonal polynomials and the underlying Lanczos process for the construction of orthogonal polynomials. The idea was originally proposed in [7] and further developed in [16, 17]. In the following, we outline this approach.

Let $A = Q\Lambda Q^T$ be the eigendecomposition of A, where Q is an orthogonal matrix and Λ is a diagonal matrix with increasingly ordered eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. The bilinear form $u^T f(A)u$ can be written as the following:

$$u^T f(A)u = u^T Q f(\Lambda) Q^T u = \tilde{u}^T f(\Lambda)\tilde{u} = \sum_{i=1}^{n} f(\lambda_i)\tilde{u}_i^2, \tag{11}$$

where $\tilde{u} = Q^T u$. Furthermore, the sum can be rewritten as a Riemann-Stieltjes integral

$$u^T f(A)u = \int_a^b f(\lambda)dw(\lambda), \tag{12}$$

where the measure function $w(\lambda)$ is a piecewise constant function defined by

$$w(\lambda) = \begin{cases} 0, & \text{if} \quad a \leq \lambda < \lambda_1 \\ \sum_{j=1}^{i} \tilde{u}_j^2, & \text{if} \quad \lambda_i \leq \lambda < \lambda_{i+1} \\ \sum_{j=1}^{n} \tilde{u}_j^2, & \text{if} \quad \lambda_n < \lambda \leq b. \end{cases} \tag{13}$$

A standard numerical method to compute a Riemann-Stieltjes integral is Gaussian quadrature. For an introduction to Gaussian quadrature, see the references [8, 13]. The Gaussian quadrature formula has the form

$$\int_a^b f(\lambda)dw(\lambda) = \sum_{j=1}^{k} \omega_j f(\theta_j) + \rho[f], \tag{14}$$

where the weights $\{\omega_j\}$ and the nodes $\{\theta_j\}$ are unknown and to be determined. $\rho[f]$ is the integration error (remainder).

Let us recall how the weights and nodes in the quadrature formula are obtained. First, we know that a sequence of polynomials $\{p_i(\lambda)\}_{i=0}^{\infty}$ can be defined such that they are orthonormal with respect to $w(\lambda)$, i.e.,

$$\int_a^b p_i(\lambda)p_j(\lambda)\, dw(\lambda) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise,} \end{cases}$$

where it is assumed that $\int dw(\lambda) = 1$. This sequence of orthonormal polynomials satisfies a three-term recurrence

$$\beta_j p_j(\lambda) = (\lambda - \alpha_j)p_{j-1}(\lambda) - \beta_{j-1}p_{j-2}(\lambda) \tag{15}$$

for $j = 1, 2, ..., k$ with $p_{-1}(\lambda) \equiv 0$ and $p_0(\lambda) \equiv 1$. Writing the recurrence in matrix form, we have

$$\lambda \mathbf{p}(\lambda) = T_k \mathbf{p}(\lambda) + \beta_k p_k(\lambda) e_k, \tag{16}$$

where

$$\mathbf{p}(\lambda)^T = [\, p_0(\lambda) \quad p_1(\lambda) \quad \cdots \quad p_{k-1}(\lambda)\,], \qquad e_k^T = [\,0 \quad 0 \quad \cdots \quad 0 \quad 1\,],$$

and

$$T_k = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{k-2} & \alpha_{k-1} & \beta_{k-1} \\ & & & \beta_{k-1} & \alpha_k \end{pmatrix}.$$

Then in the Gaussian quadrature rule, the eigenvalues of T_k (which equal to the zeros of $p_k(\lambda)$) are the nodes $\{\theta_j\}$. The weights $\{\omega_j\}$ are the squares of the first elements of the normalized eigenvectors of T_k [8].

Note that the Lanczos procedure is a natural and elegant way to compute the orthonormal polynomials $\{p_j(\lambda)\}$ [16, 24]. We shall discuss this further in Section 6. A similar approach can be derived [2, 16, 17] for computing a general bilinear form $u^T f(A)v$.

5 Trace of $f(A)$ and Monte Carlo Simulation

If we choose $u = e_i$ in the bilinear form $u^T f(A)u$ and use Gaussian quadrature to compute an estimate σ_i for $e_i^T f(A)e_i$, then the sum $\sum_{i=1}^n \sigma_i$ is an approximation of $\sum_{i=1}^n e_i^T f(A)e_i$. This provides us with a method of estimating $\mathrm{tr}(f(A))$ since $\sum_{i=1}^n e_i^T f(A)e_i = \mathrm{tr}(f(A))$. However, this approach would require computing n estimates σ_i. If estimating each σ_i costs on the order of n^2, then the total cost would still end up on the order of n^3, the same cost as a dense matrix approach.

