# Parallel Unrestricted MP2 Analytic Gradients Using the Distributed Data Interface†

**Christine M. Aikens and Mark S. Gordon***

*Department of Chemistry, Iowa State University, Ames, Iowa 50011*

*Received: October 8, 2003; In Final Form: December 23, 2003*

A scalable distributed-data parallel analytic gradient algorithm for unrestricted second-order Møller−Plesset perturbation theory is presented. Features of the implementation using the Distributed Data Interface are discussed in detail. Benchmark timing calculations on a parallel cluster system are presented for a variety of gold cluster molecules. The speedups, parallel efficiencies, and percentage parallelism for these calculations are reported.

## I. Introduction

Professor Schaefer has made many contributions in the field of electronic structure theory, including early efforts at parallelization.[1] We are pleased to be able to present this contribution in honor of his 60th birthday.

Second-order Møller−Plesset (MP2) perturbation theory[2] is a correlated electronic structure method that is widely used to calculate molecular energies and geometries since it provides a balance between the amount of electron correlation recovered and the computational cost of the calculation. The first derivation of the MP2 gradient was presented in 1979 by Pople et al.[3] Subsequent formulations by Handy et al.[4] and Pulay and Saebø[5] eliminated the storage of derivative integrals and three virtual molecular orbital integrals, used the Z-vector method of Handy and Schaefer[6] to reduce the number of unknown response vectors in the coupled perturbed Hartree−Fock (CPHF) equations from $3N$ (where $N$ is the number of nuclei) to 1, and eliminated the need to solve the CPHF equations in the occupied−occupied and virtual−virtual blocks. Frozen orbitals were introduced to reduce the computational requirements of the calculations, and an explicit derivation of the MP2 gradient using frozen orbitals was presented by Lee et al.[7] The implementation of the first direct MP2 gradient algorithm,[8] in which the two-electron repulsion integrals are recomputed as needed instead of stored on disk or in memory, allowed much larger calculations than were previously possible, although this came at a greater computational cost. Semidirect algorithms were also developed in order to minimize computation depending on available memory and disk space.[9] Recently, Head-Gordon developed an improved semidirect MP2 gradient algorithm, which reduces the required memory and disk storage to that of a semidirect MP2 energy calculation, eliminates disk-based sorting and transposition steps, and accounts for frozen orbitals.[10]

For the MP2 methods, the time requirements for a calculation grow as $O(n^5)$ with the number of basis functions $n$, while the memory requirements can increase by as much as $O(n^3)$, depending on the algorithm. As these methods are applied to increasingly larger molecules with reliable basis sets, the requirements of the calculation quickly outgrow the capabilities of a single-processor computer. Replicated data parallel schemes are relatively easy to implement and reduce the time requirements of a calculation, but the size of systems that can be treated with them is limited by the memory on a single node. Distributed data implementations are preferred, as these reduce both the single-node memory requirement and the wall clock time for a calculation.

**TABLE 1: Distributed Memory Operations**

| | |
|---|---|
| DDI_SEND | synchronous send |
| DDI_RECV | synchronous receive |
| DDI_GET | get block of distributed array |
| DDI_PUT | put block of distributed array |
| DDI_ACC | accumulate data into block |
| DDI_BCAST | broadcast data to all nodes |
| DDI_GSUMF | floating point global sum |
| DDI_GSUMI | integer global sum |
| DDI_CREATE | create distributed array |
| DDI_DESTROY | destroy distributed array |
| DDI_DISTRIB | obtain distributed matrix distribution |

The computer architecture must also be considered when designing an algorithm. Currently, clusters of workstations or PCs are an attractive alternative to large massively parallel processor (MPP) platforms due to a good performance/price ratio. Individual research groups and departments do not normally have enough resources to purchase large computer systems. Cluster computing also has the advantages that commodity parts can be used, installation is relatively easy, scalability can be attained in principle, and the computational resources are controlled locally. However, cluster computing also has some disadvantages that must be addressed. In particular, the slow-speed interconnection between nodes in clusters can be a source of poor performance and scalability of algorithms designed for MPP systems. In addition, not all parallel programming tools are available or optimized for cluster computing.

