

Sparse Matrix Algebra for Quantum Modeling of Large Systems

Emanuel H. Rubensson^{1,2}, Elias Rudberg^{1,3}, and Paweł Sałek¹

¹ Department of Theoretical Chemistry,

Royal Institute of Technology, SE-10691 Stockholm, Sweden

² Department of Physics and Chemistry,

University of Southern Denmark, DK-5230 Odense M, Denmark

³ Department of Chemistry,

University of Warwick, Coventry CV4 7AL, UK

Abstract. Matrices appearing in Hartree–Fock or density functional theory coming from discretization with help of atom–centered local basis sets become sparse when the separation between atoms exceeds some system–dependent threshold value. Efficient implementation of sparse matrix algebra is therefore essential in large–scale quantum calculations. We describe a unique combination of algorithms and data representation that provides high performance and strict error control in blocked sparse matrix algebra. This has applications to matrix–matrix multiplication, the Trace–Correcting Purification algorithm and the entire self–consistent field calculation.

1 Introduction

The properties of matter as we see it are parametrized by the behavior of single atoms. On one hand, atoms that strongly bind to each other will often make a strong material. On the other, few atoms missing from the crystal structure will introduce strain on microscopic scale that can considerably affect macroscopic mechanical or electric properties of samples consisting of many orders of magnitude more atoms. A doping concentration of 1 atom per 10^5 can make the difference between an insulator and a semiconductor [1]. Unfortunately, direct investigation of processes at this level is often difficult and sometimes outright impossible since the act of measurement itself can affect its result [2, 3]. Computer modeling of such processes becomes very important for the understanding of such systems. Since creation and breaking of chemical bonds often involves redistribution of the electronic wave function, the system must be described at the quantum level. There are many processes that do not involve considerable electron redistributions and they can be simulated at a simpler, classical level – this is however out of scope of this article. Each electron – and interesting systems consist of thousands of them – is described by its wave function determining the probability that the electron can be found at some point in space. The total electronic wave function must fulfill additional conditions. Since the electrons are indistinguishable fermions, the total wave function must change

only sign when two electrons are swapped. This constraint leads to the simple Slater determinant representation ansatz and in turn to the Hartree–Fock (HF) model of the Schrödinger equation if only one determinant is used. The HF model effectively simulates electron movement in an approximate, averaged field of all other electrons and nuclei, and all motion correlation effects are neglected. There have been many attempts to improve this by – for example – including more determinants in the expansion, but they lead to considerably increased computational cost [4]. One of the main advantages of the HF theory is that the practical implementation can be made to scale linearly with the size of the modeled system. An alternative to wave function theory is the so-called Density Functional Theory (DFT). This theory avoids to some extent the complexity associated with wave function theory by using the electronic density $\rho(\mathbf{r})$ as the primary variable. This variable choice automatically makes the electrons indistinguishable – only the total density can be determined. The major challenge of DFT is to determine the energy associated with a given density. The initial obstacle related to the kinetic energy term was overcome by the theory of Kohn and Sham (KS). This theory exploits the success of HF and introduces a concept of orbitals so that the kinetic energy functional is evaluated in a manner analogous to HF theory. Currently, many approximations exist for the remaining exchange and correlation terms.

This paper is concerned with an effective way of representing electron density in HF and KS theories. The density $\rho(\mathbf{r})$ is usually represented by a matrix expansion

$$\rho(\mathbf{r}) = \sum_{p,q=1}^K D_{pq} b_p(\mathbf{r}) b_q(\mathbf{r}) \quad (1)$$

where D_{pq} are elements of the density matrix D and $\{b_p(\mathbf{r})\}$ is a set of K basis functions. Common choices for the basis set include plane wave functions, Slater functions $e^{-\alpha r}$, or Gaussian functions $e^{-\alpha r^2}$, the two latter ones multiplied by an angular part and centered at the atoms. The objective of such calculations is to minimize the total HF/KS energy E expressed in terms of the one-electron operator matrix H_1 , the two-electron matrix $F_{2\text{el}}$ and – in case of DFT – an exchange–correlation term E_{xc}

$$E = \text{Tr} [H_1 D] + \frac{1}{2} \text{Tr} [F_{2\text{el}} D] + E_{\text{xc}}. \quad (2)$$

