JCTC Journal of Chemical Theory and Computation

Parallel Calculation of Coupled Cluster Singles and Doubles Wave Functions Using Array Files

Tomasz Janowski, Alan R. Ford, and Peter Pulay*

Department of Chemistry and Biochemistry, Fulbright College of Arts and Sciences, University of Arkansas, Fayetteville, Arkansas 72701

Received February 27, 2007

Abstract: A new parallel implementation of the Coupled Cluster Singles and Doubles (CCSD) and related wave functions (e.g Quadratic Configuration Interaction, QCI, and Coupled Electron Pair, CEPA) is described, based on the Array Files middleware. The program can handle large basis sets, even without utilizing symmetry, on modest distributed memory workstation clusters. High computational efficiency is achieved by formulating all major operations in terms of matrix multiplications. Timings are provided for systems with 50−228 valence electrons and up to 1144 basis functions, with little or no symmetry. Our largest calculation (QCISD/aug-cc-pVQZ for the parallel displaced benzene dimer) uses 1512 basis functions. Calculations on the benzene dimer show that the usual procedure of estimating the effect of basis set enlargement from second-order Møller−Plesset (MP2) calculations is less reliable than previously assumed. Replacing the weak pair amplitudes in CCSD/QCISD calculations by MP2 amplitudes affects the calculated energy only slightly.

I. Introduction

Coupled Cluster[1] (CC) and related techniques are the most accurate routinely applicable electronic structure methods (for reviews see refs 2−4). Tests performed since the first implementations of Coupled Cluster theory with single and double substitutions (CCSD) in 1978[5−7] have shown that it still has systematic errors (for an excellent treatment of the accuracy of various accurate quantum chemical methods see ref 8). However, adding triple substitutions yields essentially quantitative results for systems with only dynamical electron correlation if large basis sets are employed.[8] Because of the high cost of triples, they are included in most cases perturbatively. Among the several ways of doing this, the CCSD(T)[9,10] method became the most popular. The consistently high accuracy of coupled cluster based methods is important in several areas, for instance in accurate thermochemistry or for benchmarking more approximate methods.

The main disadvantage of CC methods is their high computational cost, both in terms of CPU time and random access and disk memory. This makes larger calculation impossible or impractical on the current generation of computers. Local correlation methods[11−15] eliminate the steep scaling of traditional CC methods for molecules with well localized electronic structure and allow calculations on large systems. However, local methods perform less well for genuinely delocalized systems or for basis sets augmented with diffuse functions. The latter are essential, e.g., for the description of dispersion forces. One obvious way to extend the applicability of full CC methods is to use parallel computing. Clusters of inexpensive personal computer based workstations, introduced independently by many research groups (including ours) 10 years ago, have excellent price/performance ratios and are available with parallel quantum chemistry programs preinstalled.[16] By using the combined resources of the system, it is possible to overcome limitations in computer time, random access memory, and disk memory. Parallelization of the perturbative triples part of CCSD(T) can be accomplished with relatively little communication between nodes, and thus it is less demanding than CCSD.[17,18] The latter requires repeated access to large (four index) arrays and is not easy to implement efficiently in parallel. The aim of the present project is a parallel CCSD program that can handle moderately large systems on modest-size (and therefore widely available) clusters with good quality basis

* Corresponding author e-mail: pulay@uark.edu.

10.1021/ct700048u CCC: $37.00
Published on Web 06/06/2007

sets. This program serves as a starting point for a CCSD(T) program. In our opinion, the ability to handle a system with 10−20 non-hydrogen and 10−20 hydrogen atoms, with basis sets ranging from cc-pVTZ for the largest molecules to aug-cc-pVQZ[19] for the smaller ones, would have the greatest impact at the present stage of computational technology. Such calculations may include up to 2000 basis functions and up to 100 valence electrons (not necessarily simultaneously).

Because of its high computational demand, CCSD appears to be an ideal candidate for parallel implementation. The first distributed-memory parallel CCSD algorithm was described by Rendell et al.[20] for the now defunct Intel Hypercube. However, when our first implementation was finished in the spring of 2005, there were only two working parallel implementations of the full CCSD model: in MOLPRO[21] and in NWChem.[22] Neither of these is described in detail in the literature. The general strategy of our program is most similar to MOLPRO which is not surprising, as both MOLPRO and PQS,[23] the vehicle used to develop the present program, trace their roots back to an early ab initio program developed initially (from 1968) by Prof. W. Meyer (Kaiserslautern, Germany) and one of us (P.P.). The parallel CCSD(T) program in NWChem is based on the algorithm of Kobayashi and Rendell.[24] In the meantime, parallel CCSD(T) has been implemented in GAMESS-US[25] and is under development[26] in the Mainz-Austin-Budapest version of the ACES II package.[27] Both the NWChem and the GAMESS-US parallel implementations use the massively parallel paradigm (hundreds of processors). Our goal was to be able to perform large calculations without using massive parallelism. Access to such a high number of processors, although improving, is still severely limited.

Both existing parallel CCSD codes, NWChem and MOLPRO, use Global Arrays[28,29] (GA) as a parallelization tool. GA simulates shared memory programming on distributed memory computer clusters by striping large arrays across nodes. Originally, it envisioned storage in fast memory only. However, in view of the massive amounts of data needed in large CC calculations, such a strategy is prohibitive on moderately sized computer clusters. Later versions of GA allow accessing data on secondary (disk) storage, but they have to be accessed through the GA subsystems. In view of the large primary memories available on the present generation of small workstations, striping individual matrices (with typical sizes of 1−4 MWords, i.e., 8−32 Mbytes) is not necessary any more and may even be counterproductive. Therefore, we have chosen Array Files[30] (AF), recently developed in this laboratory, as a parallelization tool. Array Files fits the computational requirements of our matrix-formulated CCSD particularly well.

This paper describes a parallel CCSD/QCISD program for closed shell molecules that can perform large calculations on modest workstation clusters. In addition to CC methods, the program can also calculate various approximations to CCSD, such as versions of the Coupled Electron Pair Approximation[31] (we have implemented CEPA-0 and CEPA-2), Quadratic Configuration Interaction (QCISD),[32] and lower orders of Møller−Plesset perturbation theory (MP2, MP3, and MP4(SDQ)) as well as variational Configuration Interaction with single and double substitutions (CISD). These methods can be viewed as approximations to the full CCSD method. Most calculations were carried out using the QCISD method that is comparable in quality with CCSD but is less expensive.

