JOURNAL OF CHEMICAL PHYSICS VOLUME 112, NUMBER 23 15 JUNE 2000

Efficient pseudospectral methods for density functional calculations

R. B. Murphy, Y. Cao, M. D. Beachy, and M. N. Ringnalda
Schrodinger, Inc., 1500 S.W. First Avenue, Portland, Oregon 97201

R. A. Friesner[a)]
Columbia University, Department of Chemistry, New York 10027

(Received 10 September 1999; accepted 28 January 2000)

Novel improvements of the pseudospectral method for assembling the Coulomb operator are discussed. These improvements consist of a fast atom centered multipole method and a variation of the Head–Gordan J-engine analytic integral evaluation. The details of the methodology are discussed and performance evaluations presented for larger molecules within the context of DFT energy and gradient calculations. © *2000 American Institute of Physics.* [S0021-9606(00)01415-X]

I. INTRODUCTION

Density functional (DFT) methods have, over the past decade become increasingly used in solving a wide range of chemical problems. While the methods are still far from providing a truly quantitative description of all aspects of electronic structure (for example, van der Waals energies are not properly evaluated by any current functional), the development of gradient corrected and adiabatic connection methods by Becke[1,2] and the subsequent assembly of a family of useful functionals by the Pople group[3] has dramatically enhanced the accuracy of DFT energetics and structures. These developments, combined with the relatively low computational cost of DFT calculations, have rendered the approach of the method of choice for a significant number of applications, particularly for large systems, where traditional *ab initio* correlated approaches have long been intractable.

In this paper, we describe the application of pseudospectral numerical methods to the solution of the major variants of DFT. Our previous work[4–7] has been focused upon Hartree–Fock calculations and on wave function-based localized correlation methods such as local MP2 and generalized valence bond (GVB) approaches. There has been no barrier in principle to the extension of PS methods to DFT calculations and, indeed, we have had this capability in our Jaguar suite of *ab initio* programs for several years now. However, the initial implementations were sufficiently far from optimal that we viewed the publication of them as potentially misleading in terms of the underlying viability of the methodology. Over the past two years, we have refined the relevant numerical techniques, and can now report performance that we believe to be consistent with the inherent merits (and limitations) of pseudospectral approaches to this problem. The implementation includes a newly developed fast multipole methodology that differs considerably in its details from algorithms proposed by others in the context of conventional quantum chemical technology. This algorithm will be described below and its performance explicated for a range of realistic chemical problems, as opposed to the rather specialized systems (and low level basis sets) presented in previous papers[8,9] on this subject. We also present a novel variant of the J-engine algorithm,[10] proposed by Head–Gordon and co-workers, specialized to three center two-electron integrals, which are required in pseudospectral evaluation the Coulomb operator.

We shall discuss algorithms and performance for both gradient corrected methods (for which the nonlocal HF exchange operator is not required) and adiabatic connection methods (for which the HF exchange operator is required). We choose the B-LYP[11] method as an example of the former approach and the B3-LYP[1,2] method as an example of the latter. Results are presented for a wide range of basis sets and for gradient as well as single-point energy timings; this is essential as geometry optimization and/or large basis set single-point calculations clearly consumes the great majority of CPU time is calculations that aim for high quality results. We have also completed the development of second derivative methods (including second derivatives for effective core potentials, or ECPs), but this will be reported in a separate publication.

The paper is organized as follows. In Sec. II we provide a brief overview of the now-standardized elements of PS electronic structure methods, detailed descriptions of which can be found in other publications. In Sec. III we provide an in depth discussion of the evaluation of the Coulomb operator, including our new fast multipole methods and the three center J-engine algorithm. In Sec. IV we present timing results for a range of chemical systems, focusing on large molecules.

II. OVERVIEW OF PSEUDOSPECTRAL NUMERICAL METHODS FOR *AB INITIO* SELF-CONSISTENT FIELD ELECTRONIC STRUCTURE CALCULATIONS

From an architectural point of view, PS self-consisent field (SCF) calculations, whether of the DFT or Hartree–Fock variety, proceed in analogy to conventional quantum chemical programs that expand molecular orbitals (or density orbitals, in the case of DFT) in a set of atom-centered Gaussian basis functions. The key step in the process, from the

[a)]Electronic mail: rich@chem.columbia.edu standpoint of computational effort (and all that we shall be concerned with in this paper) is assembly of the components of the Fock matrix. The one-electron Hamiltonian H_0 is constructed and added into the Fock matrix via the usual analytical one electron integrals of the kinetic energy and electron–nuclear interaction terms. The demanding part of the calculation is the construction of the Coulomb and exchange operators. The Coulomb matrix element J_{ab} between two atomic basis functions a and b is given by

$$J_{ab}=\sum_{cd} (ab|cd)\rho_{cd}, \tag{1}$$

where ρ is the density matrix and $(ab|cd)$ is a standard two-electron repulsion integral over the Gaussian basis functions *abcd*. In conventional quantum chemical programs, the two-electron integrals are evaluated analytically and the sum in Eq. (1) carried out. In the pseudospectral formulation,[4] a numerical grid is introduced and the Coulomb operator $J(g)$ is first assembled on the numerical grid

$$J(g)=\sum_{u\nu} \rho_{\mu\nu}A_{\mu\nu}(g), \tag{2}$$

where $A_{\mu\nu}(g)$ is a three-center, one-electron integral over atomic basis functions μ and ν, evaluated at grid point g. The matrix form of the Coulomb operator is then obtained as

$$J_{ab}=\sum_{g} Q_a(g)J(g)R_b(g), \tag{3}$$

where $R_b(g)$ is the atomic basis function b expressed on the grid and $Q_a(g)$ is a least squares operator, designed to take the product $J*R$ and project it onto atomic basis function a. The use of least squares is necessary to filter the aliasing error arising from the fact that in typical electronic structure calculations, the atomic basis sets employed are far from being complete in the relevant basis set space. We have discussed in detail methods used to construct an accurate and efficient representation of Q in previous publications and shall not repeat that work here; the reader is referred to Refs. 4, 5, and 13 for extensive discussions of this topic.

The Hartree–Fock exchange operator K_{ab} can be analyzed in a similar fashion. The final result for the PS representation of K is

$$K_{ab}=\sum_{g} Q_a(g)\sigma_{ab}(g)R_b(g), \tag{4}$$

$$\sigma_{ab}(g)=\sum_{ac} A_{ac}(g)\rho_{cb}, \tag{5}$$

where the various symbols have the meanings defined above.

A formal scaling analysis[4] of the two-electron and PS formulations for assembling the Coulomb and exchange operators leads to a scaling of N^4 for conventional two-electron methods and N^3 for PS methods, where N is the number of atomic basis functions used in the calculations (the number of grid points, which enters into the PS calculations, is proportional to N). For large systems, integral cutoffs reduce the scaling of both methods to N^2, and other techniques (such as fast multipoles, discussed below) can yield even further reductions to (asymptotically) linear scaling.[11] In practice, however, much of the computational effort for molecules of chemical interest does not fall into the asymptotic regime, and the underlying scaling behavior has a significant quantitative effect on performance. With proper optimization, PS methods can be made to substantially outperform conventional two-electron methods assuming that the basis set is sufficiently large. For small basis sets, such as STO-3G, analytical methods are to be preferred, as here the ratio of grid points to basis functions is big enough to make the prefactor in PS methods unfavorable.

