A general parallel sparse-blocked matrix multiply for linear scaling SCF theory.[1]

Matt Challacombe
Los Alamos National Laboratory,
Theoretical Division
Group T-12, MS B268
Los Alamos, New Mexico 87545
EMail: MChalla@T12.LANL.Gov
(October 1, 1999)

A general approach to the parallel sparse-blocked matrix-matrix multiply is developed in the context of linear scaling self-consistent-field (SCF) theory. The data-parallel message passing method uses non-blocking communication to overlap computation and communication. The space filling curve heuristic is used to achieve data locality for sparse matrix elements that decay with "separation". Load balance is achieved by solving the bin packing problem for blocks with variable size.

With this new method as the kernel, parallel performance of the simplified density matrix minimization (SDMM) for solution of the SCF equations is investigated for RHF/6-31G** water clusters and RHF/3-21G estane globules. Sustained rates above 5.7 GFLOPS for the SDMM have been achieved for $(H_2O)_{200}$ with 95 Origin 2000 processors. Scalability is found to be limited by load imbalance, which increases with decreasing granularity, due primarily to the inhomogeneous distribution of variable block sizes.

PACS Numbers: 31.15.A, 02.60, 36.40.C, 36.20.K

I. INTRODUCTION

All aspects of self-consistent-field (SCF) theory that employ Gaussian Type Linear Combination of Atomic Orbital (GT-LCAO) basis functions have now demonstrated a computational complexity for non-trivial (3-dimensional) insulating systems that is linear with system size, *i.e.* $\mathcal{O}(N_{\rm Atoms})$. This includes computation of the Coulomb matrix [1–4], the exchange matrix [5–9], the exchange-correlation matrix [10,11], and solution of the SCF equations [11,12].

The GT-LCAO basis functions are choice for linear scaling SCF theory. This is because they are compact local functions, they are well understood [13–15], they are able to achieve systematic convergence to the complete basis set limit [16–23], and because they allow the analytic evaluation of many properties and their corresponding molecular integrals [24–26].

To achieve an $\mathcal{O}(N_{\rm Atoms})$ computational effort, linear scaling GT-LCAO methods employ a number of advanced algorithms, combined with careful approximations [27,28], to exploit the locality of short-range quantum interactions and to carry out the fast evaluation of long-range sums.

While progress in reducing the cost of linear scaling methods continues at a good pace, the development of efficient parallel methods for linear scaling GT-LCAO SCF theory has been less satisfactory, but no less important. With increasing molecular size and complexity, the magnitude of statistical mechanical effects grow. Thus, the relevance of large scale electronic structure methods is tightly coupled with the ability to perform rapid configurational sampling. Parallel implementation of linear scaling GT-LCAO SCF methods offers a clear path to achieving this relevance.

A common approach to linear scaling is to avoid the diagonalization bottleneck through non-linear optimization of a sparse density matrix [11,12,29–39] or set of local orbitals [40–48]. See Ref. [49–51] for recent reviews. Central to these methods is the sparse matrix-matrix multiply. To date, the computer science community has explored only theoretical aspects of the parallel sparse matrix-matrix multiply [52–54], while many parallel linear scaling methods employ highly optimized versions that use non-standard data structures and make *a priori* assumptions about the matrix structure to achieve load balance and reduce the global redistribution of data [36,43,46] .

In this paper, a general algorithm for a parallel sparse-blocked matrix-matrix product is presented, and heuristics for achieving data locality and load balance are explored in the context of the Simplified Density Matrix Minimization (SDMM) [12] for linear scaling solution of the SCF equations. In the following, a more detailed background is presented and arguments for a general parallel sparse-blocked matrix-matrix multiply are made. Next, a parallel implementation of this general approach is developed, and methods for achieving data locality and load balance are filled out. Then, performance of the parallel SDMM is demonstrated and discussed for a series of water clusters and estane globules. Finally, conclusions are drawn.

[1]To appear in the special *Parallel Computing in Chemical Physics* issue of Computer Physics Communications. Cite this preprint as LA-UR-99-5255.

II. BACKGROUND

A. Linear scaling solution of the SCF equations

For insulating systems, those with an appreciable highest occupied molecular orbital-lowest occupied molecular orbital (HOMO-LUMO) gap, linear scaling solution of the SCF equations may be achieved through the solution of sparse matrix equations, with computation dominated by either the sparse matrix-vector or sparse matrix-matrix product. These sparse matrix equations are obtained by dropping small matrix elements that result from the approximate exponential decay of density matrix elements (or orbital coefficients) with separation of local basis function centers. This principle is illustrated in Fig. 1, which shows an approximately exponential decay in atom-blocks of the density matrix with basis function separation for a cluster of 110 water molecules at the restricted Hartree-Fock (RHF) 3-21G level of theory.

There are a number of ways to achieve matrix sparsity, and these have an effect on the matrix structure, ratio of accuracy to efficiency, and corresponding parallelization strategies. For the study of homogeneous systems such as silicon, sparsity is typically achieved by imposing a distance criterion [49–51],

$$P_{ab} = \begin{cases} 0 & |\mathbf{A}-\mathbf{B}| > R_{\mathrm{cut}}\,, \\ P_{ab} & \text{else} \end{cases} \tag{1}$$

where $\mathbf{A}$ and $\mathbf{B}$ are centers corresponding to the basis function indices a and b, and R_{cut} is the localization radius. This leads to an *a priori* knowledge of the matrix structure (the graph), with in principle all associated matrices having the same graph.

Examining Fig. 1, we see that a given separation may correspond to atom-blocks of the density matrix that span several orders of magnitude. This is very different from the small scatter observed in isotropic systems [51]. One may envision even more extreme cases such as a protein-bound heme, where the heme sub-matrix is completely delocalized, and the elements corresponding to the protein fall off much more rapidly. This suggests an alternative strategy, advocated here, which is to filter out negligible blocks after each sparse-matrix operation. For a given level of accuracy, this approach may lead to matrices that are more sparse, but has several disadvantages. First, foreknowledge of the matrix structure is lost and different matrices will tend to nonuniform graphs. Also, additional effort is required for recomputation of the matrix structure (the symbolic matrix) after each operation.