The matrix H_1 that includes kinetic energy and nuclear attraction terms depends only on the chosen basis set and atom charges and positions. The matrix $F_{2\text{el}}$ and the scalar E_{xc} both depend on the trial electron density. From now on, the total HF/KS potential matrix $F = H_1 + F_{2\text{el}}$ will be referred to as the Fock matrix. In case of DFT, an additional exchange–correlation term F_{xc} is added to F . HF/KS calculations are done in cycles, each of them involving two time consuming steps: a) evaluation of the Fock matrix for a trial electron density and b) search for the corresponding density matrix. These cycles are performed until self-consistency is reached. The formation of the new density matrix in step b) traditionally uses the so-called aufbau principle: The Fock matrix F is

diagonalized to obtain its eigenpairs. The eigenvectors C^{occ} associated with the smallest eigenvalues are combined to obtain the density matrix D :

$$FC^{\text{occ}} = \varepsilon SC^{\text{occ}} \quad \rightarrow \quad D = C^{\text{occ}}(C^{\text{occ}})^T \quad (3)$$

where S is the basis set overlap matrix. This operation scales cubically with the problem size and becomes the bottleneck for large systems. A method that takes advantage of the existing sparsity in the F matrix is needed. Several algorithms have been proposed for this purpose [5–12]. All of them compute the solution iteratively by repeated matrix–matrix multiplications. The performance is therefore closely connected to the efficiency of the multiplications. The multiplications can scale linearly if multiplications by zeros which are present in sparse matrices are avoided. Multiplication may result in new small elements appearing but this fill-in can be prevented by filtering out small elements. A systematic filtering algorithm will control the error propagation and contain it under the user-requested threshold.

This paper presents a unique combination of algorithms that provide a robust framework for sparse matrix operations and enable linear scaling in density purification methods. First, a trace-correcting purification (TC2) algorithm is reviewed and some of its computational aspects are described. We then discuss sparsity properties of the involved matrices and review shortly an efficient method for the error control crucial for the TC2 algorithm. Next, we introduce the Hierarchic Matrix Library that was used to implement this algorithm. Finally, we present some benchmarks of this library.

The benchmark systems presented in this article are water droplets with up to 888 molecules, generated by molecular dynamics at 300 K. All benchmarks were performed on an Intel Xeon EM64T 3.4 GHz and 3–21G basis set was used unless stated otherwise.

2 Trace-Correcting Purification

Our work with sparse matrix algebra was motivated by the need for fast and reliable matrix operations in density matrix purification methods. Density matrix purification has been proposed in a multitude of variants [8–12]. The purification algorithms rely on the fact that the Fock matrix and the density matrix share a common set of eigenvectors but have different eigenvalues. One therefore applies a series of eigenvector conserving transformations to the Fock matrix so that the eigenvalues corresponding to occupied eigenvectors converge to 1 and the remaining eigenvalues converge to 0.

The trace-correcting purification algorithm (TC2), developed by Niklasson [9], is not only the simplest one but is also very competitive when it comes to performance measured in number of matrix–matrix multiplications required to converge [9, 11]. The TC2 algorithm assumes orthogonal basis set, i.e. the overlap matrix $S = I$. Therefore, the generalized eigenvalue problem in Eq. 3 has to be transformed to standard form. This can for example be achieved by a Cholesky decomposition of the overlap matrix $S = U^T U$ [13]. The purification algorithm

is then applied to $F_{ort} = U^{-T}FU^{-1}$ resulting in a density D_{ort} in orthogonal basis which can be transformed back to the original basis by $D = U^{-1}D_{ort}U^{-T}$. This results in an additional cost of four extra matrix–matrix multiplications per self-consistent field cycle plus the cost of one inverse Cholesky decomposition of the overlap matrix. TC2 also requires upper and lower bounds \mathbf{lmax} and \mathbf{lmin} of the eigenvalue spectrum which can be obtained using eg. the Gershgorin theorem [14]. The algorithm is as follows:

```

compute P = (lmax*I - F)/(lmax - lmin)
while not converged
  if(trace(P) > N) then
    P := P*P
  else
    P := 2*P - P*P
end while

```

The final result of the purification is contained in P. The efficiency and reliability of the purification algorithm depend on the representation of matrices and the algorithms used for the matrix manipulations. In particular, two important operations are repeated in each iteration of the procedure: 1) The symmetric matrix square and 2) Truncation of small matrix elements. Both these operations are described later on in detail. The simplicity of the TC2 algorithm is very appealing but one has to be aware of its drawbacks. One potential deficiency of the TC2 algorithm is that the accumulated error grows exponentially during a number of iterations – see eg. [10] and [15]. It is therefore crucial to diligently control the truncation error. Also, degenerate eigenvalues at the band gap – for which the diagonalization method would randomly pick some to be occupied – cannot be automatically handled and require separate treatment.