To achieve the high precision needed for *ab initio* quantum chemical calculations, modifications of the above PS formulation are required. The most important of these is that some classes of two-electron integrals (those making the largest numerical contribution to the energy) are performed analytically. These include most one- and two-center integrals and a subset of three-center integrals of the form $(aa'|bc)$; the number of these last integrals depends upon the SCF iteration in question (the computational scheme varies on each iteration due to the use of Fock matrix updating) and a cutoff threshold for the size of the integrals. The $(aa'|bc)$ terms are required only in the Coulomb operator. A detailed analysis of these analytical correction terms in terms of the technology for implementing them and the rationalization in terms of relative amplitudes is presented in Ref. 13.

For DFT calculations, evaluation of the exchange-correlation (XC) operator is required, in addition to the Coulomb term. Gradient-corrected DFT methods such as BLYP[11] do not use Hartree–Fock exchange (the K operator presented above) and necessitate only numerical integration of the XC functional on a grid. As was indicated above, our methods for evaluation of the gradient-corrected XC functional are similar to those of others.[1–3] For hybrid functionals such as B3LYP, a component of the exchange operator is incorporated into the XC functional. In this case, Eq. (4) above is used to evaluate K as in Hartree–Fock calculations.

Our focus in this paper is on the refinement of the basic PS methodology described above to increase the efficiency of DFT calculations of both the gradient-corrected and hybrid form. We accomplish this in two ways. In this paper we focus mainly on the reduction in computational effort for assembly of the Coulomb operator. For gradient-corrected DFT, this is a very substantial part of the calculation, particularly for large systems. Our concern in this paper is not the ultralarge systems that others have made a central concern in formulating ''linear scaling'' DFT methods[12] (these papers also have a strong emphasis on molecules that are linear chains, whereas our emphasis instead is on globular, three-dimensional systems) but rather systems at the upper end (~50–150 atoms) of what are run in academic and industrial laboratories that utilize quantum chemistry at a production level, i.e., to study a large number of different molecules. Our atom-center multipole methods are capable of providing significant reductions in CPU time for gradient-corrected calculations, even for systems at the lower end of this range. A second improvement in Coulomb operator evaluation is the use of a modified version of the J-engine algorithm of Head-Gordon and co-workers.[10] This reduces computational effort for both gradient-corrected and hybrid DFT calculations. Finally, our implementation of the XC operator, combining efficient sparse matrix multiply techniques with a multigrid approach that has been implemented though is not detailed in this work, yields very large reductions in computational effort for XC energy and gradient evaluation. The efficiency of both the combined Coulomb and XC algorithms are presented in CPU timings below.

III. ACCELERATION OF PSEUDOSPECTRAL EVALUATION OF THE COULOMB MATRIX: ATOM-CENTERED MULTIPOLES AND J-ENGINE FORMALISM

A. Atom center multipole methods

1. Overview

A number of methods have now been described in the literature for acceleration of the evaluation of the Coulomb operator in electronic structure calculations via the use of fast multipole methods.[8,9,12] In general, these methods follow the protocols developed for classical simulations of charged particles[14] in that space is divided into cells, and integral product centers falling within these cells are grouped together in multipole terms. This approach makes sense when there are a large number of particles spread out more or less uniformly in the volume of interest. In an electronic structure calculation, however, this is generally not the case. The reason that multipole methods are even feasible in an electronic structure calculation for molecules of the sizes typically studied (30–200 atoms) is that there are a large number of product centers associated with the different primitive basis functions pairs. However, these product centers are very irregularly distributed. In particular, they are disproportionately concentrated around the atoms of the molecule. The reason for this is easy to understand. If the exponents of the primitives are both large compared to their separation, then the overlap of the basis functions is nearly zero and the primitive can simply be discarded. Cases where the exponents are both small compared to the product center separation are relatively infrequent (this, in essence, requires two long range functions, whereas most contracted basis functions are short range in basis sets of decent quality). When one exponent is large and the other is small (the dominant case in practice), then the product center will be located very close to the atom on which the large exponent is situated. Our atom-centered multipole method exploits this fact by locating the multipole expansion centers directly on atoms. This dramatically reduces the number of terms in the multipole series needed for convergence. The efficiency of the method is documented below.

Additionally, the implementation of multipoles with PS methods is different in its details than what is required for conventional electronic structure calculations. The difference arises from the use of three-center one-electron integrals as the primitive integrals in the theory rather than the two-electron integrals that appear in conventional approaches. Our multipole method involves an approximation to the Coulomb field in physical space rather than approximating Fock matrix elements directly. Once the multipole part of the physical space Coulomb field is constructed, the additional cost to incorporate it into the final evaluation of the Coulomb matrix elements [Eq. (3) above] is negligible.

2. Coulomb field for pseudospectral integration

The Coulomb field at a given grid point $g, J(g)$, is represented pseudospectrally by a sum over three-center one-electron integrals $A_{\mu\nu}$ multiplied by the AO density matrix, $\rho_{\mu\nu}$

$$J(g)=\sum_{\mu\nu}\rho_{\mu\nu}A_{\mu\nu}(g). \tag{6}$$

The $A_{\mu\nu}$ integral is the Coulomb field at grid point g created by the contracted basis function pair $\mu\nu$

$$A_{\mu\nu}(g)=\int\frac{\mu(r)\nu(r)dr}{|r-r_g|}. \tag{7}$$

The $\mu\nu$ basis pair is composed of products of primitive Gaussian pairs $\alpha\beta$ with fixed contraction coefficients D_α, D_β, leading to the expansion of $A_{\mu\nu}$

$$\begin{aligned}A_{\mu\nu}(g)&=\sum_{\alpha,\beta}D_\alpha D_\beta\int\frac{\alpha(r)\beta(r)dr}{|r-r_g|}\\&=\sum_{\alpha,\beta}D_\alpha D_\beta A_{\alpha\beta}(g).\end{aligned} \tag{8}$$

The $\alpha\beta$ primitive Gaussian pair is typically written in terms of the ''product center'' $\mathbf{r}_p$ defined by the primitive Gaussian centers $\mathbf{A}$, $\mathbf{B}$ and exponents $\gamma\delta$

$$\mathbf{r}_p=\frac{\gamma\mathbf{A}+\delta\mathbf{B}}{\gamma+\delta}. \tag{9}$$

The $\alpha\beta$ product has the following form:

$$\begin{aligned}\alpha(r)\beta(r)=&\left(\prod_{j=xyz}(x_j-A_j)^{nj}(x_j-B_j)^{mj}\right)\\&\times\exp[-2\gamma\delta\sigma(\mathbf{A}-\mathbf{B})^2]\\&\times\exp[-(\gamma+\delta)(\mathbf{r}-\mathbf{r}_p)^2],\end{aligned} \tag{10}$$

$$\sigma=\frac{1}{2(\gamma+\delta)},$$

where n_j and m_j denote the Cartesian angular powers of the primitive (e.g., d_{xx}).