Recomputation of the symbolic matrix, combined with indirect addressing of individual matrix elements leads to poor scalar performance for standard sparse matrix kernels. One of the most effective strategies for improving the performance of sparse matrix-vector and matrix-matrix products is to employ locally dense blocks

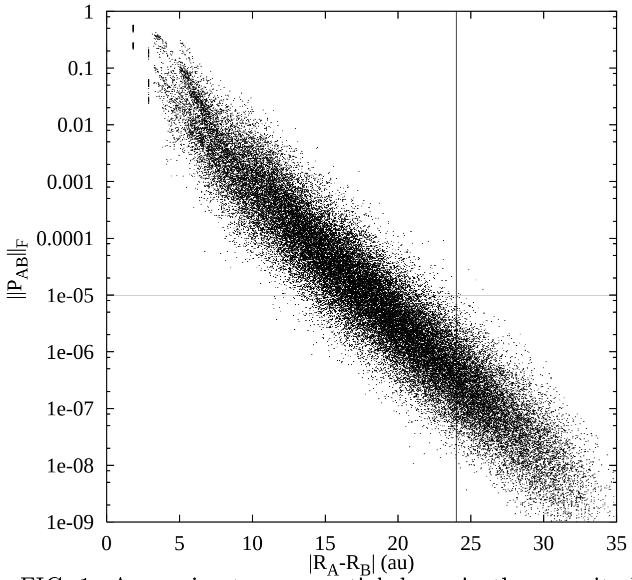

FIG. 1. Approximate exponential decay in the magnitude of density matrix blocks, $||P_{AB}||_F$, with separation $|\mathbf{R}_A - \mathbf{R}_B|$ of LCAO basis functions A and B for RHF/3-21G $(H2O)_{110}$. The horizontal line corresponds to application of Eq. (4) with a drop tolerance `Thresh%Trix` $= 10^{-5}$, while the vertical line corresponds to use of Eq. (1) with a localization radius $R_{\mathrm{cut}} = 24$ au.

where possible. This approach has been used in a number of fields including finite elements [55–57], reactive scattering [58], linear scaling electronic structure theory [12,36,43,46,59], and notably included in the PETSc parallel scientific library [60]. Because methods for dense linear algebra have reached a high level of sophistication [61–64], it is possible to achieve significant efficiencies and still maintain a globally sparse representation.

While parallel methods for the sparse matrix-vector product are well developed [55–57,65–67], general methods for the sparse matrix-matrix product are not. Indeed, an extensive literature search has turned up only theoretical work on this subject [52–54]. Part of the difficulty in the development of a parallel sparse matrix-matrix product is that the matrices being multiplied will, in general, have different structures. Thus, the graph theoretical methods [55,56,66,68] that work well for achieving data locality and load balance in the parallel sparse matrix-vector product, are not generally applicable.

B. A simplified density matrix minimization

As introduced in Ref [12], the Simplified Density Matrix Minimization (SDMM) minimizes the Li, Nunes, Vanderbilt [29] and Daw [30] (LNVD) objective,

$$\Omega = \mathrm{Tr}\left\{(3\mathbf{PP} - 2\mathbf{PPP})\,\mathbf{F}\right\} + \mu\left(N_{\mathrm{el}} - \mathrm{Tr}\mathbf{P}\right)\,, \tag{2}$$

where $\mathbf{P}$ is the orthogonal density matrix, $\mathbf{F}$ is the orthogonal Fock matrix, and N_{el} is the number of electrons.

In the SDMM, the congruence transformations from an LCAO basis to an orthogonal representation and back are carried out entirely with $\mathbf{Z}$, the inverse Cholesky factor of the overlap matrix. While congruence transformations of this sort are a standard method for solving generalized problems [69,70], using an incomplete linear solve to compute a sparse inverse Cholesky factor introduces two levels of error (one for the factorization and one for the inversion), and is costly unless the graph of the inverse is known [71,72]. In this work and in Ref. [12], $\mathbf{Z}$ is computed with the AINV algorithm of Benzi and co-workers [73–76]. This is different from the approach of Millam and Scuseria [34] in that AINV computes $\mathbf{Z}$ directly using an incomplete method to preserve sparsity. While quite inexpensive, it should be mentioned that AINV is $\mathcal{O}(N)$ without an elimination tree [77], and several strategies exist for parallelization [75,76].

The SDMM obtains the SCF by minimizing Eq. (2) with the non-linear conjugate gradient (NLCG) method employing simplified matrix equations and obtaining improved convergence. The simplifications arise by noticing that, with the starting density matrix $\mathbf{P}_0 = (N_{\rm el}/N_{\rm bas})\,\mathbf{I}$, symmetry of the converged result, $[\mathbf{P}, \mathbf{F}] = 0$, is preserved during minimization. This commutation leads to simplified forms of the equations that require fewer matrix-matrix multiplies per NLCG iteration than LNVD. More importantly, the SDMM converges more rapidly than LNVD, and after some (3-10) NLCG cycles, purification [78] leads to quadratic convergence of the density matrix. This method has many similarities to the method recently introduced by Palser and Manolopoulos [39].

C. Sparse-blocked linear algebra

A general sparse-blocked linear algebra library has been developed in Fortran 90 using derived types to ease programming and improve clarity. The fundamental types are the blocked compressed sparse row matrix (`BCSR`), and the distributed blocked compressed sparse row matrix (`DBCSR`). These types are related to the popular compressed sparse row (CSR) structures common to many sparse linear algebra methods, but in addition to row and column indices, `BCSR` and `DBCSR` matrices also have an index that points to the start of each dense block. Thus, each matrix $\mathbf{A}$ consists of a symbolic (`INTEGER`) component $\mathbf{A}^{\rm s}$, and a numeric (`REAL*8`) component $\mathbf{A}^{\rm n}$. For all but a minimal (STO-NG) basis, the memory and effort required to store and manipulate the symbolic component is negligible.

Sparse-blocked data structures are not new, and have been developed and used to significant advantage in other linear scaling electronic structure efforts [12,36,43,46,59], finite element methods [55–57], and reactive scattering theory [58]. A unique feature of the `BCSR` and `DBCSR` types is the direct correspondence with standard methods of sparse linear algebra that employ the CSR data structure *and* the support of arbitrary block sizes. This allows access to sparse linear algebra methods that have been developed to operate on CSR matrices [79,66] and extension to variable block sizes, corresponding to different basis sets, atom-types, etc. For example, both the serial `BCSR` and the parallel `DBCSR` sparse-blocked matrix-matrix multiply,

$$\mathbf{C} = \alpha \mathbf{C} + \beta \mathbf{A}\,\mathbf{B}\,, \tag{3}$$

are carried out with a modified version of Gustavson's method [80] for the CSR data structure, and use the same subroutine. This routine is the computational kernel of the SDMM, and employs the dense Double precision GEneralized Matrix Multiply (DGEMM) to accumulate products of the sparse blocks. While many current implementations of the DGEMM are able to achieve theoretical speeds, these have been optimized for very large matrix dimensions (> 100), and so are non-optimal for the matrix dimensions typically encountered in atom blocks of the first and second row of the periodic table (~ 10). To address this in a general way the customizable DGEMM `mm_double_NN_1` is used, which can be automatically tuned for any platform or matrix dimension(s) with the PHiPAC software [62–64].

Sparsity is enforced using the block-wise filter

$$P_{ab} = \begin{cases} 0 & ||\mathbf{P}_{AB}||_{\rm F} < \texttt{Thresh\%Trix} \\ P_{ab} & \text{else} \end{cases} \tag{4}$$

where `Thresh%Trix` is a drop-tolerance, and $||A||_{\rm F} = \sqrt{{\rm Tr}\left\{\mathbf{A}^T\mathbf{A}\right\}}$ is the Frobenius norm. See Fig. 1 for illustration. This permits an arbitrary and continuous control of error. It also allows the sparse matrix structure to fill optimally during the SDMM, while the use of blocks minimizes the associated cost of indirect addressing and computation of the symbolic matrix. As the magnitude of individual matrix elements decreases with increasing basis set size [81], typical values of `Thresh%Trix` may be transferable between basis sets as $||.||_{\rm F}$ is a measure of all elements in a block.