Fletcher et al. developed the Distributed Data Interface (DDI)[11,12] to provide a set of tools for distributed memory programming that is useful on both large parallel machines and clusters. It allows storage of large data arrays over the aggregate memory of distributed memory architectures. Like the global array (GA) tools,[13] DDI represents a global memory access model that is based on a data-passing model. It includes a set of basic network communication functions (Table 1) such as point-to-point operations such as send and receive and collective operations such as broadcast and global sum. It also includes

† Part of the special issue "Fritz Schaefer Festschrift".
* Author to whom correspondence may be addressed. E-mail: mark@si.fi.ameslab.gov.

features for using distributed and globally addressed arrays and one-sided communication. These functions are implemented using TCP/IP socket code or MPI-1 or SHMEM libraries, so DDI is available for a variety of architectures. The advantages of one-sided communication include reduced programming difficulty and increased efficiency due to asynchronous communication. The presence of remote memory adds a new level to the memory access time hierarchy: registers < cache(s) < main memory < remote memory < disk.

Most systems have significantly slower access time to remote memory than to local memory, so data locality must be considered in the design of an algorithm. Information on data locality is also available within DDI.

To date, DDI has been used to implement distributed storage versions of SCF energies,[14] analytic SCF Hessians,[15] multireference self-consistent field energies and gradients,[16] closed-shell MP2 energies and gradients,[11] Z-averaged perturbation theory energies,[17] multireference perturbation theory energies,[18] singles configuration interaction (CI) energies and gradients,[19] and full CI energies.[20] Other distributed data tools have been used to generate distributed memory versions of SCF energies,[21−27] analytic SCF Hessians,[28] CPHF equations,[29] perturbation theory energies[30−36] and gradients,[37,38] CI energies,[39,40] and coupled cluster energies.[41,42]

In this paper, the parallel implementation of unrestricted MP2 (UMP2) energies and gradients into the quantum chemistry package GAMESS[17] through the use of DDI is discussed. An assessment of the speedup and parallel efficiency is reported for a series of gold clusters using a cluster architecture.

## II. Parallel UMP2 Gradient Algorithm

A distributed data UMP2 gradient algorithm is discussed in this section. The details of the gradient derivation and serial implementation may be found elsewhere.[43] In general, the analytic derivative of the UMP2 electronic energy may be expressed as

$$E^x = \sum_{\mu\nu}^{\text{AO}} P_{\mu\nu}^{\text{MP2}} H_{\mu\nu}^x + \sum_{\mu\nu}^{\text{AO}} W_{\mu\nu}^{\text{MP2}} S_{\mu\nu}^x + \sum_{\mu\nu\lambda\sigma}^{\text{AO}} \Gamma_{\mu\nu\lambda\sigma}^{\text{MP2}} (\mu\nu|\lambda\sigma)^x \quad (1)$$

where $H_{\mu\nu}^x$ are the core Hamiltonian derivative integrals, $S_{\mu\nu}^x$ are the overlap derivative integrals, $(\mu\nu|\lambda\sigma)^x$ are the derivatives of the electron repulsion integrals (ERIs), and $\mu$, $\nu$, $\lambda$, and $\sigma$ index atomic orbitals. The MP2 density matrix $P_{\mu\nu}^{\text{MP2}}$, "energy-weighted" density matrix $W_{\mu\nu}^{\text{MP2}}$, and two-particle density matrix $\Gamma_{\mu\nu\lambda\sigma}^{\text{MP2}}$ in eq 1 are sums of their corresponding SCF and second-order correction terms

$$P_{\mu\nu}^{\text{MP2}} = P_{\mu\nu}^{\alpha\text{SCF}} + P_{\mu\nu}^{\beta\text{SCF}} + P_{\mu\nu}^{(2)}(\alpha\alpha) + P_{\mu\nu}^{(2)}(\beta\beta) \quad (2)$$

$$W_{\mu\nu}^{\text{MP2}} = W_{\mu\nu}^{\alpha\text{SCF}} + W_{\mu\nu}^{\beta\text{SCF}} + W_{\mu\nu}^{(2)}(\alpha\alpha) + W_{\mu\nu}^{(2)}(\beta\beta) \quad (3)$$

$$\Gamma_{\mu\nu\lambda\sigma}^{\text{MP2}} = \Gamma_{\mu\nu\lambda\sigma}^{\text{SCF}} + \Gamma_{\mu\nu\lambda\sigma}^{(2)} \quad (4)$$

The second-order correction to the two-particle density matrix is normally divided into the sum of so-called separable ($\Gamma^{\text{S}}$) and nonseparable ($\Gamma^{\text{NS}}$) terms

$$\Gamma_{\mu\nu\lambda\sigma}^{(2)} = \Gamma_{\mu\nu\lambda\sigma}^{\text{NS}} + \Gamma_{\mu\nu\lambda\sigma}^{\text{S}} \quad (5)$$

In practice, eq 1 is usually evaluated by forming the density matrices $P^{(2)}$, $W^{(2)}$, and $\Gamma^{(2)}$ in the MO basis and back transforming these matrices to the AO basis. The expressions for these matrix elements are shown in the appendix.