To reduce the computational costs, we can use a Monte Carlo simulation technique for estimating $\mathrm{tr}(f(A))$ [2]. This technique is based on the following proposition [22, 9].

Proposition 5.1 *Let H be an $n \times n$ symmetric matrix with $tr(H) \neq 0$. Let $\mathcal{Z}$ be the discrete random variable which takes the values 1 and -1 each with probability 0.5 and let z be a vector of n independent samples from $\mathcal{Z}$. Then $z^T H z$ is an unbiased estimator of $tr(H)$, i.e.,*

$$E(z^T H z) = tr(H)$$

and

$$var(z^T H z) = 2 \sum_{i \neq j} h_{ij}^2,$$

where $E(\cdot)$ denotes expected value (of a random variable) and $var(\cdot)$ denotes variance (of a random variable).

PROOF. Since $z^T H z = \sum_{i,j} z_i h_{ij} z_j$, it immediately follows that $E(z^T H z) = \mathrm{tr}(H)$. Using the symmetry of H, we have $\mathrm{var}(z^T H z) = E[(z^T H z)^2] - [E(z^T H z)]^2 = 2 \sum_{i \neq j} h_{ij}^2$.

An unbiased estimate of $\mathrm{tr}(f(A))$ can be obtained based Proposition 5.1. To do this, we first generate p sample vectors z_k and then apply the approach discussed in Section 4 to estimate the bilinear forms $z_k^T f(A) z_k$ for $k = 1, \ldots, p$. This yields p estimates σ_k of $z_k^T f(A) z_k$ of which we compute the mean giving an unbiased estimate of $\mathrm{tr}(f(A))$, i.e.,

$$\mathrm{tr}(f(A)) \approx \frac{1}{p} \sum_{k=1}^{p} \sigma_k.$$

Note that probabilistic error bounds can be derived for the estimated value [2].

6 The PES Algorithm

In this section, we shall combine the tools established in earlier sections to present an algorithm for estimating the partial eigenvalue sum of a symmetric positive definite pair (H, S). We refer to this as the PES algorithm which simply stands for the Partial Eigenvalue Sum. Given the Cholesky factorization of S: $S = LL^T$, the generalized eigenvalue problem $H\Psi = \lambda S\Psi$ is equivalent to the standard eigenvalue problem

$$(L^{-1}HL^{-T})(L^T\Psi) = \lambda(L^T\Psi).$$

Hence, computing the partial eigenvalue sum of the matrix pair (H, S) is equivalent to computing the partial eigenvalue sum of the matrix $A = L^{-1}HL^{-T}$.

The Lanczos process is a simple yet refined method for computing the orthonormal polynomials as discussed in Section 4. The symmetric Lanczos process satisfies a three-term recurrence

$$\beta_j q_j = (A - \alpha_j)q_{j-1} - \beta_{j-1}q_{j-2}$$

for $j = 1, \ldots, k$, where the initial vector q_0 has unit norm and $\beta_0 \equiv 0$. It is known [16] that the generated Lanczos vectors q_j are orthogonal and are related to the orthogonal polynomials p_j in Section 4 by

$$q_j = p_j(L^{-1}HL^{-T})q_0.$$

Thus, with initial vector $q_0 = u/\|u\|$, k steps Lanczos process generates a tridiagonal matrix T_k. The eigenvalues and squares of the first elements of the normalized eigenvectors of T_k are the nodes and weights in the Gaussian quadrature rule, respectively.

To apply the standard symmetric Lanczos algorithm to the implicitly defined matrix $A = L^{-1}HL^{-T}$, a matrix-vector product routine must be supplied that incorporates L along with H, but the actual Lanczos process need not be modified at all. It is noteworthy that it is possible to use the Lanczos process on $S^{-1}H$, again defined implicitly. This option is valuable when S cannot be factored conveniently [29]. In our current application, the matrix S can be reordered to have narrow half-bandwidth, and the usage of an implicitly defined matrix A is very satisfactory.

The following is a pseudo-code of the PES algorithm which computes an unbiased estimate of the trace of $f(L^{-1}HL^{-T})$. This, in turn, is an unbiased estimate of the sum of the eigenvalues of (H, S) less than a prescribed value μ.

PES Algorithm: Suppose (H, S) is a real $n \times n$ symmetric positive definite pair with the Cholesky factor of S: $S = LL^T$. Given scalars μ and $\kappa > 0$, the function f defined by (10), and the number of sample vectors p. This algorithm produces an estimate $\tilde{\sigma}$ of the partial eigenvalue sum τ_μ defined in (2).