In the classical limit, namely from a distance far enough from $\mathbf{r}_p$, the $\alpha\beta$ product function simply becomes a point charge located at $\mathbf{r}_p$

$$\begin{aligned}\alpha(r_p)\beta(r_p)\rightarrow&\prod_{j=xyz}(x_{pj}-A_j)^{nj}(x_{pj}-B_j)^{mj}\\&\times[\exp(-2\gamma\delta\sigma)(\mathbf{A}-\mathbf{B})^2].\end{aligned} \tag{11}$$

In this classical limit the $A_{\alpha\beta}$ integral is simply

$$A_{\alpha\beta}(g)\rightarrow\alpha(r_p)\beta(r_p)/|\mathbf{r}_p-\mathbf{r}_g|. \tag{12}$$

Although this classical limit is relatively simple, the computational cost of evaluating the $1/R$ denominator at each grid point is a nontrivial cost to be avoided if possible. Our purpose in this research is to present a computationally fast multipole expansion of $A_{\alpha\beta}$ and assembly of the final Coulomb field [Eq. (6)] within this multipole expansion. The techniques to be presented perform optimally within the pseudospectral integration scheme and are not directly related to more common multipole expansions of four-center integrals.[10]

3. Atom-centered multipole expansion

Obtaining high accuracy with a low-order multipole expansion requires a judicious choice for the center of the multipole expansion. Within the Gaussian product framework discussed above we have chosen the multipole center to be the atom nearest the $\alpha\beta$ product center $\mathbf{r}_p$. Although more optimal choices can be constructed, the nearest-atom centered method has been found to perform adequately within a multipole expansion to the sixth order and with minimal complications of the formalism. Since many primitive Gaussians have relatively high exponents, a large fraction of the product centers are close to the atomic centers, thus making a low-order atom-centered expansion feasible. The expansion parameter of a given multipole expansion is therefore the product center to multipole center ($\mathbf{r}_{mp}$) distance, $|\mathbf{r}_p-\mathbf{r}_{mp}|$. The denominator in Eq. (12) can be rewritten to exemplify this expansion

$$\frac{1}{|\mathbf{r}_g-\mathbf{r}_p|}=\frac{1}{|(\mathbf{r}_g-\mathbf{r}_{\mathrm{mp}})+(\mathbf{r}_{\mathrm{mp}}-\mathbf{r}_p)|}=\frac{1}{|\mathbf{R}_g-\boldsymbol{\delta}|}. \tag{13}$$

The multipole expansion is feasible when, relative to the multipole center, the distance to the grid point ($\mathbf{R}_g$) is much larger than the distance to the product center (δ). For a particular $\alpha\beta$ primitive pair, the minimal distance $|\mathbf{R}_g|$ for use of the classical approximation [Eq. (7)] is R_{class}

$$R_{\mathrm{class}}=\delta+\sqrt{\ln(pf/\mathrm{tol})/(\gamma+\delta)}, \tag{14}$$

where pf denotes the constants multiplying the $\alpha\beta$ pair (e.g., $D_\alpha D_\beta \rho_{\mu\nu}$) tol is a tolerance for the accuracy (1.0e-07), and γ,δ are the Gaussian exponents. The minimal distance R_{mul} for the multipole expansion of order N_{mul} to be accurate is

$$R_{\mathrm{mul}}=\frac{\mathrm{to}}{pf}(2\delta)^{(1+\mathrm{ltot}*\mathrm{Nmul})}, \tag{15}$$

with ltot the total angular momentum of the $\alpha\beta$ pair. The net cutoff R_{cut} for both classical and multipole approximations is simply

$$R_{\mathrm{cut}}=\max(R_{\mathrm{mult}},R_{\mathrm{class}}). \tag{16}$$

The multipole expansion takes the form in Cartesian coordinates

$$\frac{1}{|\mathbf{R}_g-\boldsymbol{\delta}|}=\sum_{L=1,N_{\mathrm{mul}}}\sum_{j=1,f(N_{\mathrm{mul}})}\sum_{(m,n,l)}C^j_{mnl}\delta^m_x\delta^n_y \times\delta^l_z R^{m'}_{gx}R^{n'}_{gy}R^{l'}_{gz}\delta^L/R_g^{L+1}, \tag{17}$$

where the expansion is up to order N_{mul} with $f(N_{\mathrm{mul}})$ terms at a given order. For $N_{\mathrm{mul}}=6$ there are a total of 130 multipole terms. There is no advantage in this formalism to transform to more complicated non-Cartesian coordinate systems, as typically done in conventional multipole expansions.[10]

4. Multipole assembly

The objective is to formulate an efficient algorithm for assembling the part of the Coulomb field [Eq. (6)], which can be approximated by the multipole expansion. To clarify the algorithm we first outline the structure of the Coulomb field assembly code without multipoles. The outer loop of the $J(g)$ constructor is over grid blocks $\{g\}$ containing $\sim$128 grid points that are relatively well localized. The inner loops are over contracted basis pairs ($\mu\nu$) that are then further decomposed into primitive pairs ($\alpha\beta$). For each primitive pair $A_{\alpha\beta}(g)$ is computed and its contribution to $J(g)$ including the D,ρ prefactors of Eqs. (6), (8) are calculated.

For a given grid block, any contracted basis pair ($\mu\nu$) in Eq. (7) will have some subset of primitive $\alpha\beta$ pairs, which can be treated by the classical multipole expansion [Eq. (11)]. To make an efficient algorithm it is necessary to simplify the logic for determining this subset of classical pairs. This can be achieved by referencing the classical cutoff R_{cut} [Eq. (16)] of a primitive pair to the whole grid block rather to individual grid points. Thus, if the closest approach of the grid block to the multipole center is greater than R_{cut}, the primitive $\alpha\beta$ pair can be evaluated classically over the whole grid block. The localized nature of the grid blocks makes this block-based cutoff approach practical. Since the actual grid points are not needed to make this classical/multipole assignment, the calculation of most of the multipole expansion data can be done as a preprocessing step before looping over grid blocks. This aspect of the algorithm is critical to its efficiency. We presently discuss the bulk of the multipole algorithm that assembles components of the multipole terms that are independent of the grid points.

5. Preassembly

The preassembly phase contains a loop over contracted $\mu\nu$ basis pairs on all atoms, which is followed by loops on the corresponding primitive $\alpha\beta$ pairs. Once a batch of $\alpha\beta$ pairs of specific angular momentum is accumulated, multipole data for this batch is accumulated as follows.

Density matrix/Contraction coefficients/Cutoff data. The prefactors pf($\alpha\beta$) multiplying the primitive integral are first calculated with the inclusion of the contracted density matrix $\rho_{\mu\nu}$ and the exponential factor of Eq. (11)

$$\mathrm{pf}(\alpha\beta)=D_\alpha D_\beta\rho_{\mu\nu}\exp[-2\gamma\delta\sigma(\mathbf{A}-\mathbf{B})^2]. \tag{18}$$

The folding of the density matrix into the initial preprocessing is essential to the efficiency of the algorithm. For this reason the multipole method described herein is of no practical value for assembling the analogous exchange operators. If the prefactor is small enough the entire $\alpha\beta$ integral is ignored. The atom closest to the $\alpha\beta$ product center $\mathbf{r}_p$ is assigned as the multipole center of the $\alpha\beta$ pair. The classical/multipole cutoff in Eq. (16) can then be calculated and stored either in memory or on disk. This storage is necessary to ascertain on each grid block if the $\alpha\beta$ pair can be done classically using the grid-block-based cutoff scheme described in

TABLE I. Timings (SGI R 10k seconds) for a pseudospectral portion of Coulomb assembly. Timings for multipole terms are T-pre: time to do the precomputation of multipole coefficients; T-assem: total time spent in grid assembly of multipole terms; T-Aij: time to evaluate nonmultipole three-center integrals; T-Jij: total time to construct the pseudospectral Coulomb matrix over the grid; T-Jij: total time for Coulomb matrix *without* using multipoles.