3 Sparsity

One feature that distinguishes matrices appearing in HF and KS computations is their rather limited sparsity. Sparsity patterns have previously been investigated for linear alkanes [16]. These systems are very sparse and linear scaling can therefore be achieved for relatively small systems. Three-dimensional dense systems like the water droplets we are using for benchmarks in this article are considerably more difficult to handle. The left panel of Figure 1 shows that the matrices have thousands of nonzeros per row and that this value is still increasing for 610 water molecules. While sparsity is larger in some cases, algorithms have to be able to handle semi-dense cases as well. In all computations presented in this article, the truncation of matrices was done so that, for given matrix A , truncated matrix \tilde{A} , and threshold τ , $\|A - \tilde{A}\|_F \leq \tau$ as described in section 4. A threshold of $\tau = 10^{-4}$ gives errors in total HF/KS energies of about 10^{-5} Hartree.

Previous work within this field use a blocked compressed sparse row representation [17] to store the non-vanishing submatrices. In this setting, basis functions are grouped into atom [6] or multi-atom [18] blocks where atoms and

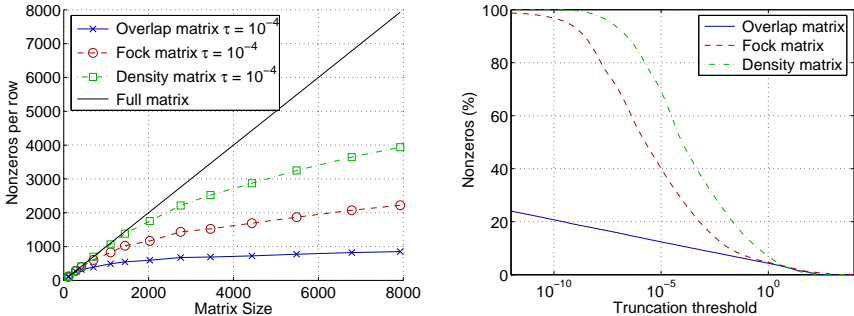


Fig. 1. Left panel: Sparsity in Hartree–Fock computations. The droplet size ranges from 8 to 610 water molecules. Full matrix given as a reference. Right panel: Percentage of nonzeros in the overlap matrix, the Fock matrix, and the density matrix for varying truncation threshold τ . Matrices were computed for a water droplet with 610 water molecules. The matrix size is 7930.

associated basis functions are reordered to group matrix elements associated with atoms close in space. Sparsity is then considered at the block level and dense block multiplications are performed using hardware-optimized linear algebra libraries [19, 20]. As a consequence, the block sizes are determined by the chosen basis set and the molecular geometry. A problem with this approach is that many linear algebra libraries perform considerably better for selected matrix sizes. Additionally, different block sizes appearing in atom and multi-atom based approaches lead to more substantial heap memory fragmentation which makes it more difficult to optimally utilize available resources. These problems can be avoided by choosing uniform block sizes as described later.

The most important matrices involved in HF and KS computations are the overlap matrix, the Fock matrix, and the density matrix. The right panel of Figure 1 shows an example of the sparsity of these three matrices for different values of the truncation threshold τ . The matrices used in the right panel of Figure 1 correspond to the largest matrix size in the left panel. We note that the change in sparsity with varying truncation threshold is different for the three matrices. This is because the sparsity in the overlap matrix is determined by the basis set only, while for the other matrices, sparsity is also dependent on the physical properties of the system.

4 Systematic Truncation of Small Elements

There exist several ways to maintain sparsity by dropping small matrix elements. One appealing way is to explicitly take advantage of the matrix element magnitude dependence on the distance between the corresponding basis function centers [18, 21]. A submatrix is dropped when the shortest distance between the two atom groups is greater than a predefined cutoff radius. It is, however, rarely

known which cutoff radius that will correspond to a certain accuracy and this approach will therefore cause severe difficulties with error control. Apart from risking larger errors than expected, one will usually also include submatrices with negligible contribution, reducing in this way the efficiency in subsequent matrix operations [15, 22]. Another way to remove unnecessary submatrices is to look at the maximum absolute element in the submatrix. The entire submatrix is dropped if this element is smaller than a preselected threshold. In this way one is able to strictly control the error. However, the error estimate obtained with this approach is far from optimally tight[15].