- **For** $k = 1, 2, \ldots, p$
  1. Generate n-vector z_k with elements uniformly distributed on $(-1, 1)$
  2. $z_k := \text{sign}(z_k)$
  3. Compute estimate $\tilde{\sigma}_k$ for $z_k^T f(L^{-1}HL^{-T})z_k$
     (a) Let $r_0 = z_k$, $q_0 = 0$ and $\beta_0 = \sqrt{r_0^T r_0}$
     (b) **For** $j = 1, 2, \ldots$, until convergence
        1. $q_j = r_{j-1}/\beta_{j-1}$
        2. $r_j = L^{-1}HL^{-T}q_j - q_{j-1}\beta_{j-1}$
        3. $\alpha_j = q_j^T r_j$

4. $r_j = r_j - q_j \alpha_j$
5. $\beta_j = \sqrt{r_j^T r_j}$
6. Compute eigenvalues $\{\theta_i\}$ and first elements $\{\omega_i\}$ of the eigenvectors of T_j
7. Compute $\sigma_j = \sum_{i=1}^{j} \omega_i^2 f(\theta_i)$

(c) **Endfor**

(d) $\tilde{\sigma}_k = \beta_0^2 \sigma_j$

- **Endfor**
- $\tilde{\sigma} = \frac{1}{p} \sum_{k=1}^{p} \tilde{\sigma}_k$

A couple of remarks are in order. First, steps 3.(b).1 to 3.(b).5 comprise one iteration of the standard symmetric Lanczos process on the matrix $A = L^{-1} H L^{-T}$. It is important to note that the matrix-vector multiplication in step 3.(b).2 is implicitly implemented, i.e., the matrix A is never formed explicitly. Instead three separate steps are done to compute the matrix-vector multiply: a triangular solve with L^T, a matrix-vector multiply with H and another triangular solve with L.

Second, although the function f is a composite function that involves an exponential, it is evaluated only at the node points used in Gaussian quadrature in Step 3.(b).7. Computing the exponential of a matrix is not required. Thus, the cost of function evaluations is minimal.

6.1 Some computational issues

The following are some additional computational issues encountered with the PES algorithm.

- We have used the Reverse-Cuthill McKee (RCM) reordering scheme from SPARSPAK [15] to reorder the matrix S such that $\widehat{S} = P^T S P$ is banded, where P is a permutation matrix. Basically, the RCM reordering is a reversal of the best ordering obtained by a breadth-first search of a graph of a matrix [26].

- We use the band Cholesky decomposition routine `DPBTRF` from LAPACK to compute the Cholesky decomposition of $\widehat{S}$: $\widehat{S} = \widehat{L}^T \widehat{L}$. Correspondingly, the matrix-vector multiplication at the step 3.(b).4 becomes $\widehat{L}^{-1} P^T H P \widehat{L}^{-T} q_j$. The band triangular system solvers (with coefficient matrices $\widehat{L}^{-T}$ and $\widehat{L}^{-1}$) are available from BLAS (Basic Linear Algebra Subprograms).

- Currently, the coordinate sparse matrix storage format is used for matrices H and S and the RCM reordering routine. In order to use LAPACK's band Cholesky decomposition routine `DPBTRF`, we convert the coordinate format to LAPACK symmetric band format. That is, we convert the storage of $\widehat{S}$ from the coordinate format to a two-dimensional array with $b_S + 1$ rows and n columns, where b_S is the half-bandwidth of $\widehat{S}$. Columns of the matrix $\widehat{S}$ are stored in corresponding columns of the array, and diagonals of the matrix are stored in rows of the array. In summary, our current implementation of the PES algorithm requires $2\eta_H + 2\eta_S + n$ integer storage and $\eta_H + \eta_S + (b_S + 5)n$ real storage, where η_H is the number of nonzeros in the upper triangular part of H and η_S is the number of nonzeros in the upper triangular part of S.

- The computational complexity of the PES algorithm is approximately

$$\underbrace{n(b_s^2+3b_s)}_{\text{band Chol.fact.}} + p\Big[\underbrace{3n}_{\text{initialization}} + \underbrace{\ell(4nb_s+4\eta_H+8n)}_{\text{Lanczos proc.}} + \underbrace{16(\ell^2+\ell)}_{\text{e-values of } T}\Big]$$

  where ℓ is the average number of Lanczos iterations for convergence and p is the number of randonm sample vectors used in the Monte Carlo simulation. In general, p is chosen about 10 to 20. The number of Lanczos iterations is about 15 to 25. Therefore, the computational complexity can be simplified to $\mathcal{O}(nb_s^2 + 4p\ell(nb_s+\eta_H))$. Note that the half-bandwidth b_s of $\widehat{S}$ plays a key role in the performance.