Molecule/basis	N_{basis}	T-pre	T-assem.	T-Aij	T-Jij	T-Jij no multipoles
$C_{42}H_{86}$ 6-31G	550	2	22	79	180	368
Porphine cc-pVTZ(-f)	678	5	4	190	288	619
Taxol 6-31G**	1185	4	28	264	495	1092

subsequent sections. The I/O associated with reading the cutoff data is minimal compared to other parts of the calculation. The $\alpha\beta$ cutoff values are ''binned'' by the nearest integer to the real cutoff R_{cut} in Eq. (16). This binning of cutoffs facilitates the summing of multipole coefficients into integer cutoff classes discussed below.

6. Multipole coefficients

In this preprocessing phase of the algorithm the concern is to obtain the parts of the multipole coefficients that can be assembled and accumulated using product center data and no grid point input. The contributions of each $\alpha\beta$ primitive pair to all possible multipoles (N_{mul}) are accumulated and summed into an array $\text{Tot}_{\text{mp}}(\,)$ indexed by multipole number, expansion center, and the integer representation of the cutoff parameter discussed above. The first task is to calculate terms that are common to several individual multipole terms. One such class of terms are factors in Eq. (11) involving powers of coordinates weighted by the prefactors of Eq. (18)

$$S(\alpha\beta)=\text{pf}(\alpha\beta)\left(\prod_{j=xyz}(x_{pj}-A_j)^{nj}(x_{pj}-B_j)^{mj}\right). \quad (19)$$

A second term of this class are the powers of δ in Eq. (17) for each primitive pair. The binomial coefficients for each multipole are computed as well. Finally, the terms for the individual multipole moments involving powers of $\delta x,\delta y,\delta z$ of Eq. (17) are computed using standard recursion relations to minimize the number of multiplications involved. These terms for individual multipoles are multiplied by the outer factor S in the above equation to produce the quantity

$$S_{\text{mp}}(\alpha\beta,\text{mul})=S(\alpha\beta)C^{j}_{nml}\delta^{m}_{x}\delta^{n}_{y}\delta^{l}_{z}. \quad (20)$$

Binning of the multipole coefficients. At this point the desired quantity $\text{Tot}_{\text{mp}}(\,)$ describing the net multipole weight from primitive $\alpha\beta$ pairs that have a common multipole center and common classical/multipole cutoff R_{cut} can be simply formed by summing over S_{mp} in the above equation using the center and cutoff data of each primitive pair,

```
do $\alpha\beta$ pairs
    ia=mpcenter($\alpha\beta$)
    icut=int[$R_{\text{cut}}(\alpha\beta)$]
    do mul=1,$M_{\text{mult}}$                                   (21)
    $\text{Tot}_{\text{mp}}$(mul, ia, icut)=$\text{Tot}_{\text{mp}}$(mul,ia,icut)
                                 $+S_{\text{mp}}(\alpha\beta,\text{mul})$.
```

In the final assembly over grid blocks discussed further below, each grid block will be within some cutoff R_c of each multipole center. The multipole weight that contributes to this grid block is a sum over integer cutoff bins from 0 up to $\text{int}(R_c)$. Therefore it is expedient to sum $\text{Tot}_{\text{mp}}(\,)$ over bins

$$\text{Tot}_{\text{mp}}(\text{mul,ia,icut})=\sum_{ic=0,\text{icut}}\text{Tot}_{\text{mp}}(\text{mul,ia,ic}). \quad (22)$$

7. Coulomb assembly over grid blocks

With the $\text{Tot}_{\text{mp}}(\,)$ array completed in preprocessing, the remaining assembly of the multipole part of $J(g)$ over the grid is relatively simple. However, because of the large number of grid points, this final assembly is the most costly operation overall. Timings presented in Table I illustrate the relative times for the preassembly and the grid-based assembly.

Each grid block is passed to a routine that assembles the multipole portion of $J(g)$ on this block as follows. The outer loop of the assembly is over multipole centers, in this case over all atoms. The minimal distance from the multipole atom center *ia* to the grid block R_{cutg} is calculated and the corresponding integer based cutoff, icut. Next, all multipole terms are looped over in a recursive fashion followed by a loop over grid points. The multipole powers for multipole m involving the grid coordinates in Eq. (17) are calculated and $J(g)$ is incremented using $\text{Tot}_{\text{mp}}(\text{mul,ia,icut})$

$$J(g)=J(g)+\text{Tot}_{\text{mp}}(\text{mul,ia,icut})R^{m'}_{gx}R^{n'}_{gy}R^{l'}_{gz}/R^{L+1}_{g}. \quad (23)$$

If the size of Tot_{mp} times the maximum of the grid coordinate powers is less than a cutoff the assembly in the equation above is skipped.

B. J-engine algorithm for three-center two-electron integrals

In this section we describe in detail an efficient algorithm for the evaluation of three-center two-electron integrals of the form $(aa'|bc)$, where a,a',b,c denote contracted basis functions on distinct atomic centers *a,b,c*. Since these integrals are the most numerous and costly subset of analytic correction integrals, we have developed an algorithm explicitly designed to optimize the construction and postprocessing of this type of integral. The algorithm we have developed and implemented is broadly related to the ''J-engine'' two-electron integral algorithm of Head-Gordon *et al.*,[10] since the method folds the $(aa'|bc)$ integral with the AO density matrix while calculating the integrals. However, the algorithm described here differs in some important respects since it was independently designed to optimize the $(aa'|bc)$ class of integrals for general basis sets.

1. Gill–Head-Gordon algorithm and notation

We follow the notation of the four-center AO integral work of Gill, Head-Gordon, *et al.*[15] and refer to their method as needed rather than rederive the parts of this algorithm that are common to theirs. For this reason, and to compare to our faster algorithm, it is appropriate to briefy review the Gill–Head-Gordon method. The $(aa'|bc)$ integral over contracted basis functions is a sum over uncontracted primitive integrals

$$(aa'|bc)=D_{ka}D_{k'a}D_{kb}D_{kc}[a_{ka}a'_{ka'}|b_{kb}c_{kc}]. \tag{24}$$

The contracted $(aa'|bc)$ integral is further contracted by the AO density matrix ρ_{ij} to form parts of the $J_{aa'}$ and $J_{bc'}$ matrix elements, e.g.,

$$J_{bc}=\sum_{aa'}(aa'|bc)\rho_{aa'}. \tag{25}$$

The Gill–Head-Gordon method assembles the $(aa'|bc)$ integrals and then does the final assembly of the Coulomb operator in Eq. (25) while the ‘‘J’’-engine method and the method presented here fold the AO density into the two-electron integral construction, thus producing the J matrix elements more directly.

The integrals are constructed as products of bra $(aa'|\text{ket}|bc)$ pairs wherein the bra/kets contain primitive pairs each with the same pair degree of angular momentum denoted by *llb* and *llk* (e.g., *sp* pairs in the bra and *dd* pairs in the ket). The outermost loop of the code is over *bc* pairs with a fixed *llk*. Each primitive pair product in the bra/ket has an associated set of angular momentum vectors $\mathbf{p}$ and $\mathbf{q}$ defined by the Hermite representation of the pair product, (p and q will be used to denote $|\mathbf{p}|$ and $|\mathbf{q}|$). The basic entity from which the integrals are constructed is the *pq*-primitive braket,

$$\left\langle \begin{array}{ccc|ccc} \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{0} & \\ \mathbf{0} & \mathbf{0} & \mathbf{p} & \mathbf{0} & \mathbf{0} & \mathbf{q} \\ \mathbf{0} & \mathbf{0} & a+b & \mathbf{0} & \mathbf{0} & c+d \end{array} \right\rangle = (-1)^q[\mathbf{p}+\mathbf{q}]^{(0)}$$
$$=[\mathbf{r}]^{(0)}. \tag{26}$$

The $[\mathbf{r}]^{(0)}$ integrals are computed via recursion from the $[\mathbf{0}]^{(m)}$ integrals via

$$[\mathbf{r}]^{(m)}=R_i[\mathbf{r}-\mathbf{1}_i]^{(m+1)}-(r_i-1)[\mathbf{r}-\mathbf{2}_i]^{(m+1)}. \tag{27}$$

The $[\mathbf{0}]^{(m)}$ are the elementary integrals among the primitive Gaussian pairs. Primitive pairs that are far enough apart or that involve high exponents allow $[\mathbf{0}]^{(m)}$ to be rapidly evaluated by a classical expression.