III. PARALLEL SPARSE-BLOCKED MATRIX-MATRIX MULTIPLY

An optimal method for performing the sparse-blocked matrix-matrix multiply in parallel will realize two interdependent objectives:

1 Minimize communication overhead.

2 Evenly distribute the workload.

In the parallel dense matrix multiply, achieving these objectives is trivial as the workload is regular and the work to communication ratio scales as $\mathcal{O}(N^2_{\rm Basis}/N_{\rm Proc})$. Here, the workload is irregular and the work to communication ratio scales as $\mathcal{O}(N_{\rm Basis}/N_{\rm Proc})$.

Communication overhead is associated with the overlap between local and non-local components of DBCSR matrices. This overhead is minimized by (**a**) overlaying communication and computation, and (**b**) improving data locality through ordering with the space filling curve (SFC) heuristic. The DBCSR matrices are decomposed row-wise and load balance is approached by sectioning the SFC so as to evenly distribute the number of non-zero matrix elements of a representative matrix. This approach has been pursued with a data parallel, message passing paradigm using MPI [82].

A. Algorithm

The DBCSR matrix is distributed row-wise, with both local and global row and column indices included in the symbolic component. The global indices are necessary for computation of overlapping regions between processors, allowing optimization of memory and communication overheads. Using the following informal notation, the DBCSR matrix is the sum of all symbolic and numeric components on each processor:

$$\mathbf{A} = \sum_{p}^{N_{\mathrm{Proc}}} \left\{ \mathbf{A}^{\mathbf{s}}_{p}, \mathbf{A}^{\mathbf{n}}_{p} \right\} . \tag{5}$$

In the following, q will indicate the a local component, and p will indicate a non-local component.

To minimize communication overhead, nonblocking methods are used that employ the immediate receive MPI_IRECV. Once the receive has been posted, control returns allowing buffers to fill while computation occurs. In this way, local work and non-local data transmission may be overlapped. All possible tasks are performed before posting a wait. Of these tasks, the symbolic matrix multiply is the least CPU intensive, involving

$$\mathbf{C}^{\mathrm{s}}_{q} = \alpha \mathbf{C}^{\mathrm{s}}_{q} + \mathbf{A}^{\mathrm{s}}_{q} \, \mathbf{B}^{\mathrm{s}}_{q} \, , \tag{6}$$

followed by

$$\mathbf{C}^{\mathrm{s}}_{q} = \mathbf{C}^{\mathrm{s}}_{q} + \sum_{p \neq q}^{N_{\mathrm{Proc}}} \mathbf{A}^{\mathrm{s}}_{q} \, \mathbf{U}^{\mathrm{s}}_{p} \, , \tag{7}$$

where $\mathbf{U}_p$ is the exact subset of $\mathbf{B}_p$ that overlaps with $\mathbf{A}_q$. Equation (7) may be carried out without communication because the symbolic component $\mathbf{U}^{\mathrm{s}}_{p}$ is computed from the local row and column indices of $\mathbf{A}^{\mathrm{s}}_{p}$, and the global row and column indices of $\mathbf{B}^{\mathrm{s}}$. The most CPU intensive local operation is the local-local numeric matrix multiply

$$\mathbf{C}^{\mathrm{n}}_{q} = \alpha \mathbf{C}^{\mathrm{n}}_{q} + \beta \mathbf{A}^{\mathrm{n}}_{q} \, \mathbf{B}^{\mathrm{n}}_{q} \, . \tag{8}$$

Before accumulation of the local-nonlocal numeric matrix products

$$\mathbf{C}^{\mathrm{n}}_{q} = \mathbf{C}^{\mathrm{n}}_{q} + \beta \sum_{p \neq q}^{N_{\mathrm{Proc}}} \mathbf{A}^{\mathrm{n}}_{q} \, \mathbf{U}^{\mathrm{n}}_{p} \tag{9}$$

can commence, the algorithm waits until one or more $\mathbf{U}^{\mathrm{n}}_{p}$ have been received. Computation of this/these contributions proceeds while the remaining arrays fill. The method cycles until the accumulation is complete.

A summary of this method, along with the MPI calls employed at each step is given in the following:

1 As necessary, perform the global accumulations $\mathbf{A}^{\mathrm{s}}_{p} \to \mathbf{A}^{\mathrm{s}}$ and/or $\mathbf{B}^{\mathrm{s}}_{p} \to \mathbf{B}^{\mathrm{s}}$ (MPI_ALL_GATHERV).

2 Calculate dimensions of $\mathbf{U}^{\mathrm{s}}_{p}$ and $\mathbf{U}^{\mathrm{n}}_{p}$, and allocate them.

3 Compute $\mathbf{U}^{\mathrm{s}}_{p}$ from $\mathbf{A}^{\mathrm{s}}_{q}$ and $\mathbf{B}^{\mathrm{s}}$.

4 Post non-blocking receives (MPI_IRECV) for all $\mathbf{U}^{\mathrm{n}}_{p}$.

5 Block (MPI_BARRIER) until all receives have posted.

6 Identify outgoing blocks of $\mathbf{B}^{\mathrm{n}}_{q}$, create subset MPI types (MPI_TYPE_INDEXED, MPI_TYPE_COMMIT), and post them with a blocking send (MPI_SEND).

7 Block (MPI_BARRIER) until all sends have posted.

8 While receives complete, compute Eqs. (6-8).

9 Wait for some of the $\mathbf{U}^{\mathrm{n}}_{p}$ to arrive (MPI_WAITSOME), and compute contributions to Eq. (9). Loop until all contributions have been accumulated.

10 Deallocate buffers and free the MPI types (MPI_TYPE_FREE).

One might expect a nonblocking send (MPI_ISEND) to more efficiently mask communication than the blocking send at **6**. However, this is not the case. Once all the receives have been posted, guaranteed by the barrier at **4**, the blocking send completes before returning, and in practice returns with minimal overhead. On the other hand, even with the barriers at **4** and **7**, a nonblocking send may allow the communication to delay until a wait is posted. While this behavior is dependent on the MPI implementation, it appears to be 1-2 orders of magnitude slower across platforms and implementations.

While not required, the barrier at **7** dramatically improves efficiency of the method. It is not obvious why this is so, but remarkably it can improve performance by up to 50%.

Another peculiarity of the implementation has to do with the allocation of Fortran 90 pointers in the DBCSR derived types and non-blocking communication. Because the Fortran 90 standard does not require allocated pointers to employ contiguous memory, a contiguous temporary buffer may be created at runtime by the compiler before a subroutine is entered; data is copied in, copied out, and then deleted. Not only is the copy-in copy-out inefficient, but the temporary buffer is filled by MPI_IRECV, rather than the memory specified by the programmer. Once the temporary buffer is deallocated, the incoming data can create a fault. Smart compilers do not suffer from copy-in copy-out, but to be safe and efficient, contiguous (`ALLOCATABLE`) memory is used as necessary to avoid this syndrome.