- The following stopping criteria is currently used:

$$|\sigma_j - \sigma_{j-1}| \le \varepsilon|\sigma_j|$$

  where ε is a user specified tolerance value. This criteria tells us that

$$|\sigma - \sigma_j| \le |\sigma - \sigma_{j-1}| + \varepsilon|\sigma_j|.$$

  Therefore, the iteration stops if the error is no longer decreasing or decreasing too slowly. An alternative stopping criterion is to estimate the error term $\rho[f]$ in the quadrature rule. This is subject to further study. In addition, it is good practice for any iterative method to have a parameter *maxit* to limit the maximum number of iterations (i.e., $j < maxit$).

7 Numerical Examples and Discussion

All numerical experiments for computing the partial eigenvalue sum and its applications in tight-binding molecular dynamics were carried out on a Convex Exemplar SPP-1200 system at the University of Kentucky. The algorithm was implemented in Fortran 77 and compiled with the command `fc -O2`. `fc` is the Convex Fortran 9.3 compiler. Program attributes were set by `mpa -m -noparallel -n a.out`. The `mpa` facility modifies attributes of program execution, and here we set it so the programs ran in serial fashion. Also, during compilation, the programs were linked to the ConvexMLIB-LAPACK and ConvexMLIB-VECLIB libraries. ConvexMLIB-LAPACK is derived from the public-domain version of LAPACK and has been specialized for CONVEX computers. ConvexMLIB-VECLIB is a collection of Fortran subprograms optimized for use on the CONVEX family of supercomputers. It contains Level 1 BLAS for vector-vector operations, Level 2 BLAS for matrix-vector operations and Level 3 (Extended) BLAS for matrix-matrix operations. The tolerance value ε for the stopping criteria of the algorithm is set to 5.0×10^{-4}.

We have performed total energy calculations on large carbon systems that include fullerenes, nanotubes, "knee" clusters and network clusters (building blocks for bulk diamond structure). These structures are now routinely produced by experimentalists. Their structure determination constitutes an important task for theoretical investigations. The generalized tight-binding method is found to be very accurate for treating these systems [27]. The Hamiltonian and overlap matrices H and S are constructed using parameters for pure carbon. In this model, each atom is characterized by 4-orbitals. As a general rule, for a system containing N atoms, one needs to construct $4N \times 4N$ matrices for both H and S that are very sparse. Due to the dependence of S on H in the model, H and S have the same sparsity pattern.

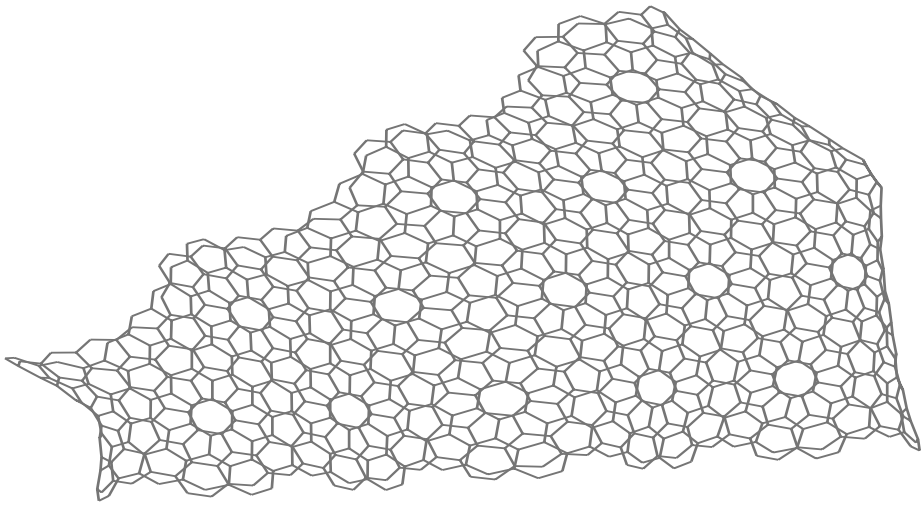


Figure 3: Carbon cluster that forms part of a "knee" structure connecting nanotubes of different diameters.

In the following, we present numerical results for two carbon systems. The first systems is a carbon cluster that forms part of a "knee" structure. The second systems is a carbon network cluster that forms part of a bulk diamond structure. For these systems, we have investigated the distribution of eigenvalues, the RCM reordering scheme and comparisons of the PES method to the QR and bisection methods available in LAPACK. In addition, the PES method is also compared to band methods and sparse eigensolver methods.