Given the $[\mathbf{r}]^{(0)}$, the Gill–Head-Gordon algorithm contracts these quantities over the primitive contraction coefficients and then transforms the contracted $[\mathbf{r}]^{(0)}$ to real space integrals using recursion relations. The expensive parts of the algorithm are the construction of the primitive $[\mathbf{0}]^{(m)}$ and the contraction of the $[\mathbf{r}]^{(0)}$. We will mainly focus on the calculations involving the nonclassical braket primitive pairs as the assembly of the classical portion represents a very small fraction of the CPU time since the braket factorizes in the classical limit. The contraction step for a given contracted braket pair follows a two-step transformation, the first of which is

```
do $|\mathbf{p}|$
do $|\mathbf{q}|(|\mathbf{r}|=|\mathbf{p}+\mathbf{q}|)$
do $\mathbf{r}$
```

$$\Gamma(k,\mathbf{r})=\sum_{k'}\mathbf{r}_{kk'}C_{k'p}, \tag{28}$$

where k is the pair contraction degree k_bk_c of Eq. (24) for the ket and similarly k' for the ket. The $C_{k'p}$ is a scaling coefficient of the ket exponents and of $|\mathbf{p}|$. This transformation has a scaling of

$$\sum_{|\mathbf{p}|,|\mathbf{q}|}N_{kk'}M(|\mathbf{r}|), \tag{29}$$

where $M(\mathbf{r})$ is the number of $\mathbf{r}$ vectors for $|\mathbf{r}|$ and $N_{kk'}$ the number of nonclassical braket primitive pairs.

The second contraction is over k

```
do $|\mathbf{p}|$
do $|\mathbf{q}|(|\mathbf{r}|=|\mathbf{p}+\mathbf{q}|)$
do $\mathbf{r}$
```

$$\beta(\mathbf{r})=\sum_{k}\Gamma(k,\mathbf{r})C_{kq}. \tag{30}$$

The $\beta(\mathbf{r})$ are subsequently mapped to $\beta(\mathbf{p},\mathbf{q})$ pairs that are used in the inexpensive recursion relations to produce the final contracted AO integrals. The overall scaling of the transformation is

$$\sum_{|\mathbf{p}|,|\mathbf{q}|}N_{kk'}M(|\mathbf{r}|)+N_kM(|\mathbf{r}|). \tag{31}$$

2. Multipath $(aa'|bc)$ Coulomb assembly

The chief disadvantage of the formalism above is that it follows a single path. Our purpose in this research is to illustrate that for the $(aa'|bc)$ class of integrals, the assembly described above can be written in terms of multiple paths (methods), including density folding. Each of these methods of assembly has a different scaling as a function of $N_k,N_{k'},N_{kk'},N_p,N_q$, etc. The least expensive path can be chosen as a function of these variables for a given class of braket pairs. This freedom to choose multiple paths necessarily lowers the computation time.

The basic method aims to construct the density folded integrals denoted by $(aa'|BC)$ and $(bc|AA')$, where

$$(aa'|BC)=\sum_{bc}(aa'|bc)\rho_{bc};$$
$$(bc|AA)=\sum_{aa'}(aa'|bc)\rho_{aa'} \tag{32}$$

is a contraction over the bc AO density matrix elements with a similar definition for $(bc|AA)$. The method presented here focuses on an efficient construction of the contracted ket $|BC)$ and $|AA)$, with the final transform of the AO bra to real space from the pq Hermite space being done cheaply, as in the Gill–Head-Gordon algorithm (and even more inexpensively than in the Gill–Head-Gordon method since there is only one ket and not $b*c$ of them).

3. Intermediates

The various definitions of intermediate functions used in the multipath three-center integral method are presently discussed in order to define the various paths from these intermediates.

Using the Gill–Head-Gordon notation, the steps from $\beta(\mathbf{p},\mathbf{q})$ contracted pairs of Eq. (30) to AO integrals are represented by the bra ket transformations from momentum to real space. The first step transforming the $\mathbf{q}$ space to bc contracted AO pair space can be written for a given bc angular pair as

$$\left\langle \begin{array}{ccc|ccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{p} & 0 & b & c \\ a & a' & p & 0 & 0 & 0 \end{array} \right\rangle = \sum_q \gamma_{bc,\mathbf{q}} \left\langle \begin{array}{ccc|ccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{p} & 0 & 0 & \mathbf{q} \\ a & a' & p & b & c & q \end{array} \right\rangle = \sum_{\mathbf{q}} \gamma_{bc,\mathbf{q}} \mathbf{r}_{aa'p,bcq}, \tag{33}$$

where $\gamma_{bc,\mathbf{q}}$ denotes a recursion coefficient that is a function of the bc product pair distance and $\mathbf{q}$. To construct the density contracted $(aa'|BC)$ integral, we first leave the $(aa'|$ bra in momentum space ($\mathbf{p}$) and density contract the transformed bc side, i.e., we form

$$(aa'\mathbf{p}|BC) = \sum_{bc} \rho_{bc}(aa'\mathbf{p}|0bc). \tag{34}$$

To avoid the restriction of having to transform the $\mathbf{r}$ quantities above, it is expedient to write an expansion for $[\mathbf{r}]^{(0)}$ using the recursion relation of Eq. (27)

$$[\mathbf{r}]^{(0)}_{kk'} = \sum_m [0]^{(m)}_{kk'} P^m(\mathbf{r},k,X,Y,Z), \tag{35}$$

where $P^m(\mathbf{r},k,X,Y,Z)$ are polynomials that depend on the coordinates (XYZ) between the bc primitive pair product center and the one center (in $|aa')$) distance. The $P^m(\mathbf{r},k)$ are independent of the one-center contraction indices k'.

Using this expansion of $[\mathbf{r}]^{(0)}$,$(aa'\mathbf{p}|BC)$ of Eq. (34) can be expanded as

$$(aa'\mathbf{p}|BC) = \sum_{bc} \rho_{bc} \sum_{kk',\mathbf{q}} [0]^{(m)}_{kk'} P^m(k,\mathbf{r}) \times C_{k'p} C_{kq} \gamma_{bc,\mathbf{q}} \quad (\mathbf{r}=\mathbf{p}+\mathbf{q}). \tag{36}$$

The methods we present focus on an efficient assembly of the $(aa'\mathbf{p}|BC)$ [and similarly $(bc\mathbf{q}|AA)$]. As noted, the final transform from $(aa'\mathbf{p}|BC)$ to the full AO integral $(aa'|BC)$ is a minor fraction of the total CPU time. The key consideration in the assembly of Eq. (36) is to decompose it into assemblies of intermediates that are independent of the $(aa'\mathbf{p}|$ bra. The multiple intermediates that can be assembled are tuned to the angular degree of the contracted pair (i.e., the extent of pq vectors) as well as the contraction degrees (k,k'). In addition, the extent to which quantities are assembled independent of the aa' side depends upon the type of integral. For example, the AO density and recursion coefficients are readily partially contracted over the possible angular terms of a pp contracted pair, e.g., p_xp_y

$$U_{\mathbf{q}} = \sum_{bc=xy,xz,yz} \rho_{bc} \gamma_{bc,\mathbf{q}}. \tag{37}$$

In this manner the multiple pair angular indices (e.g., p_xp_y of a contracted $pp'bc$ pair) are immediately removed from the scaling of consequent steps. The $U_{\mathbf{q}}$ of each contracted pair can then be multiplied by $C_{k\mathbf{q}}$ to form $U_{q\mathbf{k}}$ for each each ket contracted pair. A similar folded recursion coefficient $U_{\mathbf{p}}$ can be defined for the $(bc\mathbf{q}|AA)$ assembly.