Based on these observations, we have come to the conclusion that it is better to formulate the truncation in terms of the norm of the entire matrix that one is interested in. For example, if one is interested in a certain accuracy in the total HF/KS energy, one should employ a truncation method that is based on the Frobenius norm of the entire matrix. The idea is simple; while keeping the error in the chosen norm below some requested threshold, as many submatrices as possible should be removed. Also, the truncation should be fast. Our method which we call a Systematic Small-Submatrix Selection Algorithm (SSSA) realizes this idea [15]. The SSSA outline is the following: The norms of all nonzero submatrices are computed and placed into a vector. Subsequently, the norms are sorted in descending order. Finally, small submatrices are removed from the end of this sorted vector as long as the sum of their norms is below the requested threshold. Figure 2 shows that the careful filtering obtained by using the SSSA algorithm together with the Frobenius norm keeps the total HF/KS energy error at a predictable level.

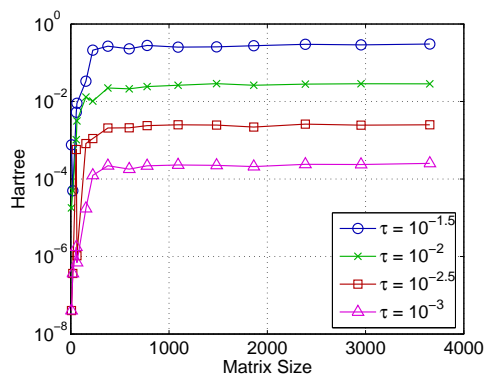


Fig. 2. Error in the final HF energy as a function of selected SSSA threshold and system size. For systems affected by truncation errors, SSSA algorithm keeps their impact at a strictly controlled level. The benchmark systems are water droplets. Basis set: STO-3G. The largest system, with 3654 basis functions consists of 522 water molecules.

5 Hierarchic Matrix Library

An optimal data structure for representation of sparse matrices appearing in HF/KS calculations has to fulfill several conditions. Since the matrices are semi-dense, the representation must not introduce much overhead. The representation must allow for quick evaluation of norms as needed for SSSA and be generally flexible to handle many matrix operations performed on matrices, like matrix multiplications by ordinary and on-the-fly transposed matrices. We propose a hierarchic matrix data structure that treats the matrix in several levels. At the lowest level in the hierarchy, the matrix elements are real numbers just like in the case of an ordinary full matrix. At higher levels, each matrix element is a hierarchic matrix. This makes it possible to consider sparsity at several levels in the hierarchy. If a submatrix is zero at a certain level in the hierarchy, it is unnecessary to reference lower levels. We have realized this idea using generic programming in C++ and called the resulting code library the Hierarchic Matrix Library (HML). A somewhat simplified building block illustrating the idea of HML is the following template:

```
template<typename Telement>
class Matrix {
    Telement* elements;
};
```

Using this template one may define an ordinary full matrix type and a hierarchic three-level matrix type as follows:

```
typedef Matrix<double> MatrixType;
typedef Matrix<Matrix<Matrix<double> > > ThreeLevelMatrixType;
```

At the lowest level we use a specialization that calls the Basic Linear Algebra Subprograms (BLAS). In this way high performance is obtained if the program is linked to a highly tuned BLAS implementation. The matrix-matrix multiplication at higher levels was implemented by the straightforward triple for-loop. Recursive algorithms have previously been used in dense matrix operations to optimally utilize deep memory hierarchies[23]. Also in HML, cache hit rate could possibly be improved by using some kind of cache oblivious algorithm combined with a more careful selection of submatrix block sizes at higher levels. In all benchmarks presented in this article the program was linked to Intel's Math Kernel Library (MKL).