B. Locality and Load

In parallel algorithm development, an important goal is to reduce interprocessor communication while evenly portioning the workload. For the sparse matrix-vector product, these objectives may be achieved through graph-partitioning [55,56,66]. Notable in this respect is use of the Reverse Cuthill-McKee method in the parallel tight-binding molecular dynamics of Colombo and Sawyer [68]. For the general parallel sparse matrix-matrix multiply involving matrices with different graphs, these methods are not generally applicable.

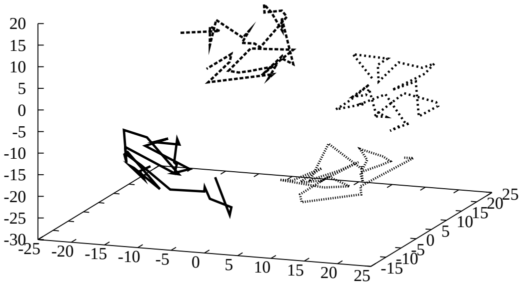

FIG. 2. The 3-dimensional Hilbert curve for domain decomposition of an $(H_2O)_{350}$ cluster. Of 32 domains, 4 of the most widely separated are shown ($p = 0, 11, 21, 31$).

Several geometric alternatives to graph-partitioning employ the space filling curve (SFC) [83,84] to achieve load balance and data locality in parallel algorithms [85–87]. The SFC is a heuristic solution to the traveling salesman problem [88,89], where reordered points that are close in space are also close on the curve.

The SFC is introduced here to achieve data locality and reduce communication overhead through bandwidth reduction. The SFC is used to re-order the atomic coordinates, producing approximate diagonal dominance and data locality for matrix elements (blocks) $\mathbf{P}_{AB}$ that fall off with $|\mathbf{R}_A - \mathbf{R}_B|$ (See Fig. 1 for example).

With the order of all matrices fixed, domain decomposition is achieved by sectioning the SFC. This is shown in Fig. 2, where 4 sections of the 3-dimensional Hilbert curve [83] are shown for a cluster of 350 water molecules distributed over 32 processors. See Section IV for a discussion of the water and estane systems discussed here.

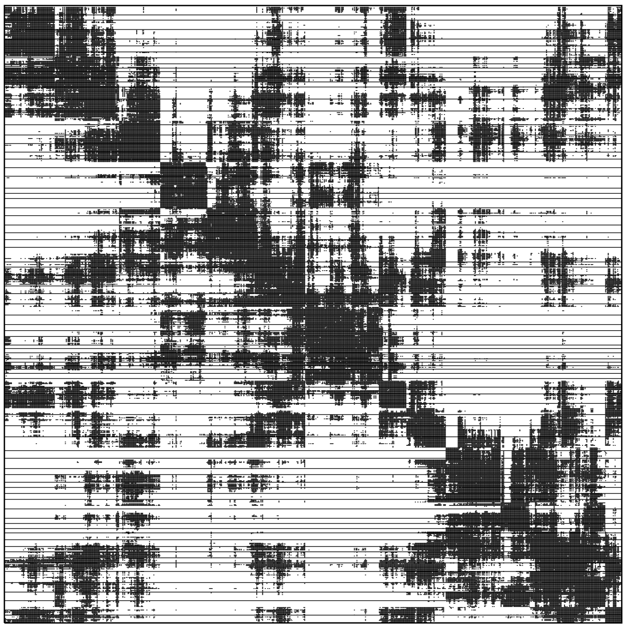

FIG. 3. Graph of the RHF/3-21G orthogonal Fock matrix of the Estane_3 globule obtained by reordering with the Hilbert curve. Also shown is the row-wise decomposition of the matrix for 100 processors.

In Fig. 3, this decomposition is shown for the graph of the orthogonal RHF/3-21G Fock matrix of Estane_3 (719 atoms) filtered using Eq. (4) with a drop tolerance $\texttt{Thresh\%Trix} = 10^{-4}$. Each point in this figure represents a dense block. This matrix is not as diagonally dominant or as sparse as one would hope. Nevertheless, this system size (719 atoms) is in the transition from a dense to sparse matrix. Larger systems will yield sparser matrices with more apparent diagonal dominance.

A hypothesis of the load balancing scheme developed here is that a uniform work load corresponds to a uniform distribution of non-zero matrix elements of some idealized reference matrix $\mathbf{A}$. With this reference, a metric $\eta\left[\mathbf{A}^s\right]$ is chosen that provides a measure of imbalance, and used to optimize the load balance. Possible representative matrices are the orthogonal Fock matrix and the density matrix from a previous cycle. One possible metric is

$$\eta = \sum_{p}^{N_{\rm Proc}} \left| \texttt{AveNon0} - \sum_{i=\texttt{Beg}(p)}^{\texttt{End}(p)} \texttt{A}(p)\%\texttt{Non0}(i) \right| , \qquad (10)$$

and another is

$$\eta = \max_{p} \sum_{i=\texttt{Beg}(p)}^{\texttt{End}(p)} \texttt{A}(p)\%\texttt{Non0}(i) \, , \qquad (11)$$

where `Beg` and `End` are sectioning arrays, $\texttt{A}(p)\%\texttt{Non0}(i)$ is the number of non-zero elements per atom, and `AveNon0`

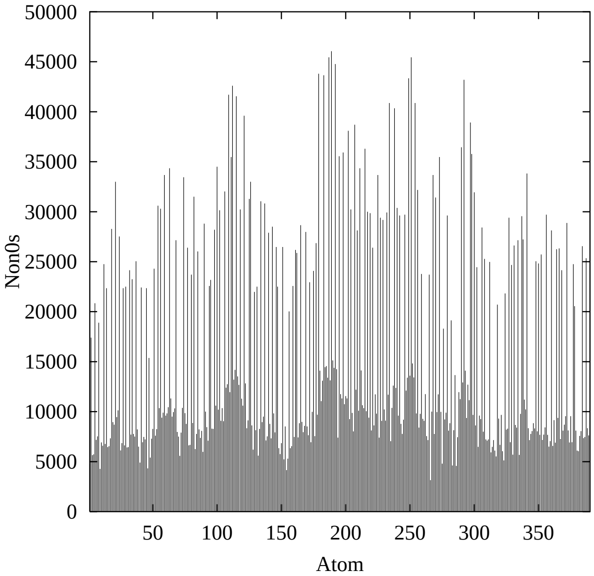

FIG. 4. Distribution of non-zero matrix elements per atom for the RHF/6-31G** orthogonal Fock matrix of $(H_2O)_{130}$ with $\texttt{Thresh\%Trix} = 10^{-4}$.