At this point we have enough of the essential definitions to represent the various possible paths for assembly of $(aa'\mathbf{p}|BC)$ and $(bc\mathbf{q}|AA)$ along with their scalings.

4. Definitions of paths and scalings

As outlined above, the general structure of the code is an outer loop over bc contracted pairs with a common pair angular momentum llk and pair contraction degree k. In addition, we have allowed for an sp angular block to be processed as a single angular type to avoid repetitive reconstruction of $[\mathbf{0}]^m_{kk'}$ terms. At this point, before the one-center index a appears, the density folded $U_{\mathbf{qk}}$ recursion coefficients of Eq. (37) are formed. The set of contracted bc pairs is then combined with a set of atoms defining the one-center parts on center a. The creation of (bc,a) triplets is made by choosing the a centers in groups such that for any bc pair a subset of the k contracted pairs can be assigned to k_{cl} classical pairs relative to any aa' pair. That is, in each ‘‘batch’’ of (bc,a) triplets k_{cl} of the kbc primitive pairs are classical with respect to any of the centers a in the batch.

At this point all unscaled primitive $[\mathbf{0}]^m_{kk'}$ are constructed and stored in memory. These quantities do not involve the contraction indices. The construction of the $[\mathbf{0}]^{(m)}$ integrals here is essential to avoid expensive redundant regeneration in later stages. The polynomials $P^m(\mathbf{r},k)$ of Eq. (35) can also be made at this location. For the first assembly method discussed below the intermediate Q in (a) of Eq. (38) can be formed at this point as well. The final loop is over individual aa' contracted pairs of a given pair angular momentum llb and contraction degree k'. Once a pair is chosen, the set of $k'_{\mathrm{cl}}aa'$ primitive pairs that can be done purely classically with the bck_{cl} pairs is computed.

Given llb,llk,k,k_{cl},k,k'_c it is now possible to choose the optimal path for construction of the $(a'|BC)$,$(bc|AA)$ integrals for this batch. The various paths of assembling contributions to the final $(aa'|bc)$ integral with this set of primitive braket pairs can now be defined. The paths discussed initially are concerned with the more computationally demanding nonclassical kk' pairs. Some of the intermediate quantities defined in Eqs. (4)–(13) will be referenced in the subsequent derivations. Unless otherwise noted, the scalings below refer to scalings per bc/aa' primitive pair assembly.

a. Path 1. For the assembly of $(aa'\mathbf{p}|BC)$, the path follows the calculations of

$$\text{(a)}\quad Q(k,m,\mathbf{p})=\left\langle \mathbf{p}\left|\sum_q p^m(k,\mathbf{r})U_{qk}\right.\right\rangle;\quad \mathbf{r}=\mathbf{p}+\mathbf{q}. \tag{38}$$

Note that this quantity is precomputed before the aa' loops have started:

$$\text{(b)}\quad Z(k,m,p)=\sum_{k'}[0]^{(m)}_{kk'}C_{k'p};$$

$$\text{(c)}\quad (aa'\mathbf{p}|BC)=\sum_{km}Z(k,m,p)Q(k,m,\mathbf{p}).$$

An intermediate V for the $(bc\mathbf{q}|AA')$ term is also incremented;

$$\text{(d)}\quad V(k,m,\mathbf{p})=V(k,m,\mathbf{p})+U_pZ(k,m,p),$$

and the increment of $(bc\mathbf{q}|AA')$ from V is made;

$$\text{(e)}\quad (bc\mathbf{q}|AA')=\left\langle \mathbf{q}\left|\sum_{\mathbf{p},k,m}P^m(k,\mathbf{r})V(k,m,\mathbf{p})\right.\right\rangle.$$

The scalings for the steps of path 1 are

$$\text{(a)}\ N_mN_\mathbf{p}N_kN_\mathbf{q}+\text{(b)}\ N_{kk'}N_pN_m+\text{(c)}\ N_kN_mN_\mathbf{p}$$
$$+\text{(d)}\ N_{kk'}N_pN_m+\text{(e)}\ N_\mathbf{p}N_\mathbf{q}N_kN_m,$$

where it must be remembered that the scaling for 14(a) is independent of the aa' loops. Path 1 is advantageous for high *llk* low *k, bc* terms (e.g., *dd* polarization) coupled with high k', e.g., $(ss'|DD')$ integrals.

b. Path 2.

$$\text{(a)}\quad Q(k,k',\mathbf{r})=\sum_{k'}[0]^{(m)}_{kk'}P^m(k,\mathbf{r}),$$

$$\text{(b)}\quad Z(k,\mathbf{r},p)=\sum_{k'}Q(k,k',\mathbf{r})C_{k'p},$$

$$\text{(c)}\quad \beta(\mathbf{r},p,q)=\sum_k Z(k,\mathbf{r},p)C_{kq},$$

$$\text{(d)}\quad (aa'\mathbf{p}|BC)=\left\langle \mathbf{p}\left|\sum_q \beta(\mathbf{r},p,q)U_\mathbf{q}\right.\right\rangle,$$

$$\text{(e)}\quad (bc\mathbf{q}|AA')=\left\langle \mathbf{q}\left|\sum_p \beta(\mathbf{r},p,q)U_\mathbf{p}\right.\right\rangle, \tag{39}$$

with scalings

$$\text{(a)}\ N_{kk'},N_mN_\mathbf{r}+\text{(b)}\ N_{kk'}N_pN_\mathbf{r}+\text{(c)}\ N_kN_qN_pN_\mathbf{r}$$
$$+\text{(d)}\ N_\mathbf{p}N_\mathbf{q}+\text{(e)}\ N_\mathbf{p}N_\mathbf{q}.$$

c. Path 3. Path 3 resembles path 2 with preference given to early k' sums rather than k

$$\text{(a)}\quad Q(k,k',\mathbf{r})=\sum_{k'}[0]^{(m)}_{kk'}P^m(k,\mathbf{r}),$$

$$\text{(b)}\quad Z(k',\mathbf{r},p)=\sum_k Q(k,k',\mathbf{r})C_{kq}$$

$$\text{(c)}\quad \beta(\mathbf{r},p,q)=\sum_{k'}Z(k',\mathbf{r},p)C_{k'p} \tag{40}$$

$$\text{(d)}\quad (aa'\mathbf{p}|BC)=\left\langle \mathbf{p}\left|\sum_q \beta(\mathbf{r},p,q)U_\mathbf{q}\right.\right\rangle$$

$$\text{(e)}\quad (bc\mathbf{q}|AA')=\left\langle \mathbf{q}\left|\sum_p \beta(\mathbf{r},p,q)U_\mathbf{p}\right.\right\rangle,$$