Knee Cluster: Consider a carbon cluster which forms part of a "knee" structure, connecting nanotubes of different diameters [10]. Carbon nanotubes are known to exhibit metallic or semiconducting properties depending on their diameters. These "knee" junctions provide an interesting microscopic metal-semiconductor contact and are of considerable technological interest. The "knee" structure shown in Figure 3 was used to construct several test pairs of various sizes. Figure 4 shows the eigenvalue distribution for the cluster containing 60, 120 and 240 atoms. Accordingly, the matrix sizes are 240, 480 and 960, respectively. There is a clear gap between eigenvalues that corresponds to the highest occupied and lowest unoccupied states.

Application of the RCM reordering algorithm on the matrix S produced half-bandwidths approximately equal to $b_S = n/9$. For example, with system size $n = 960$, the reordered matrix $\widehat{S}$ has a half-bandwidth of 110.

Table 1 lists performance statistics of the QR method, the bisection method, and the PES algorithm using $p = 10$ sample vectors. The first column of the table shows the order n of the matrix pair (H, S). The second column shows the number m of eigenvalues to the left of the gap at the origin; i.e., m is the number of eigenvalues to be summed. We see that approximately 65% of the total eigenvalues are required for the sum. The third column is the partial eigenvalue sum obtained either by using the QR method or the bisection (BI) method (the quantities computed by two methods agree up to 12 decimal digits). The fourth and fifth columns are the CPU times (in seconds) of the QR and bisection methods. The sixth and seventh columns are the estimated partial eigenvalue sum and the CPU time of the PES method, respectively. The final column shows the percentage of the relative error in the approximation.

From the table, we see that the approximated partial sums are within a range of 0.01% to 2.2%

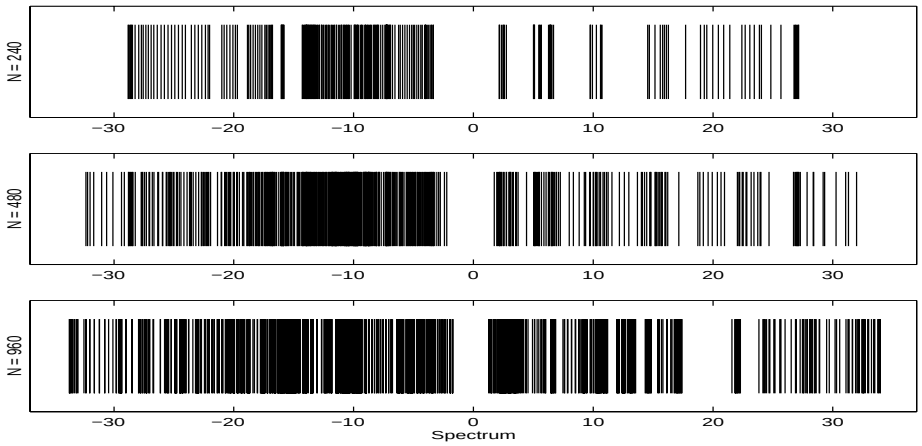


Figure 4: Distribution of eigenvalues in the torus carbon structure for several system sizes. Notice the gap around 0 between the eigenvalues.

Table 1: Performance of PES method vs. dense methods on Convex Exemplar SPP-1200. Here, 10 Monte Carlo samples were used to obtain estimates for each systems size.

		Dense methods			PES		% Relative
n	m	Partial Sum	QR Time	BI Time	Estimate	Time	Error
480	349	-4849.8	7.4	7.6	-4850.2	2.8	0.01
960	648	-9497.6	61.9	51.8	-9569.6	18.5	0.7
1000	675	-9893.3	80.1	58.6	-10114.1	22.4	2.2
1500	987	-14733.1	253.6	185.6	-14791.8	46.4	0.4
1920	1249	-18798.5	548.3	387.7	-19070.8	72.6	1.4
2000	1299	-19572.9	616.9	431.8	-19434.7	78.5	0.7
2500	1660	-24607.6	1182.2	844.6	-24739.6	117.2	0.5
3000	1976	-29471.3	1966.4	1499.7	-29750.9	143.5	0.9
3500	2276	-34259.5	3205.9	2317.4	-33738.5	294.0	1.5
4000	2571	-39028.9	4944.3	3553.2	-39318.0	306.0	0.7
4244	2701	-41299.2	5915.4	4188.0	-41389.8	339.8	0.2

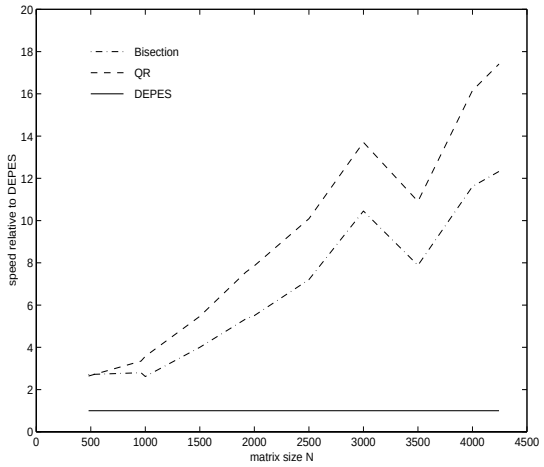


Figure 5: Speed of finding the partial eigenvalue sum relative to PES based on data in Table 1.