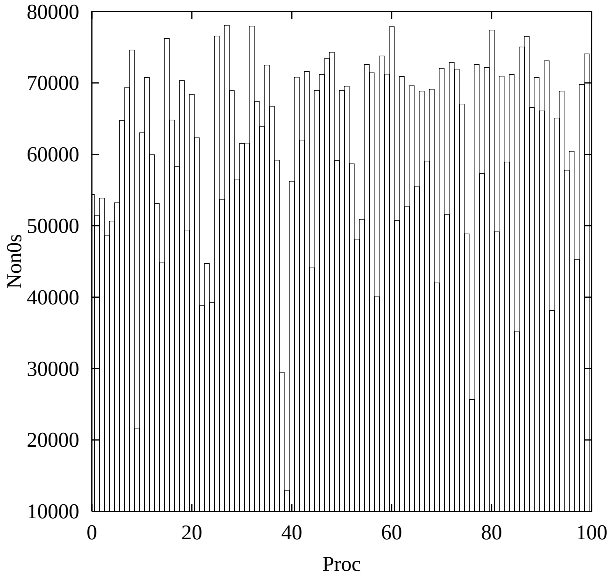

FIG. 5. Binning the non-zero matrix elements shown in Fig. 4 for 100 processors by simulated annealing solution of Eq. (10).

is the average number of non-zero elements per processor. Equation (11) is quite similar to one used by the Greedy Allocation-based Load-balancing Algorithm (GALA) of Nastea, Frieder, and El-Ghazawi [90] for balancing parallel sparse matrix-vector products. Here however, the freedom to re-order the matrix has been removed, and variable block sizes are involved rather than individual matrix elements.

Minimization of Eq. (10) or Eq. (11) is related to the variable-size bin-packing problem, for which there are a number of heuristics, see Ref. [90] and references therein. However, it has been found that these solutions are non-optimal [91], and so the sectioning arrays `Beg` and `End` are found by minimization of Eq (10) or Eq (11) with simulated annealing. In Fig. 4, the number of non-zero matrix elements per atom, $\texttt{A}(p)\%\texttt{Non0}(i)$, is shown for the RHF/6-31G** orthogonal Fock matrix corresponding to the $(H_2O)_{130}$ cluster. The corresponding simulated annealing solution to Eq. (10) is shown in Fig. 5.

C. Relationship to other methods

The methods developed here have significant overlap with other methods for the parallel sparse matrix multiply developed in the context of electronic structure theory [36,43,46], foremost being the use of dense atom-blocks and a row-wise decomposition. There is similarity of the present work with that of Canning *et. al* [46], where "xyz ordering" leads to a list with spatial locality. Load balance is achieved by sectioning the xyz ordered list. Differences exist however in that these methods have been developed for homogeneous systems (carbon [43,46] and silicon [36]), they are perfectly balanced when $N_{\text{Atoms}}/N_{\text{Proc}} = 1$, and they involve matrices with the same graph.

In Field's parallel sparse-blocked matrix-vector product [55] non-blocking sends and receives are used with resulting wait times that appear artificially large, which is consistent with the behavior observed here for the non-blocking send. The present work also has similarities to the Block-Cyclic Data Distribution method by Walker and Otto [92], in that non-blocking receives are combined with blocking sends.

Finally, it is worth mentioning the largely theoretical work on aspects of the parallel sparse matrix-matrix multiply in Refs [52–54].

IV. RESULTS

To demonstrate generality and portability, calculations were carried out in the SGI shared memory environment and on a Myrinet LINUX cluster. All results are for a complete computation of the density matrix starting from an orthogonal Fock matrix using program `SDMM` of the MondoSCF suite [93]. A DGEMM tuned by PHiPAC [62–64,94] for 5×5 blocks and Intel LINUX was used.

For the SGI calculations, the MIPSpro F90 compiler version 7.2.1 with `-O3` was used, while C code was compiled with the MIPSpro C compiler version 7.2.1 with `-O3`. All SGI calculations were run on part of ASCI Blue Mountain [95], using a dedicated 128 node symmetric multiprocessor (SMP) Origin 2000 with 250 MHz MIPS R1000 processors.

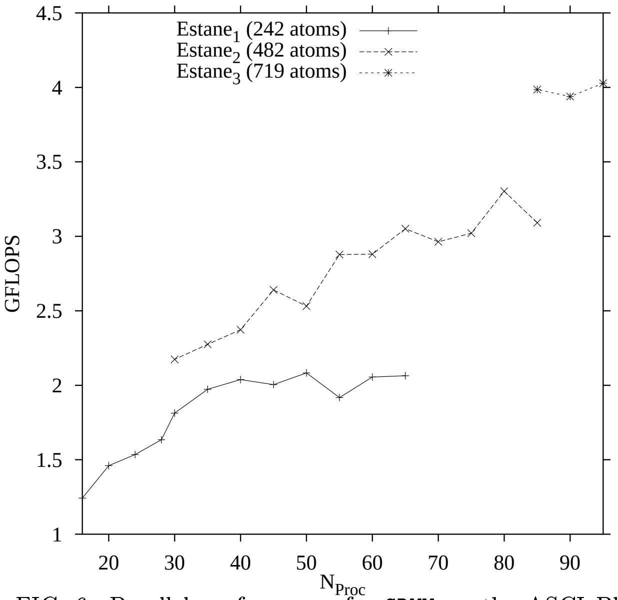

FIG. 6. Parallel performance for SDMM on the ASCI Blue Mountain Origin 2000 for RHF/3-21G estane globules.

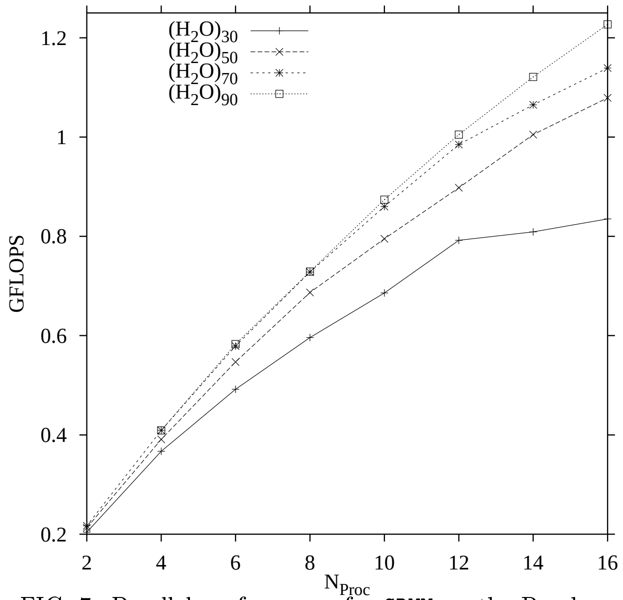

FIG. 7. Parallel performance for SDMM on the Roadrunner Linux cluster for RHF/6-31G** clusters of water molecules.

For the LINUX calculations, version 3.03 of the Portland Group pgf90 compiler [96] with `-O2 -tp p6 -pc 64 -Mdalign -Mnoonetrip` was used, while C code was compiled with version 2.7.2.3 of the GNU gcc compiler using `-ffast-math -fexpensive-optimizations -fomit-frame-pointer -O4`. All LINUX calculations were run on the Roadrunner Cluster [97], with 450 MHz Intel Pentium II nodes connected by Myrinet, and running MPICH version 1.1.1 on top of version 1.08 of the GM libraries.