and scales as

$$\text{(a)}\ N_{kk'}N_mN_\mathbf{r}+\text{(b)}\ N_{kk'}N_qN_\mathbf{r}+\text{(c)}\ N_{k'}N_qN_pN_\mathbf{r}$$
$$+\text{(d)}\ N_\mathbf{p}N_\mathbf{q}+\text{(e)}\ N_\mathbf{p}N_\mathbf{q}.$$

d. Path 4. This path has the most in common with the Gill–Head-Gordon (17) path discussed above

$$\text{(a)}\quad Z(k,m,p)=\sum_{k'}[0]^{(m)}_{kk'}C_{k'p},$$

$$\text{(b)}\quad Q(k,\mathbf{r},p)=\sum_m P^m(k,\mathbf{r})Z(k,m,p),$$

$$\text{(c)}\quad \beta(\mathbf{r},p,q)=\sum_k Q(k,\mathbf{r},p)C_{kq}, \tag{41}$$

$$\text{(d)}\quad (aa'\mathbf{p}|BC)=\left\langle \mathbf{p}\left|\sum_\mathbf{q} \beta(\mathbf{r},p,q)U_\mathbf{q}\right.\right\rangle,$$

$$\text{(e)}\quad (bc\mathbf{q}|AA')=\left\langle \mathbf{q}\left|\sum_\mathbf{p} \beta(\mathbf{r},p,q)U_\mathbf{p}\right.\right\rangle,$$

and scales as

$$\text{(a)}\ N_mN_{kk'}N_p+\text{(b)}\ N_mN_kN_pN_\mathbf{r}+\text{(c)}\ N_kN_qN_pN_\mathbf{r}$$
$$+\text{(d)}\ N_\mathbf{p}N_\mathbf{q}+\text{(e)}\ N_\mathbf{p}N_\mathbf{q}.$$

5. Classical assembly

The integral assembly in the classical limit derives its speed from the independence of the bra and ket or multiplicative nature of the integral expression. In the following k,k' denote classical braket pairs.

These two terms are preassembled before the aa' loop is entered

$$\gamma(k,\mathbf{r})=\sum_m P^m(k,\mathbf{r})[0]^{(m)}_k, \tag{42}$$

$$Q(\mathbf{p})=\left\langle \mathbf{p}\left|\sum_{k,\mathbf{q}}\gamma(k,\mathbf{r})U_{\mathbf{q}k}\right.\right\rangle, \tag{43}$$

and inside the aa' loop

$$(aa'\mathbf{p}|BC)=\sum_{k'}Q(\mathbf{p})C_{k'p}, \tag{44}$$

TABLE II. Scalings of Coulomb matrix construction. Times as defined in Table I and with T-Jana, the time for analytic correction terms. T-Jij-tot is the total Coulomb matrix assembly time. The scaling derived from these timings are seen to be in the linear to quadratic regime ($\sim N^{1.5}$) for a two- to three-dimensional molecule.

Molecule/basis	N_{basis}	T-pre	T-assem.	T-Aij	T-Jij	T-Jana	T-Jij-tot
Taxol 6-31G	660	2	23	189	373	173	546
Taxol 6-31G**	1032	2	26	264	500	559	1059
Taxol 6-311G**	1422	7	34	433	829	980	1809
						Net scaling$\sim N^{1.5}$	

with increments of the function

$$\beta(\mathbf{p})=\beta(\mathbf{p})+\sum_{k'} C_{k'\mathbf{p}}U_{\mathbf{p}}, \tag{45}$$

following the aa' loop

$$\delta(\mathbf{r},q)=\sum_{k} \gamma(k,\mathbf{r})C_{kq}, \tag{46}$$

$$\langle bc\mathbf{q}|AA'\rangle=\left\langle \mathbf{q}\left|\sum_{p} \beta(\mathbf{p})\,\delta(\mathbf{r},q)\right.\right\rangle. \tag{47}$$

6. Gradient evaluation

The assembly of analogous gradient terms such as

$$\left(a\frac{\partial}{\partial x}a'\middle|BC\right) \quad \text{and} \quad \left(b\frac{\partial}{\partial x}c'\middle|AA'\right) \tag{48}$$

follows paths analogous to those discussed for the energy by simply accounting for the raising and lowering of angular momenta that occurs upon differentiation. As seen in Table III, the scaling advantages of this multipath method for gradients are more substantial than for the energy because of the strength of the multipath method in efficiently assembling the higher angular momentum terms.

IV. IMPLEMENTATION AND RESULTS

We have implemented the above methods in the Jaguar suite of *ab initio* electronic structure programs. We use standard basis sets and (when relevant) effective core potentials in the data reported below. The grids used for the DFT calculations use a geometric progression for the radial spacing and Lebedev angular distributions on each radial shell. The grid weights are produced using a method similar to that of Becke.[11] In converging SCF iterations, Jaguar utilizes Fock matrix updating, which allows some iterations to be evaluated on coarse grids, thus saving considerable CPU time. This and other details of SCF convergence have been discussed in detail in other publications (including construction of the initial guess for the wave function), and we shall not repeat those discussions here.

Our principal purpose in the present paper is to present performance data for the new algorithms discussed above, as well as overall timing data for running DFT calculations of various types in Jaguar. First, we examine how much time is saved by employing the atom-centered multiple and J-engine algorithms for a set of test cases. Then, we present overall CPU times for DFT calculations for a series of molecules that vary in size, composition, and overall topology (i.e., quasi-one-dimensional versus quasi-three-dimensional). Both single point and gradient timings are included for the 6-31G* basis set, which is the level at which one typically carries out geometry optimization. For the cc-pVTZ (-f) basis set of Dunning,[16] we present single-point timings; again, that is what this basis set is typically used for. For gradient timings, we report the average CPU time for an entire gradient cycle in a geometry optimization. The issue of how many geometry steps are required to achieve convergence is an important one, dependent upon both the quality of the optimizer and how much noise there is in the gradient (it is possible to reduce gradient times by ‘‘cheating’’ on the numerical precision in any type of calculation, however, this has the effect of increasing the number of geometry steps required for convergence). Our observation of Jaguar geometry optimization with the current implementation of the program is that the number of cycles required for convergence are comparable to those reported in the literature for other *ab initio* codes, indicating that the gradients are being computed at an acceptable level of accuracy (note that for DFT all codes have this issue since the XC operator must be evaluated numerically). In a future publication we will examine in detail the specific performance of the Jaguar optimizer.

We begin by assessing the reductions in CPU time obtained using the atom centered multiple and J-engine algorithms. Table I presents timings for assembling the multiple portion of the Coulomb field on one of the more expensive (nonupdating) SCF iterations. The total time spent by the multipole code (T-pre+T-assem) is seen to be a relatively small fraction of the total time for J matrix assembly over the grid (T-Jij). Comparing against timings without the multipole method, it is apparent that the multipole method is saving from a factor of 2–3 in the J matrix evaluation over the grid with the largest savings occurring in the largest semi-three-dimensional case, taxol. We note that we have not considered here ultralarge systems, such as have been investigated in Ref. 17 to assess multipole performance. Our

TABLE III. CPU times (SGI R 10k seconds) for calculating the three-center $(aa'|bc)$ Coulomb matrix elements using the multipath method, $J(aa|BC)$, and the Gill–Head-Gordon (Ref. 17), approach, $J(aa|bc)$.