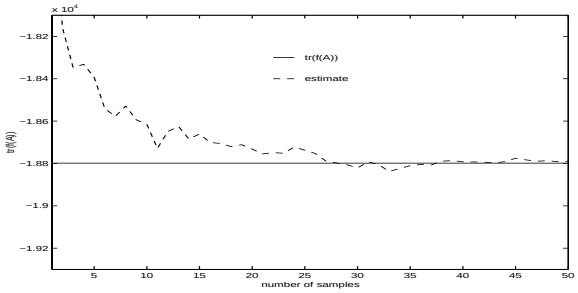


Figure 6: Monte Carlo simulation of PES method for $p = 50$ sample vectors.

relative error to the "exact" value computed by the QR or bisection method. This approximation is acceptable in practice. Figure 5 shows the speed of computing the partial eigenvalue sum relative to the PES algorithm. The horizontal axis is matrix dimension, and the vertical axis is a method's CPU time divided by the time for PES . Therefore the PES curve is a horizontal line at 1, and the other curves measure how many times slower the QR and bisection methods are than PES . From the figure, we see that PES method is more than a factor of 2 times faster than QR and bisection methods for small problems. For larger problems, the PES method is about 12 times faster than the bisection method and 17 times faster than the QR method. The total savings in CPU time is significant since the partial eigenvalue sum is required to be repeatly computed as many as a hundred times in our application.

We note that the RCM reordering procedure takes a significant portion of the total CPU time in the PES algorithm. It consumes approximately 66% of the CPU time for $n > 2500$. Even if we were to increase the number of samples to $p = 20$ to improve the estimate in Monte Carlo simulation, then the RCM reordering procedure still takes approximately 50% of the total CPU time.

Figure 6 shows typical stochastic behavior we observed with matrix order $n = 1920$ and $p = 50$ samples. As with any stochastic technique, a large number of samples usually result in better approximations. Using additional sample vectors only increases computational time linearly. The RCM reordering and Cholesky factorization costs are incurred only once.

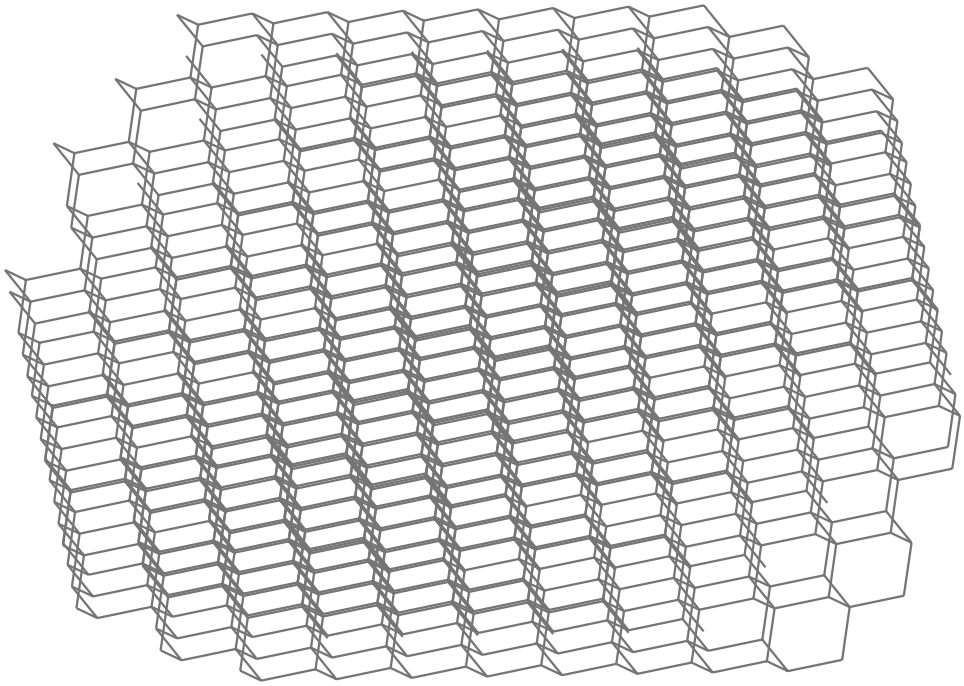


Figure 7: Carbon network cluster of 1156 atoms that forms part of a bulk diamond structure

Bulk Diamond Cluster: We have applied the PES method to the 1156-atom carbon network cluster shown in Figure 7. This cluster forms part of a bulk diamond structure, one of the most stable forms of carbon. The carbon atoms in the interior of this cluster are arranged so that each has four neighbors. The system has a relatively large gap in the eigenvalue spectrum.