**A. Implementation Considerations.** In the MO basis, the expressions for the density matrix elements require 18 different types of integrals, which may be divided into five classes. Using "v" to denote a virtual MO and "o" to denote an occupied MO, these integral types may be expressed as:

1. $(o^\alpha o^\alpha|o^\alpha o^\alpha)$ $(o^\alpha o^\alpha|o^\beta o^\beta)$ $(o^\beta o^\beta|o^\beta o^\beta)$
2. $(v^\alpha o^\alpha|o^\alpha o^\alpha)$ $(v^\alpha o^\alpha|o^\beta o^\beta)$ $(v^\beta o^\beta|o^\alpha o^\alpha)$ $(v^\beta o^\beta|o^\beta o^\beta)$
3. $(v^\alpha v^\alpha|o^\alpha o^\alpha)$ $(v^\alpha v^\alpha|o^\beta o^\beta)$ $(v^\beta v^\beta|o^\alpha o^\alpha)$ $(v^\beta v^\beta|o^\beta o^\beta)$
4. $(v^\alpha o^\alpha|v^\alpha o^\alpha)$ $(v^\alpha o^\alpha|v^\beta o^\beta)$ $(v^\beta o^\beta|v^\beta o^\beta)$
5. $(v^\alpha v^\alpha|v^\alpha o^\alpha)$ $(v^\alpha v^\alpha|v^\beta o^\beta)$ $(v^\beta v^\beta|v^\alpha o^\alpha)$ $(v^\beta v^\beta|v^\beta o^\beta)$

Parallel processor systems with global memories in the terabyte range are now common, so it is feasible that the smaller integral classes may be stored in memory even for larger problems. Thus, global memory on the order of $n_o^2n^2$ is assumed, where $n_o$ is the number of occupied orbitals and $n$ is the number of basis functions and it is assumed that $n_o$ is much smaller than $n$. A different algorithm setup that trades additional computation for reduced memory requirements could also be envisioned but will not be discussed here. The first four integral classes fall within the assumption of $O(n_o^2n^2)$ global memory, so these integrals may be stored in a distributed fashion across the nodes. The fifth class is closer in size to $O(n_on^3)$. However, this class is only used in the construction of the Lagrangian, so the terms involving these integrals will be calculated separately and in a direct fashion to avoid storage of this larger integral class. In addition, since $n^4$ memory is not assumed, the AO integrals will be calculated in a direct fashion. The density matrices, MO coefficients, orbital energies, and other data of order $n^2$ or less are replicated across the nodes.

The implementation of the following parallel UMP2 gradient scheme will be discussed in more detail in the subsequent sections:

1. Parallel direct transformation to yield MO integrals that contain at most two virtual orbital indices (stored in global memory).

2. Evaluation of UMP2 energy and construction of immediately obtainable terms of $P^{(2)}$, $W^{(2)}$, and the Lagrangian $L$.

3. Computation of the Lagrangian terms involving (vv|vo) class integrals directly from AO integrals.

4. Solution of the Z-vector equations[6] in the MO basis.

5. Completion of one-particle density matrices and the one-particle contribution to the gradient.

6. Back transformation of integrals to generate $\Gamma^{\text{NS}}$ and completion of the two-particle contribution to the gradient.

**B. Integral Transformation and Storage.** First, the 14 distributed arrays that hold the first four classes of transformed integrals must be created. These are depicted in Figure 1. Triangular pairings of two occupied or virtual indices are used wherever possible to reduce the number of integrals stored. Integral type $(v^\alpha o^\alpha|v^\beta o^\beta)$ has no available triangular pairings, so the full range of each index must be used for this array. Overall, the limiting storage requirement for the seven two virtual integral types (and thus for the algorithm) is $([n_0^\alpha]^2 + [n_0^\beta]^2 + n_0^\alpha n_0^\beta)n^2$. The "supermatrix" symmetry of integral types $(o^\alpha o^\alpha|o^\alpha o^\alpha)$ and $(o^\beta o^\beta|o^\beta o^\beta)$ is not used so that the arrays have a regular rectangular structure. The excess storage cost for this is not large since this is the smallest integral class. For all integral types, all pairs of occupied indices are divided among the available nodes, with the full range of the other indices located on a given node. The integrals are distributed in this manner so that the long summations over virtual indices will be local.