Our reordering of basis functions is similar to the one proposed in Ref. [18] in that it is based on distances in space between atom centers. We have, however, chosen to keep the block size fixed rather than constraining it to match the number of basis functions on a set of atom centers. The reason for this is twofold: 1) At the matrix sizes we are interested in – typically in the range 0 to 200 – BLAS performance is often highly dependent on the matrix size. 2) Use of a uniform block size reduces the probability of memory fragmentation. In Figure 3, timings for two important matrix operations are plotted as functions of block size. The investigated operations are general matrix-matrix multiply (gemm) and symmetric matrix square (sysq). The gemm operation is $C = \alpha \text{op}(A) \text{op}(B) + \beta C$

and the sysq operation is $S = \alpha T^2 + \beta S$ where α and β are scalars, A , B , and C are general matrices, S and T are symmetric matrices, and $\text{op}(A) = A$ or $\text{op}(A) = A^T$. Note that in this figure the block sizes have been chosen as a multiple of 4 since MKL’s gemm operation is optimized for such matrix sizes. Scanning block sizes with step 1 would result in a highly jagged plot, with performance drops up to 40%. This plot may look different for other underlying BLAS libraries but suggests that the multiplication performance is fairly immune to changes of the block size as long as the block size is in the range 24 to 64.

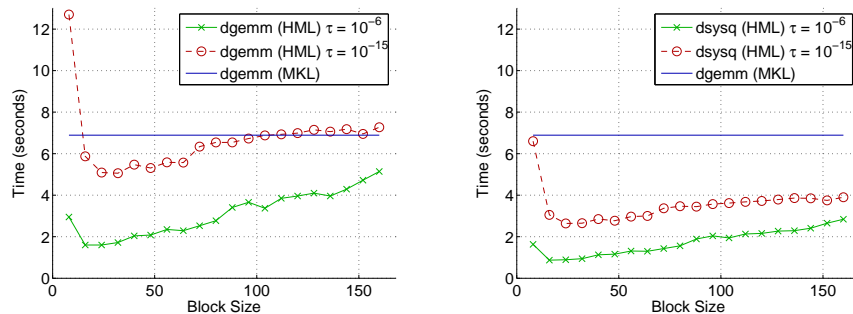


Fig. 3. Sparse matrix–matrix multiplication performance for different choices of block size and truncation threshold with fixed matrix size = 2756, with MKL’s full matrix dgemm implementation used as reference. The benchmarks were performed on an overlap matrix from a HF/KS calculation on a water droplet containing 212 water molecules. Left panel: Matrix–matrix multiply (dgemm). Right panel: Symmetric matrix square (dsysq).

6 Performance

An advantage with the hierarchic data structure compared to the blocked compressed sparse row representation is that symmetry in matrices can easily be utilized, often increasing the computational speed and reducing the memory usage by a factor of 2. This can also be seen in Figure 3 where the time for a sysq operation for all block sizes is less than 55% of the time for a gemm operation. In the context of hierarchic matrices this is achieved by recursive computation of symmetric matrix squares (sysq), symmetric rank–k matrix updates (syrk), symmetric matrix–matrix multiplies (symm), and matrix–matrix multiplies (gemm). Figure 4 displays the performance of the TC2 algorithm using HML compared to the traditional diagonalization method. It can be seen that purification can be used for all matrix sizes, not only for large ones, without significant loss of performance. This is achieved thanks to the limited overhead in the HML implementation.

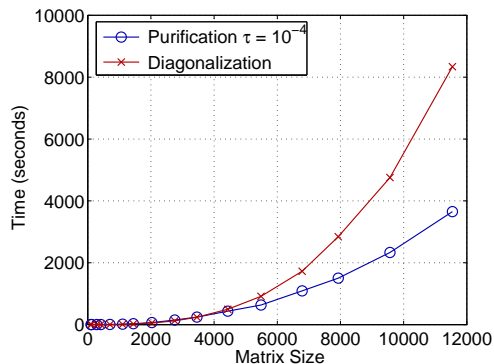


Fig. 4. Benchmark of the trace-correcting purification algorithm implemented with routines from the Hierarchic Matrix Library (HML) (truncation threshold $\tau = 10^{-4}$) compared to diagonalization with the Math Kernel Library’s dsygv.

7 Summary and Conclusions

We have presented a unique combination of theoretical methods, algorithms and data representation that allows to efficiently store and process matrices appearing in HF and KS theories. Our implementation uses a hierarchic data structure to efficiently store and access elements of sparse matrices, maintaining the truncation error under a requested threshold. This data structure makes it possible to implement many interesting algorithms in an efficient and transparent manner. As an example of such an algorithm, we have chosen the purification algorithm that thanks to our implementation can be used for robust and efficient calculations of electron density matrices in the HF and KS theories. Benchmarks that we present show that purification as implemented by us is competitive to the diagonalization method, becoming superior already for 4000 basis functions, even for dense, three-dimensional systems like water droplets, while maintaining high accuracy.