II. Theory

In this section, we recapitulate the theory and establish the notation. The CCSD energy and wave function are defined by the vanishing of the residuals in eqs 1−3:

$$\langle \Phi^0|\mathbf{H} - E|\Psi^0 + \exp(\sum_{ia} T_a^i \mathbf{E}_i^a + \sum_{ijab} T_{ab}^{ij} \mathbf{E}_{ij}^{ab})\Phi^0\rangle = 0 \quad (1)$$

$$\mathbf{R}_c^k = \langle \tilde{\Psi}_k^c|\mathbf{H} - E|\Psi^0 + \exp(\sum_{ia} T_a^i \mathbf{E}_i^a + \sum_{ijab} T_{ab}^{ij} \mathbf{E}_{ij}^{ab})\Phi^0\rangle = 0 \quad (2)$$

$$\mathbf{R}_{cd}^{kl} = \langle \tilde{\Psi}_{kl}^{cd}|\mathbf{H} - E|\Psi^0 + \exp(\sum_{ia} T_a^i \mathbf{E}_i^a + \sum_{ijab} T_{ab}^{ij} \mathbf{E}_{ij}^{ab})\Phi^0\rangle = 0 \quad (3)$$

Here Φ^0 is a normalized reference wave function, in our case a closed-shell determinant, and the substitution operators $\mathbf{E}$ acting on Φ^0 transfer one or two electrons from the occupied space (indices *i,j,k,l*) to the virtual space (*a,b,c,d*). The bra projectors $\tilde{\Psi}_k^c$, $\tilde{\Psi}_{kl}^{cd}$ span the singly and doubly substituted space of $\mathbf{E}_i^a\Phi^0$ and $\mathbf{E}_{ij}^{ab}\Phi^0$. The tilda notation in eqs 2 and 3 emphasizes that the individual substituted functions Ψ may be different on the right- and left-hand sides; only the spaces spanned by them are identical. In other words, we can use different linear combinations of the substituted (sometimes called by the pedagogically unfortunate name "excited") wave functions on the two sides. This becomes important in the spin adapted version of the theory. Spin adaptation confers significant computational advantages for closed-shell reference wave functions.

It has been shown that a biorthogonal version of the spin-adapted closed shell coupled cluster doubles (CCD) theory[33] halves the computational effort for the pair coupling terms, compared to formulations employing fully orthogonal spin-adapted functions. We have called this the generator state formulation since the right-hand basis functions are the generator states advocated by Matsen.[34] It is worth noting that Čižek's original formulation[1] used implicitly generator states. Subsequent work concentrated mostly on orthogonal functions. In the generator state form, the singly and doubly substituted configurations are defined as

$$\Psi_i^a = \mathbf{E}_i^a\Phi^0 = (\mathbf{e}_i^a + \mathbf{e}_{\bar{i}}^{\bar{a}})\Phi^0 = \Phi_i^a + \Phi^{\bar{i}\bar{a}} \quad (4)$$

$$\Psi_{ij}^{ab} = \mathbf{E}_{ij}^{ab}\Phi^0 = \mathbf{E}_i^a\mathbf{E}_j^b\Phi^0 = \Phi_{ij}^{ab} + \Phi^{\bar{i}\bar{j}\bar{a}\bar{b}} + \Phi_{ij}^{a\bar{b}} + \Phi_{ij}^{\overline{ab}}; \quad i \geq j \quad (5)$$

The spin-summed single substitution operators are the sums of spin−orbit substitutions:

$$\mathbf{E}_i^a = \mathbf{e}_i^a + \mathbf{e}^{\bar{i}\bar{a}} \quad (6)$$

In eqs 4−6, $\mathbf{e}$ is a spin-orbital substitution operator, replacing an occupied spin orbital *i* or *j* by virtual orbitals *a* or *b*; indices without overbars refer to α spin and with overbars

β spin. The functions Φ are substituted Slater determinants where the subscript spin-orbitals have been replaced by the superscript spin-orbitals. Interchanging a and b in eq 5 generates a linearly independent doubly substituted function (unless $i=j$ or $a=b$) that is not orthogonal to Ψ_{ij}^{ab}. This pairwise nonorthogonality causes no computational problems if the left-hand (contravariant) projection functions are biorthogonal to the substituted functions on the right-hand side. This is achieved by defining the projection functions as the biorthogonal partners of the expansion functions[33]

$$\tilde{\Psi}_{i}^{a} = \frac{1}{2}\Psi_{i}^{a} \tag{7}$$

$$\tilde{\Psi}_{ij}^{ab} = \frac{1}{6}(2\Psi_{ij}^{ab} - \Psi_{ij}^{ba}) \tag{8}$$

The present program is based on the elegant matrix formulation of the singles and doubles correlation problem introduced by Meyer[35] under the acronym SCEP (Self-Consistent Electron Pair) theory coupled with the generator state spin adaptation. This yields very compact formulas for the CC equations. For reference in the next section, we give the CCD equations[33] explicitly below. Although CCD is seldom used by itself today, it constitutes the computationally most significant part of the CCSD equations.

As usual in SCEP theory, integrals with two internal (occupied) indices are collected in internal Coulomb and exchange matrices:

$$\mathbf{J}_{ab}^{ij} = (ij|ab)\ \mathbf{K}_{ab}^{ij} = (ai|jb) \tag{9}$$

The CCD amplitudes are calculated by iteratively refining the doubles amplitudes until the doubles residuals vanish:

$$\mathbf{R}^{ij} = \mathbf{K}^{ij} + \mathbf{K}[\mathbf{T}^{ij}] + \mathbf{Q}^{ij} + (\mathbf{Q}^{ji})^{\dagger} + \mathbf{G}^{ij} + (\mathbf{G}^{ji})^{\dagger} = \mathbf{0} \tag{10}$$

Here

$$\mathbf{Q}^{ij} = \mathbf{T}^{ij}(\mathbf{F}\text{-}\mathbf{A}) + {}^{1}/_{2}\Sigma_{k}[(2\mathbf{T}^{ik} - \mathbf{T}^{ki})\mathbf{Y}^{kj} - \mathbf{T}^{ki}\mathbf{Z}^{kj} - 2(\mathbf{T}^{ki}\mathbf{Z}^{kj})^{\dagger} - \beta_{ki}\mathbf{T}^{kj}] \tag{11}$$

$$\mathbf{A} = \Sigma_{kl}(2\mathbf{K}^{kl} - \mathbf{K}^{lk})\mathbf{T}^{lk} \tag{12}$$

$$\mathbf{Y}^{kj} = 2\mathbf{K}^{kj} - \mathbf{J}^{kj} + \Sigma_{l}(2\mathbf{K}^{kl} - \mathbf{K}^{lk})(2\mathbf{T}^{lj} - \mathbf{T}^{jl}) \tag{13}$$

$$\mathbf{Z}^{kj} = \mathbf{J}^{kj} - \Sigma_{l}\mathbf{K}^{lk}\mathbf{T}^{jl} \tag{14}$$

$$\beta_{ki} = \mathbf{F}_{ki} + \Sigma_{l}\mathrm{Tr}[(2\mathbf{K}^{kl} - \mathbf{K}^{lk})\mathbf{T}^{li}] \tag{15}$$

$$\mathbf{G}^{ij} = \Sigma_{kl}\alpha_{ij,kl}\mathbf{T}^{kl} \tag{16}$$

$$\alpha_{ij,kl} = (ik|jl) + \mathrm{Tr}[\mathbf{T}^{ij}\mathbf{K}^{lk}] \tag{17}$$

In the above formulas, the bold-faced quantities are matrices in the external (virtual) space, **F** is the Fock matrix, and the external exchange matrix is defined as

$$\mathbf{K}[\mathbf{T}^{ij}]_{ab} = \Sigma_{cd}(ac|bd)(\mathbf{T}^{ij})_{cd} \tag{18}$$

The quantities $(ik|jl)$ and $(ac|bd)$ are two-electron integrals in the Mulliken notation. The exact notation in eqs 10−17 is that of Hampel et al.[36] who have generalized the CCD equations of ref 33 to the CCSD case. Prior to this, Scuseria at al.[37] developed a formulation of the CCSD theory that achieves the same computational savings but does not use a matrix form. We consider the matrix/tensor formulation preferable, not only because of its simplicity but also because modern computers are very efficient for matrix manipulations, particularly matrix multiplications.