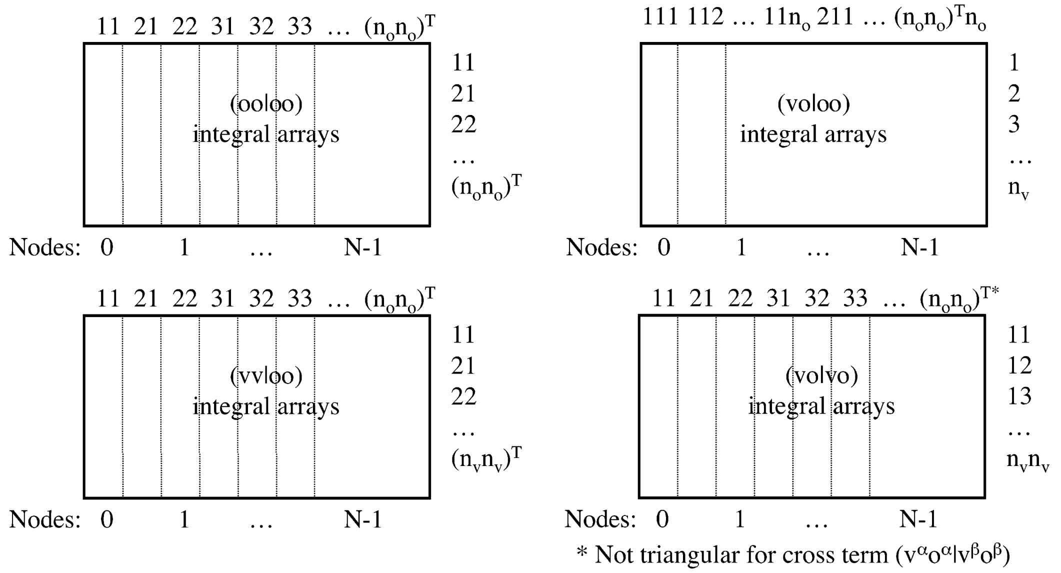


**Figure 1.** Indices used for integral arrays. Triangular are indices used wherever possible to reduce amount of memory required.

Next, the 14 integral types that correspond to the first four classes must be generated and stored in the distributed arrays. The general parallel transformation 4-fold loop procedure has been described previously,[11] so the details will not be presented here. The appropriate alpha and beta MO coefficients are used in the transformation of each index to generate the required integral types. The replicated data requirement for this procedure is $l^2n(n_o{}^\alpha + n_o{}^\beta)$, where $l$ is the length of a shell (1 for s, 3 for p, 6 for Cartesian d, etc.). After the integrals are generated, they are stored in the appropriate distributed arrays through the use of the DDI_PUT operation.

**C. Density Matrix Creation.** Most of the terms in the one-particle density matrices and the Lagrangian (eqs A1−A6 and A13) may be evaluated from the integrals discussed in section B. The amplitudes for the virtual−virtual blocks of $P^{(2)}$ (eq A3) and $W^{(2)}$ (terms 1 and 2 of eq A5) are calculated from (vo|vo) integrals on the local node and multiplied by (vo|vo) integrals that are also locally held. Thus, these blocks do not require remote memory access. The occupied−occupied blocks of $P^{(2)}$ (eqs A1 and A2) and $W^{(2)}$ (terms 1 and 2 of eq A4) require remote DDI_GET operations to obtain a set of (vo|vo) integrals for a given vv pair. Then, the amplitude generation and integral multiplication steps may proceed locally. A global sum of $P^{(2)}$ is performed so that each node holds the complete occupied-occupied and virtual−virtual blocks.

The fourth and fifth terms of eq A4 use the (oo|oo) integrals when $p$ and $q$ are both occupied indices. Since this is the only matrix block that requires these integrals, the memory for these is released after the terms are computed. The (vv|oo) class of integrals contributes to terms 4 and 5 of eq A4 when $p$ and $q$ are both virtual indices. This class is also used to compute the orbital Hessian in eqs A7 and A8, so it is retained in memory. The (vo|oo) class of integrals is used in the calculation of terms 1 and 2 of eq A6 and terms 1, 2, 5, and 6 of eq A13. It will also contribute to the fourth and fifth terms of eq A4 after the virtual−occupied block of $P^{(2)}$ is determined, so these integrals are retained in memory.