For the bulk diamond structure of size $n = 4624$, the matrices H and S have 437,184 nonzero entries. The RCM reordered matrix $\widehat{S}$ has a half-bandwidth of $b_s = 1468$, approximately one-third the order of the matrix. For this example, the number of eigenvalues to the left of the gap was $m = 2587$, or 56% of all eigenvalues. Using the QR method to compute all of the eigenvalues, it took 87 CPU minutes. The computed partial eigenvalue sum was $\tau_\mu = -43967.3$. The PES method with $p = 10$ samples produced an estimate of -43843.6 with relative error 0.2% in 17 CPU minutes. The RCM reordering procedure took 11 minutes of the total 17 minutes. With $p = 20$ samples, PES produced an estimate of -43951.7 in 21 minutes. The relative error is less than 0.1%.

We should note that the speedup of the PES method was not as significant for the bulk diamond structure as it was for the "knee" structure. We observe that for the bulk diamond structure, the half-bandwidth b_s of the reordered matrix $\widehat{S}$ is much larger than in the "knee" structure. It is about one third of the matrix dimension. As we discussed in section 6, this is one of the key factors responsible for the performance of the PES method. An efficient reordering scheme to reduce the bandwidth is certainly highly desired.

Finally, let us briefly report the performance of two other approches we have investigated. The first approach explored band structure in both matrices H and S. The method has four steps:

1. Use the RCM reordering algorithm so that $\widehat{S} = P^T S P$ is banded. Since H has the same sparsity structure in our application, then $\widehat{H} = P^T H P$ has the same half-bandwidth.

2. Compute the band Cholesky decomposition of $\widehat{S}$: $\widehat{S} = LL^T$.

3. Form the banded matrix $A = Z^T L^{-1} \widehat{H} L^{-T} Z$, where Z is chosen to preserve the band structure of $\widehat{H}$.

4. Use the band QR algorithm to compute all the eigenvalues of A or the bisection method to compute only the desired eigenvalues of A.

The implementation of this method is easily done with LAPACK. A performance profile for the "knee" cluster showed that step 3 took more than 60% of the total CPU time of this approach. This step is even significantly slower than the direct dense transformation without exploiting band structure although a dense transformation takes significantly more flops. As a result, this band method can be from 1.5 times to 2 times slower than the dense QR or bisection methods, for matrix sizes as small as 480. This in turns is 3 to 30 times slower than the PES method.

The second approach we have investigated is to use a sparse eigensolver. For example, we used ARPACK [25] to find a few of the algebraically smallest eigenvalues, then used deflation to find the next few eigenvalues, and repeated until all desired eigenvalues are found. We have observed that for computing just the 6 algebraically smallest eigenvalues with $\texttt{tol} = 10^{-3}$ with the matrix size $n = 960$ in the "knee" cluster, ARPACK takes about 1/3 of the total CPU time of the PES method. Since we need to find about half of eigenvalues, we expect that this approach would not be competitive in terms of speed and memory.

8 Concluding Remarks

A new method for computing a partial eigenvalue sum of a symmetric definite pair of matrices is presented in this paper. We have shown that the PES method is significantly faster than those methods which directly compute individual eigenvalues and then sum them up. However, this new approach is generally less accurate. But for our application of electronic structure calculations in molecular dynamics, the achieved accuracy is acceptable. This method could easily be modified to estimate a partial eigenvalue sum over a specified interval $[\alpha, \beta]$. The estimates in Monte Carlo simulation can be improved by using a better sampling technique. It deserves further study.

Acknowledgement

Z. B. was supported in part by an NSF grant ASC-9313958, an DOE grant DE-FG03-94ER25219 and an DARPA grant DM28E04120 and P-95006 via a subcontract from Argonne National Laboratory. M. F. was supported in part by a fellowship from the University of Kentucky Center for Computational Sciences. G. G. was supported by NSF grant DMS-9403899. M. M. and E. R. were supported in part by NSF grant OSR 94-52895 and by the University of Kentucky Center for Computational Sciences. Part of this research was carried out using the facilities of the High-Performance Computing Laboratory of the College of Arts and Sciences at the University of Kentucky, supported in part by NSF Grant DMS-9508543.

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide (second edition)*, SIAM, Philadelphia, 1995.

[2] Z. Bai, M. Fahey, and G. Golub, *Some large scale matrix computation problems*, J. of Comput. and Appl. Math., 74 (1996), pp. 71–89.