There are several plausible orbital choices for both the internal (occupied) and the external (virtual) space. The occupied orbitals can be either canonical or localized. The former usually converge slightly faster but are less efficient for the utilization of sparsity; symmetry is also simpler to implement with localized orbitals. The virtual space in the above formulation, just like SCEP,[35] can be easily generalized to nonorthogonal atomic basis functions (AOs) or AOs projected against the internal space[11,12] instead of virtual molecular orbitals (MOs). This makes it particularly suitable for AO-based local correlation theories.[11,12,38] Our program can use either AOs or canonical MOs in the virtual space; the transformation between the MO and AO representations[11] is straightforward. Sparsity can be better exploited in the AO form, but the dimension of the matrices (the number of AOs) is higher than in the MO representation (the number of virtual orbitals). Note that all representations yield strictly identical results (within the limits of numerical precision) if no further approximations are made.

Even if AOs are used for the virtual space, updating the amplitudes is done in an orthogonal MO basis, according to first-order perturbation theory as

$$\Delta\mathbf{T}_{ab}^{ij} = -\mathbf{R}_{ab}^{ij}/(\epsilon_a + \epsilon_b - \epsilon_i - \epsilon_j + \delta) \tag{19}$$

where a and b are canonical virtual orbitals, ϵ_a and ϵ_b are their orbital energies, and i and j are either canonical occupied or localized orbitals. In the first case, ϵ_i and ϵ_j are orbital energies; in the second case, they are the Coulson energies of the localized orbitals, e.g., $\epsilon_i = \langle\varphi_i|\mathbf{F}|\varphi_j\rangle$. The quantity δ is a level shift and changes the convergence rate but has no effect on the converged results.

The same program is used to calculate all wave functions available in the program (CEPA-2 and CEPA-0, MP3, MP4-(SDQ), CCD, QCISD, QCID, CISD, and CID), except MP2, since all these many-body methods are computationally simplified versions of CCSD.

III. Algorithm

The parallel CCSD program has been implemented in the academic version of the PQS[23] program package. In its basic architecture, it is similar to the original Self-Consistent Electron Pair program,[39] our earlier Local Electron Correlation program,[11,12] and the implementation in MOLPRO,[13,40] with special attention paid to parallel performance.

The first step of the algorithm is the calculation and storage of the internal Coulomb and exchange operators, eq 9, and the determination of the MP2 energy and amplitudes. The latter serves as the first approximation to the CC amplitudes and can be substituted for the pair amplitudes of weak pairs[12] in a localized calculation with negligible loss of accuracy

Table 1. Megaflop Ratings for Dense Matrix Multiplication[a]

matrix dimensions	method	Mflops/s
1000 × 1000	DGEMM	5000
1000 × 1000	DDOT	487
1000 × 1000	DAXPY	307
1000 × 50	DGEMM	3570
1000 × 25	DGEMM	2600
1000 × 15	DGEMM	1970

[a] On a 3.2 GHz Intel Nocona processor, using the Goto BLAS library.[45]

and considerable gain in efficiency (see the Results section). The **J** and **K** matrices are distributed on the aggregate disk storage of the nodes in the cluster. We use the recently developed Array Files[30] (AF) for all distributed disk storage. AF allows transparent access to disk records distributed across nodes in a computer cluster. Calculation of the **J** and **K** matrices and the MP2 amplitudes constitutes only a small fraction of the total computational time and follows our efficient canonical MP2[41] and parallel MP2[42] algorithms. Note, however, that the MP2 algorithms become iterative if localized orbitals are used, as shown in the first full formulation of MP2 with noncanonical orbitals;[43] the quadruples contributions in MP4 likewise become iterative.[11] An alternative to iterative noncanonical MP2 is the Laplace transform MP2 of Almlöf.[44]

Our main algorithmic goal was, besides minimizing disk access and internode communication, to formulate all computationally significant operations as matrix multiplications. Note that this differs significantly from the "vectorization" strategy of the 1980s and early 1990s. On a typical vector computer, all typical vector operations run at approximately the same speed. On modern CPUs, arithmetic operations are so fast that the main computational bottleneck is fetching data from memory. Matrix multiplication, if implemented in efficient blocked form, allows the reuse of data in cache memory, leading to nearly theoretical efficiency (on many modern microprocessors, the floating point operation rate is twice the clock rate). Table 1 shows that the dot product form of matrix multiplication (DDOT) is 10.5 times slower than a state-of-the art level 3 BLAS routine[45] DGEMM for a dense 1000 × 1000 matrix, and the DAXPY ("outer product") form is 16.7 times slower. The megaflop rating for multiplying two 1000 × 50 matrices is over 70% of the limiting rate, and even 1000 × 15 matrices give nearly 2 Gflops/s (40%) performance.

For large basis sets, the computational effort is usually dominated by the external exchange, the scaling of which is $O(n^2N^4)$ vs $O(n^3N^3)$ for the other sixth-order terms. Here n is the number of correlated occupied orbitals, and N is either the number of AOs (in the AO formulation) or the number of virtual orbitals. The external exchange term is evaluated in an integral-direct manner by transforming the CC amplitudes to AO basis, evaluating eq 18 in AO basis according to

$$\mathbf{K}[\mathbf{T}^{ij}]_{\mu\lambda} = \Sigma_{\nu\sigma}(\mu\nu|\lambda\sigma)\mathbf{T}_{\nu\sigma} \tag{20}$$

In eq 20, μ, ν, λ, and σ denote AOs and i,j correlated occupied orbitals. The resulting **K** matrices are transformed to the virtual basis used in the program. Our program can use either canonical virtual orbitals or projected atomic orbitals. This strategy, based on Meyer's SCEP,[35] avoids the storage of integrals with three and four virtual indices and has been adopted by several programs: the Saebo-Pulay local correlation program,[11,12] MOLPRO,[40] and the program of Kobayashi and Rendell.[24] Without the integral-direct calculation of the external exchange, the storage of integrals with three and four virtual indices becomes a bottleneck, particularly on a single node. E.g. the QCISD calculation with the aug-cc-pVQZ calculation for the benzene dimer, described later, would require about 5.9 Tbytes (5900 Gbytes) for the storage of the all-external ($ab|cd$) integrals alone if symmetry is disregarded. However, in view of rapidly increasing disk capacities and the possibility of distributed storage, storing all transformed integrals on distributed disk memory may be a viable option in the near future.