Acknowledgment This research has been supported by the Sixth Framework Programme Marie Curie Research Training Network under contract number MRTN-CT-2003-506842.

References

1. Kittel, C.: 8. In: Introduction to Solid State Physics. 7 edn. John Wiley&Sons, Inc. (1996)
2. Schrödinger, E.: Die gegenwertige situation in der quantenmechanik. Naturwissenschaften **23** (1935) 807

3. Schrödinger, E.: The present situation in quantum mechanics: a translation of Schrödinger's "cat paradox". In: Proc. Am. Phil. Soc. Volume 124. (1980) 323
4. See, e.g., C.J. Cramer, *Essentials of computational chemistry, 2nd ed.*, Wiley, Chichester, 2004.
5. S. Goedecker: Linear scaling electronic structure methods. Rev. Mod. Phys. **71**, 1085 (1999).
6. M. Challacombe: A simplified density matrix minimization for linear scaling self-consistent field theory. J. Chem. Phys. **110**, 2332 (1999).
7. Y. Shao, C. Saravanan, M. Head-Gordon, and C. A. White: Curvy steps for density matrix-based energy minimization: Application to large-scale self-consistent-field calculations. J. Chem. Phys. **118**, 6144 (2003).
8. A.H.R. Palser and D.E. Manolopoulos: Canonical purification of the density matrix in electronic structure theory. Phys. Rev. B **58**, 12704 (1998).
9. A.M.N. Niklasson: Expansion algorithm for the density matrix. Phys. Rev. B **66**, 155115 (2002).
10. A.M.N. Niklasson, C.J. Tymczak, and M. Challacombe: Trace resetting density matrix purification in O(N) self-consistent-field theory. J. Chem. Phys. **118**, 8611 (2003).
11. D.A. Mazziotti: Towards idempotent reduced density matrices via particle-hole duality: McWeeny's purification and beyond. Phys. Rev. E **68**, 066701 (2003).
12. A. Holas: Transforms for idempotency purification of density matrices in linear-scaling electronic-structure calculations. Chem. Phys. Lett. **340**, 552 (2001).
13. J. M. Millam and G. E. Scuseria: Linear scaling conjugate gradient density matrix search as an alternative to diagonalization for first principles electronic structure calculations. J. Chem. Phys. **106**, 5569 (1997).
14. J.W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.
15. E.H. Rubensson and P. Salek: Systematic sparse matrix error control for linear scaling electronic structure calculations. J. Comp. Chem. **26**, 1628-1637 (2005).
16. P.E. Maslen, C. Ochsenfeld, C.A. White, M.S. Lee, and M. Head-Gordon: Locality and Sparsity of Ab Initio One-Particle Density Matrices and Localized Orbitals. J. Phys. Chem. A **102**, 2215 (1998).
17. F. G. Gustavsson: Two Fast Algorithms for Sparse Matrices: Multiplication and Permuted Transposition. ACM Trans. Math. Softw. **4**, 250 (1978).
18. C. Saravanan, Y. Shao, R. Baer, P.N. Ross, and M. Head-Gordon: Sparse matrix multiplications for linear scaling electronic structure calculations in an atom-centered basis set using multiatom blocks. J. Comp. Chem. **24**, 618 (2003).
19. E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, D. Sorensen, *LAPACK Users' Guide*, release 2.0 SIAM, Philadelphia (1994).
20. R. C. Whaley and A. Petitet: Minimizing development and maintenance costs in supporting persistently optimized BLAS. Softw. Pract. Exper. **35**, 101 (2005).
21. X.-P. Li, R.W. Nunes, and D. Vanderbilt: Density-matrix electronic-structure method with linear system-size scaling. Phys. Rev. B **47**, 10891 (1993).
22. M. Challacombe: A general parallel sparse-blocked matrix multiply for linear scaling SCF theory. Comp. Phys. Comm. **128**, 93 (2000).
23. E. Elmroth, F. Gustavson, I. Jonsson, and B. Kågström: Recursive blocked algorithms and hybrid data structures for dense matrix library software. SIAM Rev. **46**, 3 (2004).