[3] D. Beeman, *Some multistep methods for use in molecular dynamics calculations*, J. Comp. Phys., 20 (1976), pp. 130–139.

[4] H. J. C. Berndsen and W. F. van Gunsteren, *Molecular Dynamics Simulation of Statistical Mechanical Systems*, North-Holland, Amsterdam, 1986.

[5] C. Bischof, S. Huss-Lederman, X. Sun, A. Tsao, and T. Turnbull, *Parallel performance of a symmetric eigensolver based on the invariant subspace decomposition approach*, in Scalable High Performance Computing Conference 1994, IEEE Computer Society, 1994.

[6] L. S. Blackford, J. Choi, A. Cleary, D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. Whaley, *ScaLAPACK User's Guide*, SIAM, Philadelphia, 1995.

[7] G. Dahlquist, S. Eisenstat, and G. H. Golub, *Bounds for the error of linear systems of equations using the theory of moments*, J. Math. Anal. and Appl., 37 (1972), pp. 151–166.

[8] P. Davis and P. Rabinowitz, *Methods of Numerical Integration*, Academic Press, New York, 1984.

[9] S. Dong and K. Liu, *Stochastic estimation with z_2 noise*, Physics Letters B, 328 (1994), pp. 130–136.

[10] B. I. Dunlap, *Relating carbon tubules*, Phys. Rev. B, 49 (1994), pp. 5643–5650.

[11] A. Edelman, T. Arias, and T. Smith, *The geometry of algorithms with orthogonality constraints*, SIAM J. on Mat. Anal. Appl. to appear.

[12] T. Ericsson and A. Ruhe, *The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems*, Math. of Comp., 35 (1980), pp. 1251–1268.

[13] W. Gautschi, *A survey of Gauss-Christoffel quadrature formulae*, in E. B. Christoffel – the Influence of His Work on on Mathematics and the Physical Sciences, P. L. Bultzer and F. Feher, eds., Birkhauser, Boston, 1981, pp. 73–157.

[14] C. W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Engelwood Cliffs, NJ, 1971.

[15] J. A. George and J. W. H. Liu, *The design of a user interface for a sparse matrix package*, ACM Trans. Math. Software, 5 (1979), pp. 139–162.

[16] G. Golub and G. Meurant, *Matrics, moments and quadrature*, in Proceedings of the 15th Dundee Conference, June 1993, D. F. Griffiths and G. A. Watson, eds., Longman Scientific & Technical, 1994.

[17] G. Golub and Z. Strakoš, *Estimates in quadratic formulas*, Numerical Algorithms, 8 (1994), pp. 241–268.

[18] G. H. Golub and U. von Matt, *Quadratically constrained least squares and quadratic problems*, Numer. Math., 59 (1991), pp. 561–580.

[19] ——, *Generalized cross-validation for large scale problems*, Technical Report SCCM-96-05, Stanford University, Stanford, 1996.

[20] R. Grimes, J. Lewis, and H. Simon, *A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 228–272.

[21] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge Univ. Press, Cambridge, United Kingdom, 1985.

[22] M. Hutchinson, *A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines*, Commun. Statist. -Simula., 18 (1989), pp. 1059–1076.

[23] K. Kerstin and J. R. Dorman, *A Course in Statistical Thermodynamics*, Academic Press, New York, 1971.

[24] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Natl. Bur. Stand, 45 (1950), pp. 225–280.

[25] R. B. Lehoucq, D. C. Sorenson, and C. Yaag, *ARPACK Users' Guide: Solution of Large Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods*, July 1996. Draft.

[26] W. Liu and A. Sherman, *Comparative analysis of the Cuthill-McKee and the Reverse Cuthill-McKee ordering algorithms for sparse matrices*, SIAM J. Numer. Anal., 13 (1976), pp. 198–213.

[27] M. Menon, E. Richter, and K. R. Subbaswamy, *Structural and vibrational properties of fullerenes and nanotubes in a nonorthogonal tight-binding scheme*, J. Chem. Phys., 104 (1996), pp. 5875–5882.

[28] M. Menon and K. Subbaswamy, *Transferable nonorthogonal tight-binding scheme for silicon*, Phys. Rev. B, 50 (1994), pp. 11577–11582.

[29] B. Parlett, *The Symmetric Eigenvalue Problem*, Prentice Hall, Englewood Cliffs, NJ, 1980.

[30] D. C. Rapaport, *The Art of Molecular Dynamics Simulation*, Cambridge University Press, 1995.

[31] A. Sameh and J. A. Wisniewski, *A trace minimization algorithm for the generalized eigenvalue problem*, SIAM J. Numer. Anal., 19 (1982), pp. 1243–1982.