The computational cost of the external exchange can be reduced by a factor of 2 if symmetric and antisymmetric combinations of the AO integrals are used, according to

$$\Sigma\mathbf{K}^{\pm}[\mathbf{T}^{ij}]_{\mu\lambda} = \Sigma_{\nu\sigma}(1+\delta_{\nu\sigma})^{-1}[(\mu\nu|\lambda\sigma) \pm (\mu\sigma|\lambda\nu)][(\mathbf{T}^{ij})_{\nu\sigma} \pm (\mathbf{T}^{ij})_{\sigma\nu}] \quad \mu \geq \lambda, \nu \geq \sigma \tag{21}$$

$$\mathbf{K}[\mathbf{T}^{ij}]_{\mu\lambda} = {}^1/_2(\mathbf{K}^{+}[\mathbf{T}^{ij}]_{\mu\lambda} + \mathbf{K}^{-}[\mathbf{T}^{ij}]_{\mu\lambda}); \quad \mathbf{K}[\mathbf{T}^{ij}]_{\lambda\mu} = {}^1/_2(\mathbf{K}^{+}[\mathbf{T}^{ij}]_{\mu\lambda} - \mathbf{K}^{-}[\mathbf{T}^{ij}]_{\mu\lambda}) \tag{22}$$

This algorithm was first explicitly described in ref 11, but it appears that it had been used, at least for symmetrical molecules, in the original SCEP program.[35] It has been adopted by Scuseria et al.[37] and Kobayashi and Rendell.[24] A disadvantage, which it shares with our MP2,[41] is that the number of AO integrals evaluated is approximately four times larger than the minimum necessary if all integral permutation symmetry is utilized. The evaluation of eq 21 requires formally $n^2N^4/2$ floating-point operations, while integral evaluation is CN^4 where C is relatively large and independent of the number of correlated internal orbitals n. Therefore this algorithm is most advantageous for large systems ($n^2 \gg C$).

The matrix formulation used here leads automatically to a highly efficient program for almost all terms. The exception is the external exchange operator. If performed for a single (ij) pair, eq 20 is a scalar product that performs poorly on modern CPUs. To increase its performance, we try to construct **K** matrices simultaneously for as many (ij) pairs as local fast memory permits. Note that it is important to use the size of the actual fast memory and not the virtual memory here. Treating (ij), ($\mu\lambda$), and ($\nu\sigma$) as single indices, eq 20 becomes a matrix operation, although the range of (ij) and ($\mu\lambda$) is much smaller than that of ($\nu\sigma$). A pseudocode of the essential parts of the algorithm is shown in Figure 1. Note that, depending on the size of the shells and the available memory, a large number of ($\mu\lambda$) shells may be processed together. This improves both the CPU timing because the matrix dimensions become larger but has an even bigger effect by reducing the I/O needed to fetch the amplitudes from the distributed disk storage. Although

```
do M=1,NSH   (shells of AOs)
  do Λ=1,M   (shells of AOs)
    Calculate all AO integrals (μν|λσ), μ∈M, λ∈Λ
    Form the matrices X±μλ,νσ=(μν|λσ)±(μσ|λν) ν≥σ
    Accumulate the matrices X in memory until the memory
     set aside for integrals is full
    if the integral memory is full
      do for batches of (ij)
        Read a batch of T^ij amplitudes and transform
        them to AO basis; put the transformed
        amplitudes in the matrix Tνσ,ij
        Calculate K±μλ,ij = Σνσ X±μλ,νσ×Tνσ,ij [Eq.(21)]
        Calculate Kμλ,ij [Eq. (22)]
      end do batches of (ij)
      release integral memory
    end if
  end do Λ
end do M
```

Figure 1. Pseudocode of the external exchange matrix construction, eqs 21 and 22.

MOLPRO uses a different algorithm for building the external exchange operators, it incorporates a similar "shell merging" feature.[40]

Because of the high cost of the external exchange operator, it is important to utilize the natural sparsity of the integral list. However, this is possible only to a limited extent. According to our experience one has to maintain a sharp integral threshold. Aggressive neglect of the integrals in eqs 20−22 can cause the CCSD iteration to diverge, particularly if the basis contains diffuse functions. Integral sparsity is used in the following manner. Integrals $(\mu\nu|\lambda\sigma)$, for all $\nu\sigma$ and as many $\mu\lambda$ as the memory allows, are collected in the fast memory as a matrix with composite row index $\mu\lambda$ and column index $\nu\sigma$. This matrix is divided into horizontal stripes, usually so that all $\mu\lambda$ pairs that come from a common pair of shells constitute a stripe. Each stripe is inspected for columns having all integrals below a threshold and is independently compressed by removing these columns. Integrals with basis functions from the same shells share the sparsity pattern, and therefore most negligible integrals are removed at this stage. An indexing array keeps track of the numbering of the original columns. When the RAM memory is full, the multiplication with the amplitudes is performed, separately for each stripe. To enable the use highly efficient dense matrix multiplication routines, the amplitude matrices have to be also compressed by removing the rows corresponding to columns removed from a stripe. For higher angular momentum functions the stripes are sufficiently wide to guarantee high performance in this step, cf. Table 1. The disadvantage of his algorithm is that the same amplitude matrix has to be compressed separately for each integral stripe. This is an argument for having as big integral stripes as possible, but in this case sparsity deteriorates. Conversely, sparsity is best if each stripe covers only one shell pair. However, this leads to small dimensions for low angular momentum functions and the corresponding loss of efficiency in the matrix multiplication. In the extreme case of two *s* type shells, the matrix multiplication becomes a dot product which is ∼10 times less efficient. The best compromise appears to be to treat larger shell pairs (*pd*, *dd*, *df*, *ff*) as separate stripes but merge smaller shell pairs to have a minimum dimension of ∼15.

The screening algorithm speeds up the calculation by lowering the amount of integrals calculated, decreasing the flop count during matrix multiplication, and it also saves memory because the zero columns are not kept in memory, so more integrals can be stored before performing the multiplication with amplitudes. This allows reducing I/O considerably: the more integrals can be stored in memory the fewer disks reads of amplitudes is needed. For very large calculations (more than 1500 basis functions on a computer with modest memory), disk I/O becomes the dominant part of the EEO's calculation.

In the present implementation of the CCSD equations the **Q**, **Y**, and **Z** matrices are precalculated and stored on disk before they are used in the residuum construction loop. In order to minimize disk access, the calculation is performed by blocks using a method similar to that employed to speed up matrix multiplication by taking advantage of fast cache memory. Here the RAM memory plays the role of the cache.