**D. Three-Virtual Terms for Lagrangian.** Terms 3, 4, 7, and 8 of eq A13 require the (vv|vo) integrals. These terms in the Lagrangian may be rearranged to form terms in a mixed MO/AO basis as discussed previously.[11,37,38] For the UMP2 gradient, modified integrals instead of amplitudes will be transformed. For example, term 8 of eq A13 may be rearranged as

$$L^8_{a^\alpha i^\alpha} = \sum_{j^\beta}^{\mathrm{act}^\beta}\sum_{b^\alpha}^{\mathrm{virt}^\alpha}\sum_{c^\beta}^{\mathrm{virt}^\beta}\frac{(i^\alpha b^\alpha|j^\beta c^\beta)}{D^{b^\alpha c^\beta}_{i^\alpha j^\beta}}(a^\alpha b^\alpha|j^\beta c^\beta) = \sum_{\nu\lambda\sigma}\sum_{j^\beta}^{\mathrm{act}^\beta}\sum_{b^\alpha}^{\mathrm{virt}^\alpha}\sum_{c^\beta}^{\mathrm{virt}^\beta} C^\alpha_{\lambda a}C^\alpha_{\sigma b}\,C^\beta_{\nu c}(j^\beta\nu|\lambda\sigma)\,\frac{(i^\alpha b^\alpha|j^\beta c^\beta)}{D^{b^\alpha c^\beta}_{i^\alpha j^\beta}} \quad (6)$$

where $i$ and $j$ index occupied orbitals, a, b, and c index virtual orbitals, and $\lambda$, $\nu$, and $\sigma$ index atomic orbitals. A mixed MO/AO Lagrangian is introduced on the left-hand side of eq 6 as

$$L^8_{a^\alpha i^\alpha} = \sum_\lambda C^\alpha_{\lambda a}\, L^8_{\lambda i^\alpha} \quad (7)$$

and further rearrangement of eq 6 yields

$$\sum_\lambda C^\alpha_{\lambda a}\, L^8_{\lambda i^\alpha} = \sum_{\nu\lambda\sigma}\sum_{j^\beta}^{\mathrm{act}^\beta} C^\alpha_{\lambda a}(j^\beta\nu|\lambda\sigma)\left(\sum_{b^\alpha}^{\mathrm{virt}^\alpha}\sum_{c^\beta}^{\mathrm{virt}^\beta} C^\alpha_{\sigma b}\, C^\beta_{\nu c}\frac{(i^\alpha b^\alpha|j^\beta c^\beta)}{D^{b^\alpha c^\beta}_{i^\alpha j^\beta}}\right) \quad (8)$$

Thus

$$L^8_{\lambda i^\alpha} = \sum_{\nu\sigma}\sum_{j^\beta}^{\mathrm{act}^\beta}(j^\beta\nu|\lambda\sigma)I^{\sigma\nu}_{i^\alpha j^\beta} \quad (9)$$

where the half-transformed integral is

$$I^{\sigma\nu}_{i^\alpha j^\beta} = \sum_{b^\alpha}^{\mathrm{virt}^\alpha}\sum_{c^\beta}^{\mathrm{virt}^\beta} C^\alpha_{\sigma b}\, C^\beta_{\nu c}\frac{(i^\alpha b^\alpha|j^\beta c^\beta)}{D^{b^\alpha c^\beta}_{i^\alpha j^\beta}} \quad (10)$$

Term 7 of eq A13 can be represented as a difference of half-transformed integrals

$$L^7_{\lambda i^\alpha} = \sum_{\nu\sigma}\sum_{j^\alpha}^{\mathrm{act}^\alpha}(j^\alpha\nu|\lambda\sigma)(I^{\sigma\nu}_{i^\alpha j^\alpha} - I^{\nu\sigma}_{i^\alpha j^\alpha}) \quad (11)$$