The performance of SDMM has been benchmarked using the water cluster and estane globule geometries from the standard linear scaling benchmark series of molecular geometries [98]. The water clusters correspond to standard temperature and pressure. Estane 5703 polyester urethane is a segmented copolymer which is a constituent in the plastic-bonded high explosive PBX-9501 [99]. A fundamental repeat unit, $N_{\rm seg}=1$, is a length of chain comprised of two 4,4'-diphenylmethane diisocyanate and 1,4-butanediol segments bonded to five poly tetramethylene adipate segments. The estane globules have been generated by molecular dynamics equilibration followed by energy minimization, and include configurations for degrees of polymerization corresponding to $N_{\rm seg}=1$, 2, and 3 (Estane$_1$, Estane$_2$, and Estane$_3$), involving 242, 482 and 719 atoms respectively.

Atomic coordinates were reordered using the 3-dimensional Hilbert curve and 64 bit keys [83,84]. Also, the drop tolerance `Thresh%Trix` $= 10^{-4}$ was used throughout.

In Fig 6, shown are sustained Giga Floating Point Operations Per Second (GFLOPS) with processor number, achieved by SDMM in solving the SCF equations of RHF/3-21G Estane$_1$, Estane$_2$, and Estane$_3$ on ASCI Blue Mountain. Load balance was achieved by minimization of Eq. (11) using the orthogonal Fock matrix as reference.

Sustained GFLOPS per number of processors are shown in Fig. 7 for SDMM applied to RHF/6-31G** water clusters for the Roadrunner LINUX cluster, and in Fig. 8 for ASCI Blue Mountain. Load balance was achieved by minimization of Eq. (10) using the orthogonal Fock matrix as reference.

Experiments with both Eq. (10) and Eq. (11) were carried out, and found to yield different results only in as much as where one metric gave bad results (a dip in performance), the other tended to give good results (no dip).

Communication times for the global accumulation of local row and column indices $\mathbf{A}^{\rm s}_p \rightarrow \mathbf{A}^{\rm s}$ (item **1**, Section III A), and the wait for incoming numerical buffers $\mathbf{U}^{\rm n}_p$ (item **9**, Section III A), were found to be negligible on both platforms. Performance degradation was found to be related primarily to load imbalance in the loop over DGEMMs, and to correlate well with deviation of the average non-zeros per processor (`AveNon0`) from the maximum number of non-zeros on any processor. It also appears that for the larger system sizes and number of processors, performance was decreased due to paging (approximately 256 MB RAM per processor on Blue Mountain).

V. DISCUSSION

The premise that a good load balance corresponds to a uniform distribution of non-zero matrix elements of the orthogonal Fock matrix works well when $N_{\rm Atoms}/N_{\rm Proc} > 10$ for the inhomogeneous systems benchmarked here. However, with decreasing granularity it becomes impossible to achieve an optimal distribution due to differences in block size. Also, as the distribution of atom types per

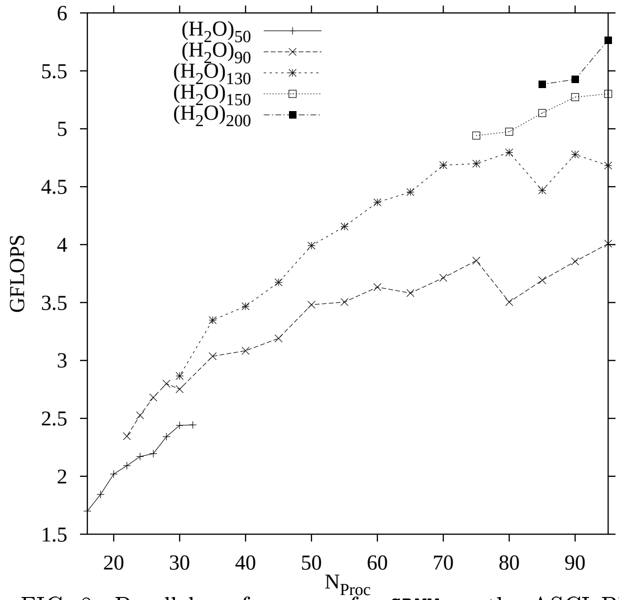

FIG. 8. Parallel performance for `SDMM` on the ASCI Blue Mountain Origin 2000 for RHF/6-31G** clusters of water molecules.

processor becomes inhomogeneous, the fact that different block sizes execute at different rates due to cache effects becomes more pronounced, and irregularities related to differences in the graphs of $\mathbf{A}^s$ and $\mathbf{B}^s$ become accentuated.

The results presented here correspond to a relatively fine level of parallelism, ∼3–6 atoms per processor. For a given number of processors, larger systems yield better results due to improved load balance, as shown in Figs. 6–7. Also, while the $\texttt{Thresh\%Trix} = 10^{-4}$ level of approximation is relatively crude, it does establish the performance of this approach with matrices that are irregular and sparse.

Achieving better results with fine grained parallelism is important. Several simple modifications of the load balancing scheme may yield an improved performance with only moderate effort. The average efficiencies of the DGEMM for various block sizes could be computed at start up, and used in a modified load balancing objective η. The DGEMM could be tuned for a somewhat broader range of matrix dimensions (and for the SGI platform) to obtain a more uniform execution rate.

By posting all receives at once, and all sends at once, the ability to overlap communication with computation has been decoupled from the load balancing problem. However, this is at the expense of increased memory demands as a buffer for all incoming $\mathbf{U}_p^n$ must be provided. It is certainly possible to reduce the memory used by looping over small batches of sends, receives, and local-nonlocal matrix multiplies. However, this requires a more involved approach to scheduling and interprocessor alignment. Improved methods for load balancing are expected to mitigate potential overheads associated with reduced memory variants.

While the general sparse-blocked matrix-matrix multiply developed here has been employed solely in the context of the SDMM, the ease of programming with the Fortran 90 `BCSR` and `DBCSR` derived types and the associated library of linear algebra routines make the implementation of new methods based on the matrix-multiply simple and straight forward. In particular, DIIS extrapolation [100,101] of the LNVD gradient, recently proposed by Bowler and Gillan [38], is a very promising method for obtaining accurate variational solutions to the SCF equations. It appears that this approach will allow a trade-off in the performance to accuracy ratio between basis set size and matrix sparsity.

VI. CONCLUSIONS

As linear scaling LCAO-SCF theory is applied to molecular systems of increasing size, the ability to rapidly evaluate energies and forces becomes more important due to statistical mechanical effects. Having already reduced the computational complexity to $\mathcal{O}(N_{\rm Atoms})$, a sufficient increase in speed can be achieved through parallelism.

A general message passing data parallel algorithm for the sparse-blocked matrix-matrix multiply has been presented. In association with this new algorithm, the space filling curve (SFC) heuristic has been introduced to computational chemistry as a method for obtaining approximate diagonal dominance and data locality in distributed sparse matrices with matrix elements that fall off with "separation". These methods form the computational kernel of program `SDMM` for linear scaling solution of the SCF equations. Sustained parallel performance for solving the SCF equations with `SDMM` have demonstrated a performance of over 5.7 GFLOPS with 95 processors of an SGI Origin 2000 and a resolution of 6 atoms per processor.