This technique is illustrated for the calculation of $\tilde{Q}^{ij}$, the **Y** contributions to $\mathbf{Q}^{ij}$ in eq 11. The $\tilde{Q}^{ij}$ matrix is considered the *ij* element of the supermatrix Λ, $2\mathbf{T}^{ij} - \mathbf{T}^{ji}$ is the *ij* element of a supermatrix Ω, and the matrix $\mathbf{Y}^{ij}$ is an element of the supermatrix Θ. The calculation of $\tilde{Q}^{ij}$

$$\tilde{Q}^{ij} = \sum_k (2\mathbf{T}^{ik} - \boldsymbol{T}^{ki})\mathbf{Y}^{kj} \tag{23}$$

can be written as $\Lambda = \Omega \cdot \Theta$, i.e. as a matrix multiplication where each matrix element is a matrix itself. By dividing Λ

```
Do ii=1, N  (N is number of submatrices in a row)
  Do jj=1, M  (M is number of submatrices in a column)
    Reserve space for the Q^ij matrices, i ∈ ii, j ∈ jj.
    Initialize Q^ij with zero values.
    Do k=1, n  (n is number of correlated orbitals)
      Store all matrices T^ik, i ∈ ii group
      Store all matrices K^kj, j ∈ jj group
      Calculate from them all possible contributions
            to Q^ij, i ∈ ii, j ∈ jj group.
      Add the calculated contributions to
         current values of Q^ij
    End do
    Write all Q^ij
  End do
End do
```

Figure 2. Pseudocode of the construction of the **Y** contribution to the pair coupling terms, eqs 11 and 23.

into smaller square or rectangular submatrices and calculating all elements of a submatrix before proceeding to the next submatrix, the I/O associated with this operation can be substantially reduced because the matrices in fast memory are reused several times. The submatrix sizes are determined by the available memory and the number of nodes working on the calculation, so that each of them gets at least one submatrix to work on. The pseudocode of this algorithm is shown in the Figure 2.

Some terms in eqs 11−17, e.g. $\alpha_{ij,kl}$ in eq 17, are expressed as traces of matrix products, Tr(**AB**); this operation is in effect a dot (scalar) product of two vectors and is not efficient on modern CPUs. The efficiency of this part of the code can be significantly increased by a method similar to the one used for the external exchange operator. Introducing the composite indices *ij*, *kl*, and *ab* for **T** and **K** transforms eq 17 in a matrix multiplication:

$$\alpha_{ij,kl} = (ik|jl) + \sum_{ab} \mathbf{T}_{ij,ab} \mathbf{K}_{kl,ab} \tag{24}$$

However, the 4-index quantities $\mathbf{T}_{ij,ab}$ and $\mathbf{K}_{kl,ab}$ do not fit into fast memory for larger systems. Therefore the indices *ij* and *kl* are subdivided in blocks of appropriate size that that allow the storage of these quantities but still give reasonably high efficiency in the matrix multiplications. A similar method is used for the calculation of the **G** matrices, eq 16.

The parallelization of the CCSD program with the AF tool was designed as a simple master-slave scheme. All computational tasks in our code are formulated as relatively long loops, typically over pairs of internal (occupied) orbitals or pairs of AO indices. The master assigns the current task, labeled by the loop index, dynamically to the first idle slave. No other programming is needed, because all data can be transparently accessed from each node. Figure 3 shows the algorithm for the distributed computation of the residual matrices, eq 10, parallelized by a pair of occupied indices *ij*. The Coulomb, exchange, EEO, and three-external integral modules were parallelized similarly, except that the main loop was over μ and ν AO indices instead *ij* pairs. This is similar to the method used in our parallel MP2 algorithm.[30] To minimize I/O, the **Q**, **Y**, and **Z** matrices are calculated in batches of *ij*'s in both the serial and the parallel program. This makes the parallel code more sensitive to load balancing. We have both a dynamic distribution of these batches and a static distribution that usually allows better load balancing because the computational task per *ij* index pair is always the same.

As a message passing software the PVM was used, both for communication with Array Files and master-slave messages. However, both the base PQS code and the AF subsystem also have an MPI version. The major network load comes from nodes to AF traffic (file data reads and writes), the master-slave communication reduces to control messages only.

IV. Results

Table 2 shows representative QCISD timings for some medium-sized molecules with basis sets ranging from small (6-31G*) to large (PC-2[46] and aug-cc-pVTZ[47]). The number of atoms varies from 20 to 73, and the number of basis functions varies from 282 to 1144. Most calculations were run on a 15-node home-built cluster. The timings demonstrate that CCSD/QCISD calculations can be performed routinely for drug-size molecules with good basis sets and for larger molecules with smaller basis sets. Except for glycine-10, the calculations were performed with an earlier version of the code in which some smaller computational tasks were not yet parallelized. The current code is about 13% faster for 8 nodes and 20% faster for 15 nodes.

Table 3 shows timings for the benzene dimer at the QCISD level using six different basis sets, including very large ones (over 1500 basis functions). The benzene dimer became an important benchmark in testing ab initio techniques for the prediction of dispersion forces, in particular $\pi-\pi$ interactions.[48−52] It is surprisingly difficult to obtain converged results that are correct for the right reason, i.e., without semiempirical adjustment. SCF theory, and most density functional methods, if corrected for basis set superposition error, give a repulsive potential curve. The simplest theoretical level that accounts qualitatively for $\pi-\pi$ attraction is MP2. However, MP2, in the basis set limit, overestimates the well depth by as much as a factor of 2 and consequently underestimates the van der Waals distance. CCSD, and the very similar QCISD, overcorrect this defect and lead to an *underestimation* of the well depth (and an overestimation of the distance).

Table 3 demonstrates that, as expected, the external exchange part becomes dominant as the basis set increases