Molecule/basis	N_{basis}	Time $J(aa\|BC)$ (energy/gradient)	Time $J(aa\|bc)$ (Gill method) (energy/gradient)
$C_{42}H_{86}$ 6-31G	550	57/129	163/526
Porphine cc-pVTZ(-f)	678	243/736	683/4627
Porphine 6-31G*	430	59/180	95/560

TABLE IV. Summary of improvements from multipole and J-engine algorithms. Porphine cc-pVTZ(-f) BLYP timings (SGI R 10k minutes). T-Jij: total time to evaluate the J matrix over the grid for 9 SCF iterations/one gradient. T-jana: total time to evaluate analytic corrections to the J matrix over nine SCF iterations/one gradient. T-SCF: total SCF time, T-grad: total gradient time.

Method	T-Jij	T-Jana	T-SCF	T-grad
New	40/10	18/22	78	40
Old	91/17	37/93	147	117

expectation is that much larger gains will be realized in these cases due to the growth in the computational effort to evaluate the Coulomb operator as the system size increases. However, we have chosen to focus in this paper on systems that are in the range of typical academic and industrial applications. Taxol, the largest system we consider, is an important pharmaceutical compound; the new methods provide substantial acceleration for this case. Even for the smaller systems on the order of 50 atoms, though, significant improvements are observed. This reflects the efficacy of the atom-centered multipole approach, which has a negligible amount of overhead yet still is able to incorporate a considerable number of product centers effectively due to the details of its design.

The net scaling of the Coulomb matrix construction is calculated from the timings in Table II, including all analytic integral correction timings. For taxol, which falls between a two- and three-dimensional class, the net scaling is calculated to lie between the linear and quadratic regimes, $N^{1.5}$.

The efficiency of the J-engine method for the $(aa'|bc)$ analytic integrals is displayed in Table III for both energy and gradient. There are factors of 1.6–2.9 improvements in the energy integral evaluation speed and from 3.1–6.2 in the gradient evaulations. The improvements are larger for the gradient and for the correlation-consistent basis sets.

The overall improvements in the J matrix assembly times are presented in Table IV for the case of a porphine BLYPcc-pVTZ(-f)SCF+gradient calculation. In general, the combination of the multipole and J-engine methods results in a factor 2 improvement in the energy evaluations and a factor of 3 in the gradient for this case, both for the components of the J matrix evaluation and in the total SCF/gradient timings.

Overall CPU times are presented in Table V. These results reflect a 5-25*x* speedup as compared to GAUSSIAN 92 timings, as we have demonstrated in detail elsewhere[18] and have partially reproduced in Table VI. Some of the advantage is due to intrinsic efficiencies of PS methods while others can be attributed to the new algorithms described above. The speedups are larger for gradient-corrected DFT due to the advantages conferred by the atom-centered multipole method. These timings represent a major improvement as compared to conventional electronic structure codes, and allow larger and more complex systems to be addressed with DFT methods on a routine basis.

TABLE V. CPU times (minutes) of single-point energy and average energy+gradient times for density functional calculations with Jaguar 3.5 on an SGI Irix 62-r10k work station. All molecules were run in C1 symmetry except for $C_{42}H_{48}$, which has Cs symmetry. The LACVP** basis was used for Fe and the LACVP* basis for Ge.

Molecule	Nbas	SCF-Iters	SCF time	SCF+grad time
		B3LYP 6-31G**		
Porphine	388	10	23	27
sMMO active site	623	19	217	167
$C_{34}H_{38}O_4Si_2Ge$	653	10	95	100
$C_{42}H_{86}$	802	8	46	61
Taxol	1032	11	207	216
		B3-LYP 6-31G**		
Porphine	388	10	39	43
sMMO active site	623	17	337	259
$C_{34}H_{38}O_4Si_2Ge$	653	9	179	210
$C_{42}H_{86}$	802	8	78	105
Taxol	1032	10	421	472
		BLYP cc-pVTZ(-f) single point		
Porphine	678	5	106	
BPh	1172	7	616	
$C_{42}H_{86}$	1740	6	394	
Taxol	1885	8	1428	
		B3-LYP cc-pVTZ(-f) single point		
Porphine	678	5	191	
BPH	1172	7	950	
$C_{42}H_{86}$	1740	6	546	
Taxol	1885	8	2674	

TABLE VI. A comparison of B-LYP SCF CPU times for Jaguar and GAUSSIAN 92. All timings are in hours using a 125 MHz HP 735.

Molecule	Basis	N_{basis}	Jaguar	GAUSSIAN 92
Alanine tetrapeptide	6-31G*	388	0.7	9.2
Porphine	6-31G*	344	1.0	13.0
Porphine	cc-pVTZ(-f)	678	6.7	84.4

V. CONCLUSION

We have presented pseudospectral methods for carrying out both gradient-corrected and hybrid functional DFT calculations. New algorithms were introduced to substantially reduce computational effort for assembly of the Coulomb operator; assembly of the exchange-correlation operator was also accelerated by multigrid techniques. Current performance levels allow calculations on 100–200 atom systems to be carried out routinely on workstations with a modest amount of memory and disk storage.

ACKNOWLEDGMENT

This work was supported in part by a grant from the Department of Energy (DE-FG02-90ERH162) to R.A.F.

[1] A. D. Becke, J. Chem. Phys. **98**, 1372 (1993).
[2] A. D. Becke, J. Chem. Phys. **98**, 5648 (1993).
[3] B. G. Johnson, P. M. W. Gill, and J. A. Pople, J. Chem. Phys. **98**, 5612 (1993).
[4] R. A. Friesner, J. Phys. Chem. **92**, 3091 (1988).
[5] J. M. Langlois, R. P. Muller, T. R. Coley, W. A. Goddard III, M. N. Ringnalda, Y. W. Won, and R. A. Friesner, J. Chem. Phys. **92**, 7488 (1990).
[6] R. B. Murphy, M. D. Beachy, R. A. Friesner, and M. N. Ringnalda, J. Chem. Phys. **103**, 1481 (1995).
[7] R. B. Murphy, W. T. Pollard, and R. A. Friesner, J. Chem. Phys. **106**, 5073 (1997).

[8] R. Kutteck, E. Apra, and J. Nichols, Chem. Phys. Lett. **238**, 173 (1995).
[9] C. A. White, B. G. Johnson, P. M. W. Gill, and M. H. Gordon, Chem. Phys. **253**, 269 (1996).
[10] C. A. White and M. Head-Gordon, J. Chem. Phys. **104**, 2620 (1996).
[11] A. D. Becke, Phys. Rev. A **38**, 3098 (1988).
[12] M. Challacombe and E. Schwegler, J. Chem. Phys. **106**, 5526 (1997).
[13] B. H. Greeley, T. V. Russo, D. T. Mainz, R. A. Friesner, J. M. Langlois, W. A. Goddard III, R. E. Donnelly Jr., and M. N. Ringnalda, J. Chem. Phys. **101**, 4028 (1994).
[14] L. Greenard and V. Rokhlin, J. Comput. Phys. **60**, 187 (1985).
[15] P. M. W. Gill, M. Head-Gordon, and J. A. Pople, J. Phys. Chem. **94**, 5564 (1990).
[16] T. H. Dunning, J. Chem. Phys. **90**, 1007 (1989).
[17] G. Scuseria, M. C. Strain, and M. J. Frisch, Science **271**, 51 (1996).
[18] R. A. Friesner, R. B. Murphy, M. D. Beachy, Y. Cao, W. T. Pollard, and M. N. Ringnalda, J. Phys. Chem. A **103**, 1913 (1999).