With the `BCSR` and `DBCSR` types well established, layering data parallel implementations for linear scaling Fock matrix construction on top of the MondoSCF linear algebra libraries can proceed with confidence. Parallel implementations of linear scaling Fock builds will greatly facilitate further development of the methods presented here.

It is worth mentioning that the SFC technology used here to promote data locality has been used successfully in massively parallel implementations of classical tree-codes [102–106], and should therefore be useful in parallelization of the Quantum Chemical Tree-Code [2] for computation of the Coulomb matrix. Looking ahead, future work on data parallel linear scaling Fock matrix construction involves the development of algorithms that efficiently span the following:

Atoms ⇔ Sparse Matrices ⇔ Charge Distributions ⇔ Grids ⇔ Multipole Trees

Modern methods of scientific computation typically involve highly specialized algorithms appropriate to only one level of this hierarchy (*e.g.* sparse matrices and graph theory). Parallel, linear scaling quantum chemistry involves a rich mix of advanced numerical methods, demanding a more universal approach that maintains data locality as this hierarchy is traversed. The space filling curve concept, introduced here to computational chemistry, may be useful in this respect.

VII. ACKNOWLEDGEMENTS

The University of New Mexico is thanked for access to their Albuquerque High Performance Computing Center. This work utilized the UNM-Alliance Roadrunner Supercluster. Sincere thanks to P. Kovatch for technical assistance on this platform.

Advice from M. Tůma and M. Benzi on linear algebra with CSR data structures is gratefully acknowledged. M. Benzi and J. Kress are thanked for a careful reading of this manuscript.

The ASCI project at LANL is gratefully acknowledged for support of this work. The Advanced Computing Laboratory of Los Alamos National Laboratory, Los Alamos, NM 87545 is acknowledged. Some of this work was performed on computing resources located at this facility.

---

[1] C. A. White, B. G. Johnson, P. M. W. Gill, and M. Head-Gordon, Chem. Phys. Lett. **253**, 268 (1996).
[2] M. Challacombe and E. Schwegler, J. Chem. Phys. **106**, 5526 (1997).
[3] J. M. Perezjorda and W. T. Yang, J. Chem. Phys. **107**, 1218 (1997).
[4] K. N. Kudin and G. E. Scuseria, Chem. Phys. Let. **289**, 611 (1998).
[5] J. C. Burant, R. E. Stratmann, and M. J. Frisch, J. Chem. Phys. **105**, 8969 (1996).
[6] E. Schwelger, M. Challacombe, and M. Head-Gordon, J. Chem. Phys. **106**, 9708 (1997).
[7] C. Ochsenfeld, C. A. White, and M. Head-Gordon, J. Chem. Phys. **109**, 1663 (1998).
[8] E. Schwegler, Ph.D. thesis, University of Minnessota, 1998, at `http://www.t12.lanl.gov/home/mchalla/`.
[9] E. Schwelger and M. Challacombe, J. Chem. Phys. **111**, 6223 (1999).
[10] R. E. Stratmann, G. E. Scuseria, and M. J. Frisch, Chem. Phys. Let. **257**, 213 (1996).
[11] G. Scuseria, J. Phys. Chem. **103**, 4782 (1999).
[12] M. Challacombe, J. Chem. Phys. **110**, 2332 (1999).
[13] S. Huzinaga, Comp. Phys. Rep. **2**, 281 (1985).
[14] E. R. Davidson and D. Feller, Chem. Rev. **86**, 681 (1986).
[15] I. Shavitt, Israel J. Chem. **33**, 357 (1993).
[16] R. D. Bardo and K. Ruddenberg, J. Chem. Phys. **60**, 918 (1974).
[17] D. F. Feller and K. Ruedenberg, Theo. Chim. Acta **52**, 231 (1979).
[18] M. W. Schmidt and K. Ruedenberg, J. Chem. Phys. **71**, 3951 (1979).
[19] S. Huzinaga and M. Klobukowski, Theochem **28**, 403 (1986).
[20] S. Huzinaga and B. Miguel, Chem. Phys. Let. **175**, 289 (1990).
[21] S. Wilson, Can. J. Chem. Rev. **70**, 290 (1992).
[22] D. Moncrieff and S. Wilson, J. Phys. B **26**, 1605 (1993).
[23] M. Klobukowski, Can. J. Chem. Rev. **72**, 1741 (1994).
[24] S. Obara and A. Saika, J. Chem. Phys. **89**, 1540 (1988).
[25] M. Honda, K. Sato, and S. Obara, J. Chem. Phys. **94**, 3790 (1991).
[26] P. M. W. Gill, Adv. Q. Chem. **25**, 141 (1994).
[27] E. Schwegler, M. Challacombe, and M. Head-Gordon, J. Chem. Phys. **109**, 8764 (1998).
[28] K. N. Kudin and G. E. Scuseria, J. Chem. Phys. **111**, 2351 (1999).
[29] X. P. Li, R. W. Nunes, and D. Vanderbilt, Phys. Rev. B **47**, 10891 (1993).
[30] M. S. Daw, Phys. Rev. B **47**, 10895 (1993).
[31] S. Y. Qiu, C. Z. Wang, K. M. Ho, and C. T. Chan, J. Phys. Cond. Matt. **6**, 9153 (1994).
[32] R. W. Nunes and D. Vanderbilt, Phys. Rev. B **50**, 17611 (1994).
[33] E. Hernández and M. J. Gillan, Phys. Rev. B **51**, 10157 (1995).
[34] J. M. Millam and G. E. Scuseria, J. Chem. Phys. **106**, 5569 (1997).
[35] A. D. Daniels, J. M. Millam, and G. E. Scuseria, J. Chem. Phys. **107**, 425 (1997).
[36] C. M. Goringe, E. Hernández, M. J. Gillan, and I. J. Bush, Comp. Phys. Comm. **94**, 89 (1996).
[37] D. R. Bowler, I. J. Bush, and M. J. Gillan, (1999), submitted.
[38] D. R. Bowler and M. J. Gillan, Comp. Phys. Comm. **120**, 95 (1999).
[39] A. H. R. Palser and D. E. Manolopoulos, Phys. Rev. B **58**, 12704 (1999).
[40] P. Ordejón, D. A. Drabold, M. P. Grumbach, and R. M. Martin, Phys. Rev. B **48**, 14646 (1993).
[41] F. Mauri, G. Galli, and R. Car, Phys. Rev. B **47**, 9973 (1993).
[42] F. Mauri and G. Galli, Phys. Rev. B **50**, 4316 (1994).
[43] S. Itoh, P. Ordejón, and R. M. Martin, Comp. Phys. Comm. **88**, 173 (1995).
[44] J. Kim, F. Mauri, and G. Galli, Phys. Rev. B **51**, 1456 (1995).
[45] P. Ordejón, E. Artacho, and J. M. Soler, Phys. Rev. B **53**, 10441 (1996).
[46] A. Canning *et al.*, Comp. Phys. Comm. **94**, 89 (1996).
[47] D. Sánchez-Portal, P. Ordejón, E. Artacho, and J. M. Soler, Int. J. Quant. Chem. **65**, 453 (1997).
[48] E. Artacho *et al.*, Phys. Stat. Sol. B **215**, 809 (1999).
[49] G. Galli, Cur. Op. Sol. State Mat. Sci. **1**, 864 (1996).
[50] D. R. Bowler *et al.*, Mod. Sim. Mat. Sci. Eng. **5**, 199 (1997).
[51] S. Goedecker, Rev. Mod. Phys. **71**, 1085 (1999).
[52] C. P. Kruskal, L. Rudolph, and M. Snir, Theo. Comp.