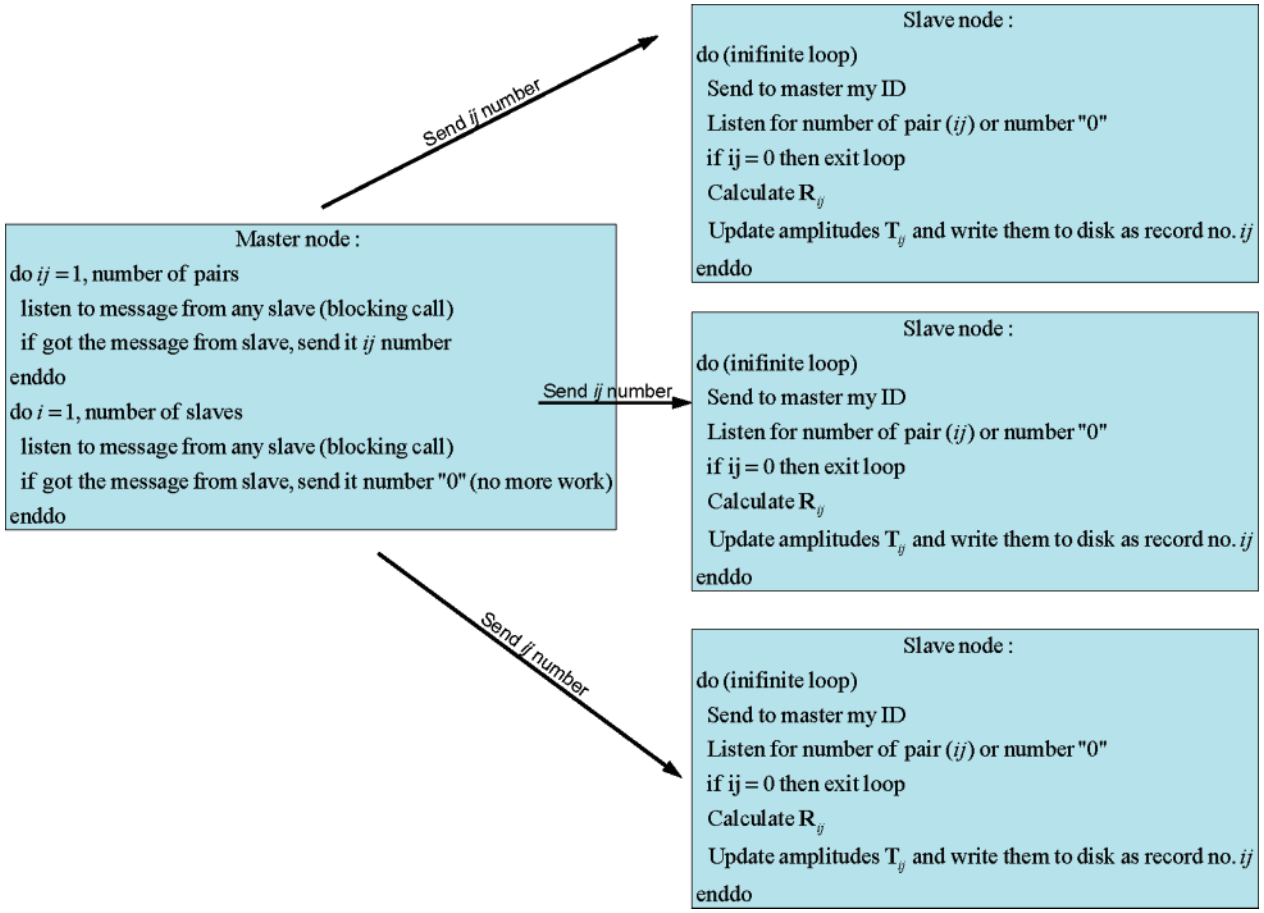

Figure 3. The pseudocode of the parallelization scheme for the main CCSD residuum loop. The same scheme was used for all other quantities calculated.

Table 2. Elapsed Times per QCISD Iteration for Medium-Sized Molecules with a Variety of Basis Sets[a]

molecule	empirical formula	basis set	n^b	N^c	nodes	iterations	time/iter (min)
naphthalene+H_2	$C_{10}H_{10}$	aug-cc-pVTZ[d]	25	690	14	11	98
aspirin	$C_9H_8O_4$	6-311G**	34	282	6	12	12
sucrose		6-31G*	68	366	11	*e*	51
sucrose	$C_{12}H_{22}O_{11}$	6-311G**	68	546	11	*e*	170
yohimbine	$C_{21}H_{26}N_2O_3$	PC-2 [f]	69	1144	20	18[g]	2074
glycine-10[h]	$C_{20}H_{32}N_{10}O_{11}$	6-31G*	114	524	15	12	409

[a] On a cluster of 3 GHz dual-core Pentium D processors, except for yohimbine which was run on 20 nodes of the University of Arkansas Red Diamond a 128-node cluster of dual-processor nodes equipped with 3.2 GHz Xeon processors. [b] Number of correlated occupied orbitals. [c] Number of basis functions. [d] Reference 47. [e] These calculations were stopped before full convergence was obtained. [f] Reference 46. [g] The number of iterations is larger than the usual ~12 because the DIIS extrapolation had to be restricted to 4 vectors, instead of the usual 6, due to limited local storage on the Red Diamond cluster. [h] Glycine polypeptide, α helix conformation.

for the same molecule. However, the composition of the basis set also has an effect: diffuse functions reduce the sparsity of the integral list and increase the cost of the external exchange operator. The recalculation of the integrals, although lower scaling in principle, is also quite expensive. The CPU efficiencies in Table 3 are reasonably high, particularly for larger basis sets. The best CPU efficiency is obtained for moderate size systems, i.e., 500−1000 basis functions. For smaller systems the latency associated with message passing becomes significant, as the amount of actual calculation is very small and a parallel synchronization becomes major part of the calculation time. For the systems bigger than 1000 basis functions disk I/O becomes a bottleneck, especially in the external exchange, because multipassing is necessary. The performance of our program strongly depends on the network throughput and increases dramatically if all nodes are directly connected to the same network switch. Unfortunately, because of the resource limitations we were unable to perform all calculations with the same network configuration.

For big CCSD type calculations memory becomes as important a resource as CPU speed. Since the current algorithm reads the amplitudes many times in order to multiply them by the integrals stored in memory, doubling the amount of memory decreases I/O by the same factor. For large calculations (1500 basis function and more) that are limited by disk I/O, the computational speed almost doubles. Note the timing difference for the aug-cc-pvtz calculation using 32-bit code with 180 MW of memory used and 64-bit code with 380 MW.

Table 3. Timings for Benzene Dimer Using Different Basis Sets on a Cluster of 3.2 GHz Dual-Processor Xeon Machines[f]

system/basis	N^a	N^b	memory/slave (MW)	code (bit)	no. of slaves	time/iter (m)	EEO (%)[c]	CPU efficncy[d] (%)
M, cc-pvdz	15	228	180	32	20	1.33	55	45
M, cc-pvtz	15	528	180	32	20	13.5	63	63
M, cc-pvqz	15	1020	180	32	20	160	83	80
M, aug-cc-pvdz	15	384	180	32	20	10.1	79	75
M, aug-cc-pvtz	15	828	180	32	20	117	87	75
M, aug-cc-pvqz	15	1512	400	64	31	858	94	81[e]
D, cc-pvdz	30	228	180	32	20	2.67	49	40
D, cc-pvtz	30	528	180	32	20	31.7	62	70
D, cc-pvqz	30	1020	180	32	20	682	90	60
D, aug-cc-pvdz	30	384	180	32	20	17.2	69	70
D, aug-cc-pvtz	30	828	180	32	20	275	87	70
D, aug-cc-pvtz	30	828	380	64	20	175	85	80[e]
D, aug-cc-pvqz	30	1512	470	64	31	1917	95	65[e]

[a] Number of correlated orbitals. [b] Number of atomic basis functions. [c] Calculation of the atomic orbital integrals and their contraction with the amplitudes to form the External Exchange Operator (EEO), eq 18. [d] The CPU efficiency is calculated as the sum of the CPU times on the slaves divided by the product of the elapsed time and the number of slaves. [e] The CPU timings are not reliable for the 64-bit architecture (kernel from Linux 2.4 series) for the multithreaded parts of program. [f] The machines were equipped with 4 GB of RAM memory (1 MW ≈ 8 MB), D = dimer in dimer basis Set, M = monomer in dimer basis set. No symmetry was used.

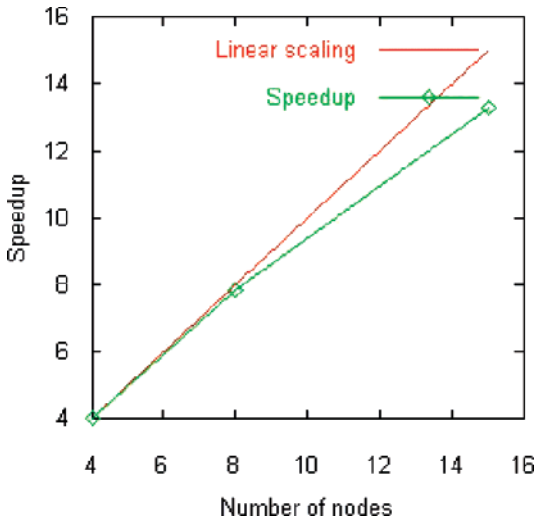

Figure 4. Parallel scaling of a calculation on glycine-10 (see Table 2). The efficiency of the 4-node calculation is taken as 4.