Sci. **64**, 135 (1989).
[53] G. Manzini, J. Parallel Dist. Comp. **21**, 169 (1994).
[54] M. Middendorf, H. Schmeck, and G. Turner, Australian Comp. J. **27**, 37 (1995).
[55] M. R. Field, SIAM J. Sci. Comp. **19**, 27 (1998).
[56] J. Shadid *et al.*, Parallel Comp. **23**, 1307 (1997).
[57] M. Ast, R. Fischer, J. Labarta, and H. Manz, Adv. Eng. Soft. **29**, 241 (1998).
[58] B. K. Kendrick and E. F. Hayes, J. Chem. Phys. (1999), to be submitted.
[59] S. Goedecker *et al.*, Technical Report No. LA-UR-97-1504, Los Alamos National Laborarory, Los Alamos, NM (unpublished).
[60] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, in *Management of Parallelism in Object-Oriented Numerical Software Libraries. Modern Software Tools in Scientific Computing*, edited by E. Arge, A. M. Bruaset, and H. P. Langtangen (Birkhauser Press, New York, 1997).
[61] J. Dongarra, J. Du Croz, I. Duff, and S. Hammarling., ACM Trans. Math. Soft. **16**, 1 (1990).
[62] J. Bilmes *et al.*, LAPACK working note 111, University of Tennessee (unpublished).
[63] J. Bilmes, K. Asanović, C. Chin, and J. Demmel, in *Proceedings of International Conference on Supercomputing* (ACM SIGARC, Vienna, Austria, 1997).
[64] J. Bilmes *et al.*, The PHiPAC WWW Home Page, `http://www.icsi.berkeley.edu/~bilmes/phipac`.
[65] L. F. Romero and E. L. Zapata, Parallel Comp. **21**, 583 (1995).
[66] Y. Saad, *Iterative methods for Sparse Linear Systems* (PWS Publishing, Boston, 1996).
[67] R. S. Tuminaro, J. S. Shadid, and S. A. Hutchinson, Conc. Pract. Exp. **10**, 229 (1998).
[68] L. Colombo and W. Sawyer, Mat. Sci. Eng. **B37**, 228 (1996).
[69] J. H. Wilkinson, *The aglebraic Eigenvalue Problem* (Clarendon Press, Oxford, 1965).
[70] G. W. Stewart, *Introduction to Matrix Computations* (Acadmeic Press, London, 1973).
[71] M. Benzi and G. H. Golub, BIT **39**, 417 (1999).
[72] T. Huckle, App. Num. Math. **30**, 291 (1999).
[73] M. Benzi and C. D. Meyer, SIAM J. Sci. Comput. **16**, 1159 (1995).
[74] M. Benzi, C. D. Meyer, and M. Tůma, SIAM J. Sci. Comput. **17**, 1135 (1996).
[75] M. Benzi, J. Marin, and M. Tůma, in *Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing*, edited by B. Hendrickson *et. al* (SIAM, Philadelphia, 1999), CD-ROM.
[76] M. Benzi, J. Marin, and M. Tůma, in *Fourth IMACS International Symposium on Iterative Methods in Scientific Computation II*, edited by D. R. Kincaid *et. al* (IMACS, Austin, TX, 1999).
[77] M. Benzi, personal communication.
[78] R. McWeeny, Rev. Mod. Phys. **126**, 1028 (1962).
[79] S. Pissanetzky, *Sparse Matrix Technology* (Academic Press, London, 1984).
[80] F. G. Gustavson, ACM Trans. Math. Soft. **4**, 250 (1978).
[81] P. Maslen *et al.*, J. Phys. Chem. A **102**, 2215 (1998).
[82] Message Passing Interface Forum. MPI: A message-passing interface standard (version 1.1), 1995, `http://www.mpi-forum.org`.
[83] T. Bially, IEEE Trans. Inf. Th. **15**, 658 (1969).
[84] C. Faloutsos and S. Roseman, in *Proceedings of the Eighth ACM Symposium on Principles of Database Systems* (ACM Press, Philadelphia,Pennsylvania, 1989), pp. 247–253.
[85] J. R. Pilkington and S. B. Baden, IEEE Trans. Parallel and Distributed Systems. **7**, 288 (1996).
[86] C. Ou and S. Ranka, J. Parallel Dist. Comp. **42**, 109 (1997).
[87] M. Griebel and G. Zumbusch, Parallel Comp. **25**, 827 (1999).
[88] L. K. Platzman and J. J. Bartholdi, J. ACM **36**, 719 (1969).
[89] M. G. Norman and P. Moscato, Chaos, Solitons and Fractals **6**, 389 (1995).
[90] S. G. Nastea, O. Frieder, and T. El-Ghazawi, J. Parallel Dist. Comp. **46**, 180 (1997).
[91] R. L. Rao and S. S. Iyengar, Comp. Math. App. **27**, 71 (1994).
[92] D. W. Walker and S. W. Otto, Concur. Pract. Exp. **8**, 707 (1996).
[93] M. Challacombe and E. Schwegler, MondoSCF v0.1, a suite of programs for linear scaling SCF theory, unpublished.
[94] R. Vuduc, PHiPAC DGEMM tuned for 5×5 matrices and Intel LINUX.
[95] The ASCI Blue Mountain WWW page, `http://ext.lanl.gov/projects/asci/bluemtn/`.
[96] The Portland Group compilers WWW page, `http://www.pgroup.com/`.
[97] The alliance Road Runner WWW page, `http://www.alliance.unm.edu/`.
[98] Standard coordinates for benchmarking linear scaling SCF theory. Water cluster and estane globule series available from `http://www.t12.lanl.gov/home/mchalla/`.
[99] T. R. Gibbs and A. Popolato, *LASL Explosive Property Data* (University of California Press, Berkeley, 1980), p. 109.
[100] P. Pulay, Chem. Phys. Let. **73**, 393 (1980).
[101] P. Pulay, J. Comp. Chem. **3**, 556 (1982).
[102] M. S. Warren and J. K. Salmon, in *Supercomputing '92* (IEEE Comp. Soc., Los Alamitos, 1992), pp. 570–576, (1992 Gordon Bell Prize winner).
[103] M. S. Warren and J. K. Salmon, in *Supercomputing '93* (IEEE Comp. Soc., Los Alamitos, 1993), pp. 12–21.
[104] J. K. Salmon and M. S. Warren, J. Comp. Phys. **111**, 136 (1994).
[105] M. S. Warren and J. K. Salmon, Computer Physics Communications **87**, (1995).
[106] M. S. Warren and J. K. Salmon, in *Seventh SIAM Conference on Parallel Processing for Scientific Computing* (SIAM, Philadelphia, 1995).