Table 4. Counterpoise Corrected Binding Energies for Benzene Dimer in Different Basis Sets[a]

basis set	SCF	MP2	MP3	MP4(SDQ)	QCISD
aug-cc-pvdz	−5.168	4.219	0.765	0.919	0.833
aug-cc-pvtz	−5.159	4.645	1.097	0.976	0.998
aug-cc-pvqz	−5.157	4.790	1.216	0.947	1.047

[a] At the dimer geometry of Sinnokrot and Sherrill[52] (interplane distance = 3.4 Å, lateral shift = 1.6 Å).

Figure 4 shows the scaling of the glycine-10 calculation from 4 to 15 nodes. Scaling is almost linear from 4 to 8 nodes but only 88% efficient in going from 8 to 15 nodes.

Table 4 shows the counterpoise corrected binding energies obtained for the parallel displaced benzene dimer geometry at various levels of theory and for six different basis sets using the geometry of Sinnocrot and Sherrill.[52] This geometry has been optimized at the MP2/aug-cc-pVQZ* level, and, due to the overestimation of the binding energy, the optimized interplane distance, 3.4 Å, is likely to be too small.

Table 5. Counterpoise Corrected Binding Energies for Benzene Dimer in Different Basis Sets[a]

basis set	SCF energy	MP2 energy	QCISD energy
cc-pvdz	−2.554	1.726	0.159
cc-pvtz	−2.414	3.184	1.032
aug-cc-pvdz	−2.392	3.623	1.453
aug-cc-pvtz	−2.359	3.856	1.524

[a] The shift and distance optimized at the QCISD/aug-cc-pvtz level (distance = 3.675 Å, shift = 1.870 Å).

The efficiency of our program enabled us to optimize the intermolecular distance and shift of the parallel displaced benzene dimer at the aug-cc-pvtz/QCISD level. The optimum interplane distance we found is $R_1 = 3.675$ Å, and the lateral shift is $R_2 = 1.870$ Å. As QCISD underestimates the binding energy, R_1 is almost certainly too long; the most likely distance in the real molecule is probably bracketed between these two values and is estimated to be around 3.55 Å, close to the estimated value of ref 52. Table 5 shows the calculated binding energies at the aug-cc-pvTZ/QCISD geometry.

Tables 4 and 5 allow a critical evaluation of a frequently used procedure, the extrapolation of correlation energy from smaller basis calculations to larger basis sets, using MP2 energy increments. As these tables show, this procedure, although reasonably accurate for total correlation energies, overestimates the basis set effect on the binding energy, just like MP2 overestimates the binding energy. As Tables 4 and 5 show, the change in the binding energy on enlarging the basis set is overestimated by almost a factor of 3 at the MP2 level, relative to QCISD, both for the DZ-TZ and the TZ-QZ transition. Table 5 also shows that neither MP3 nor MP4-(SDQ) perform significantly better than MP2. Sinnocrot and Sherrill argue, on the basis of calculations carried out with smaller basis sets at the CCSD(T) level, that the effects of higher substitutions, $\Delta\text{CCSD(T)} = E(\text{CSD(T)}) - E(\text{MP2})$, are not sensitive to the basis sets. However, this would be true only in the unlikely case if the effect of triple substitutions cancels the effect of CCSD/QCISD. We believe that the extrapolated binding energy derived by Sinnokrot

Table 6. CCSD Weak Pairs Amplitudes Approximated by the MP2 Amplitudes[a]

threshold	no. of strong pairs	dimerization energy	total elapsed time per iteration	elapsed time for EEO	elapsed time for Q terms
0.0	1225	−1.717	3404	1057	380
1e-5	756	−1.719	3131	796	320
1e-4	514	−1.746	2982	638	300
5e-4	345	−1.625	2771	529	275

[a] The accuracy achieved for various thresholds and timings for dimer calculation, 12 nodes. If the weak pair's energy is below the given threshold, then the pair amplitudes are substituted by MP2 amplitudes.

and Sherrill somewhat overestimates the binding energy. Similar results have been obtained in a recent work of Hill et al.[53]

Table 6 presents the results for the MP2 weak pair approximation. In this approach, the amplitudes of the weak pairs (pairs with MP2 correlation energy below a threshold) are kept fixed at the MP2 level throughout the whole iteration process. The remaining pairs are treated as strong and are fully optimized. The external exchange, **Q** and **G** matrices for weak pairs are not needed and omitted. However, in the first iteration the external exchange is calculated for all pairs because they are needed for the singles residual calculation.

As Table 6 and our other preliminary results show, the interaction energies are well reproduced. However, the overall improvement in timings is disappointing, mainly because the evaluation of the AO integrals imposes a significant overhead. It appears that this approximation works best for a large molecule with a moderate basis set. Another reason for the less-than-expected gain is that the current implementation does not take advantage of the fixed amplitudes in the calculation of the **Y** and **Z** matrices because this would require additional disk storage and would interfere with the blocking algorithm.

Acknowledgment. This work was supported by the National Science Foundation under grant numbers CHE-0219267000 and CHE-0515922 and by the Mildred B. Cooper Chair at the University of Arkanses.

References

(1) Cizek, J. *J. Chem. Phys.* **1966**, *45*, 4256−4266.

(2) Bartlett, R. J. In *Modern Electronic Structure Theory*; Yarkony, D. R., Ed.; World Scientific: Singapore, 1995; pp 1047−1131.

(3) Lee, T. J.; Scuseria, G. E. In *Quantum Mechanical Electronic Structure Calculations*; Langhoff, S. R., Ed.; Kluwer: Dordrecht, 1995; pp 47−108.

(4) Crawford, T. D.; Schaefer, H. F., III *Rev. Comput. Chem.* **2000**, *14*, 33−136.

(5) Taylor, P. R.; Bacskay, G. B.; Hush, N. S.; Hurley, A. C. *J. Chem. Phys.* **1978**, *69*, 1971−1979.

(6) Pople, J. A.; Krishnan, R.; Schlegel, H. B.; Binkley, J. S. *Int. J. Quantum Chem. Symp.* **1978**, *14*, 545−560.

(7) Bartlett, R. J.; Purvis, G. D., III *Int. J. Quantum Chem. Symp.* **1978**, *14*, 561−581.

(8) Helgaker, T.; Jørgensen, P.; Olsen, J. *Molecular Electronic Structure Theory*; Wiley: Chichester, 2000; pp 817−833.

(9) Raghavachari, K.; Trucks, G. W.; Pople, J. A.; Head-Gordon, M. *Chem. Phys. Lett.* **1989**, *157*, 479−483.

(10) Bartlett, R. J.; Watts, J. D.; Kucharski, S. A.; Noga, J. *Chem. Phys. Lett.* **1990**, *165*, 513−522.

(11) Saebo, S.; Pulay, P. *J. Chem. Phys.* **1987**, *86*, 914−922.

(12) Saebo, S.; Pulay, P. *J. Chem. Phys.* **1988**, *88*, 1884−1890.

(13) Hampel, C.; Werner, H.-J. *J. Chem. Phys.* **1996**, *104*, 6286−6297.

(14) Schütz, M.; Werner, H.-J. *J. Chem. Phys.* **2001**, *114*, 661−681.

(15) Subotnik, J. E.; Sodt, A.; Head-Gordon, M. *J. Chem. Phys.* **2006**, *125*, 074116/1−12.

(16) See: www.pqs-chem.com.

(17) Baker, D. J.; Moncrieff, D.; Saunders, V. R.; Wilson, S. *Comput. Phys. Commun.* **1990**, *62*, 25−41.

(18) Rendell, A. P.; Lee, T. J.; Komornicki, A. *Chem. Phys. Lett.* **1991**, *178*, 462−470.

(19) Dunning, T. H., Jr. *J. Chem. Phys*. **1989**, *90*, 1007−1023.

(20) Rendell, A. P.; Lee, T. J.; Lindh, R. *Chem. Phys. Lett.* **1992**, *194*, 84−94.

(21) MOLPRO, a package of ab initio programs designed by H.-J. Werner and P. J. Knowles. Amos, R. D.; Bernhardsson, A.; Berning, A.; Celani, P.; Cooper, D. L.; Deegan, M. J. O.; Dobbyn, A. J.; Eckert, F.; Hampel, C.; Hetzer, G.; Knowles, P. J.; Korona, T.; Lindh, R.; Lloyd, A.W.; McNicholas, S. J.; Manby, F. R.; Meyer, W.; Mura, M. E.; Nicklass, A.; Palmieri, P.; Pitzer, R.; Rauhut, G.; Schütz, M.; Schumann, U.; Stoll, H.; Stone, A. J.; Tarroni, R.; Thorsteinsson, T.; Werner, H.-J. *Version 2002.1*.

(22) Kendall, R. A.; Apra, E.; Bernholdt, D. E.; Bylaska, E. J.; Dupuis, M.; Fann, G. I.; Harrison, R. J.; Ju, J.; Nichols, J. A.; Nieplocha, J.; Straatsma, T. P.; Windus, T. L.; Wong, A. T. *Comput. Phys. Comm.* **2000**, *128*, 260−283.

(23) *PQS version 3.2*; Parallel Quantum Solutions, 2013 Green Acres Road, Fayetteville, AR 72703; 2005. See: www.pqs-chem.com.

(24) Kobayashi, R.; Rendell, A. P. *Chem. Phys. Lett.* **1997**, *265*, 1−11.

(25) http://www.msg.ameslab.gov/GAMESS/changes.html (accessed September 2006).

(26) Szalay, P.; Gauss, J. private communication, 2006.

(27) Stanton, J. F.; Gauss, J.; Watts, J. D.; Szalay, P. G.; Bartlett, R. J. with contributions from Auer, A. A.; Bernholdt, D. B.; Christiansen, O.; Harding, M. E.; Heckert, M.; Heun, O.; Huber, C.; Jonsson, D.; Jusélius, J.; Lauderdale, W. J.; Metzroth, T.; Ruud, K. and the integral packages: MOLECULE (J. Almlöf and P. R. Taylor), PROPS (P. R. Taylor), and ABACUS (T. Helgaker, H. J. Aa. Jensen, P. Jørgensen, and J. Olsen).

(28) Nieplocha, J.; Harrison, R. J.; Littlefield, R. *Proc. Supercomputing 1994*; IEEE Computer Society Press: Washington, D.C., 1994; pp 340−346.

(29) Nieplocha, J.; Palmer, B.; Tipparaju, V.; Manojkumar, K.; Trease, H.; Apra, E. *Int. J. High Perform, Comput. Appl.* **2006**, *20*, 203−231.

(30) Ford, A. R.; Janowski, T.; Pulay, P. *J. Comput. Chem.* **2007**, *28*, xxxx−xxxx.

(31) Meyer, W. *J. Chem. Phys.* **1973**, *58*, 1017−1035.

(32) Curtiss, L. A.; Raghavachari, K.; Redfern, P. C.; Pople, J. A. *J. Chem. Phys.* **1997**, *106*, 1063−1079.

(33) Pulay, P.; Saebo, S.; Meyer, W. *J. Chem. Phys.* **1984**, *81*, 1901−1905.

(34) Matsen, F. A. *Int. J. Quantum Chem. Symp.* **1981**, *15*, 163−175.

(35) Meyer, W. *J. Chem. Phys.* **1976**, *64*, 2901−2907.

(36) Hampel, C.; Peterson, K. A.; Werner, H.-J. *Chem. Phys. Lett.* **1992**, *190*, 1−12.

(37) Scuseria, G. E.; Janssen, C. L.; Schaefer, H. F., III *J. Chem. Phys.* **1988**, *89*, 7382−7387.

(38) Schütz, M.; Werner, H.-J. *J. Chem. Phys.* **2001**, *114*, 661−681.

(39) Dykstra, Schaefer, H. F., III; Meyer, W. *J. Chem. Phys.* **1976**, *65*, 2740−2750.

(40) Schütz, M.; Lindh, R.; Werner, H.-J. *Mol. Phys.* **1999**, *96*, 719−733.

(41) Pulay, P.; Saebo, S.; Wolinski, K. *Chem. Phys. Lett.* **2001**, *344*, 543−552.

(42) Baker, J.; Pulay, P. *J. Comput. Chem.* **2002**, *23*, 1150−1156.

(43) Pulay, P.; Saebo, S. *Theor. Chim. Acta* **1986**, *69*, 357−368.

(44) Almlöf, J. *Chem. Phys. Lett.* **1991**, *176*, 319−320.

(45) Goto, K.; van de Geijn, R. *ACM Trans. Math. Software* Submitted for publication.

(46) Jensen, F. *J. Chem. Phys.* **2002**, *116*, 7372−7379.

(47) Kendall, R. A.; Dunning, T. H., Jr.; Harrison, R. J. *J. Chem. Phys.* **1992**, *96*, 6796−6806.

(48) Hobza, P.; Selzle, H. L.; Schlag, E. W. *J. Am. Chem. Soc.* **1994**, *116*, 3500−3506.

(49) Jaffe, R. L.; Smith, G. D. *J. Chem. Phys.* **1996**, *105*, 2780−2788.

(50) Tsuzuki, S.; Uchimaru, T.; Matsumura, K.; Mikami, M.; Tanabe, K. *Chem. Phys. Lett.* **2000**, *319*, 547−554.

(51) Sinnocrot, M. O.; Valeev, E. F.; Sherrill, C. D. *J. Am. Chem. Soc.* **2002**, *124*, 10887−10207.

(52) Sinnokrot, M. O.; Sherrill, C. D. *J. Phys. Chem. A* **2004**, *108*, 10200−10207.

(53) Hill, J. G.; Platts, J. A.; Werner, H.-J. *Phys. Chem. Chem. Phys.* **2006**, *8*, 4072−4078.

CT700048U