The half-transformed integrals may be generated from locally

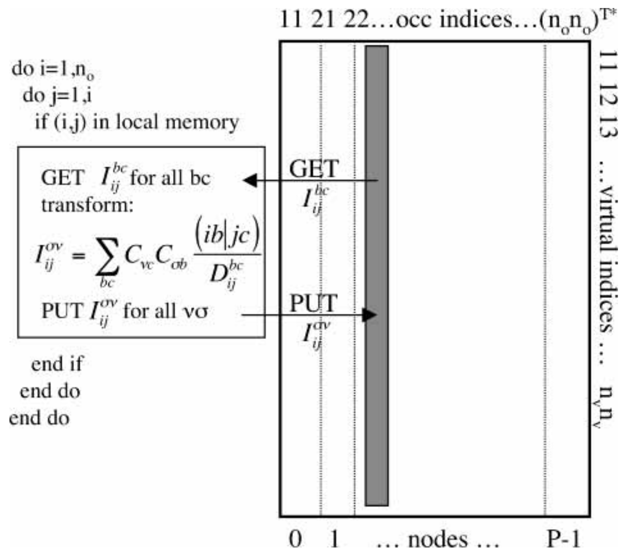

11 21 22…occ indices…(n_o n_o)^T*
do i=1,n_o
do j=1,i
if (i,j) in local memory
GET $I_{ij}^{bc}$ for all bc
transform:
$I_{ij}^{\sigma\nu} = \sum_{bc} C_{\nu c} C_{\sigma b} \frac{(ib|jc)}{D_{ij}^{bc}}$
PUT $I_{ij}^{\sigma\nu}$ for all νσ
end if
end do
end do
GET $I_{ij}^{bc}$
PUT $I_{ij}^{\sigma\nu}$
11 12 13 …virtual indices … n_v n_v
0 1 … nodes … P-1

**Figure 2.** First half back transformation of the (vo|vo) integrals. This procedure is used in the creation of (vv|vo) integrals and the nonseparable two-particle density matrix. No communication is required in this step.

held (vo|vo) integrals without communication since all gets and puts are done locally (Figure 2). To save memory at the cost of extra computation later, the half-transformed integrals replace the (vo|vo) integrals in memory.

In a similar manner, terms 3 and 4 of eq A13 may be rearranged to the form

$$L_{\nu i^\alpha}^3 = \sum_{\lambda\sigma} (i^\alpha \nu|\lambda\sigma) P_{\lambda\sigma}^{(2)}(\alpha\alpha) \tag{12}$$

$$L_{\nu i^\alpha}^4 = \sum_{\lambda\sigma} (i^\alpha \nu|\lambda\sigma) P_{\lambda\sigma}^{(2)}(\beta\beta) \tag{13}$$

where the virtual−virtual density matrix has been back transformed to the AO basis. Since the required storage for the AO density matrix is O($n^2$) and the time required for the back transformation is trivial, this back transformation is not distributed.

Representation of these Lagrangian terms in the mixed MO/AO basis has the benefit that the quarter-transformed ERIs with one MO and three AO indices may be used as soon as they are generated. A second 4-fold loop over shells is used to calculate and quarter transform the ERIs as described previously.[11] After the inner two loops, a DDI_GET operation must be performed to read in half-transformed integrals such as $I_{i\alpha j\beta}^{\sigma\nu}$ for a given $\sigma,\nu$ pair. This requires communication among the processors. Once the quarter-transformed ERIs are contracted with the respective half-transformed integrals or AO density matrix elements and the mixed MO/AO Lagrangian terms are formed, these terms are transformed to the MO basis and combined with the other four terms of the Lagrangian. The Lagrangian is summed globally so each node holds the full matrix.

Once the Lagrangian has been calculated, the (vo|vo) integrals must be restored. The half-transformed integrals are transformed with the inverse MO coefficient matrix $\mathbf{C}^{-1} = \mathbf{C}^{\mathrm{T}}\mathbf{S}$, where $\mathbf{S}$ is the overlap matrix over AOs, which restores the virtual MO indices, and are then multiplied by the appropriate orbital energy factor (such as $D_{i\alpha j\beta}^{b^\alpha c^\beta}$) to recreate the original (vo|vo) integrals. Since the entire set of indices for a given occupied−occupied pair is on the local processor, this step requires no communication.

**E. Solution of Z-Vector Equations.** A reduced Lagrangian and α and $\beta$ trial vectors containing only symmetry-allowed elements are used in the solution of the Z-vector equations. In each step of the iterations, blocks of the orbital Hessian are constructed from (vo|vo) and (vv|oo) integrals. If the integrals for a given pair of occupied indices are held locally, the (vo|vo) and (vv|oo) integrals for the α, $\beta$, and mixed-spin terms are used to form blocks of the orbital Hessian in eqs A7 and A8 (Figure 3). These blocks are multiplied by appropriate sections of the trial vectors, and the products are globally summed to

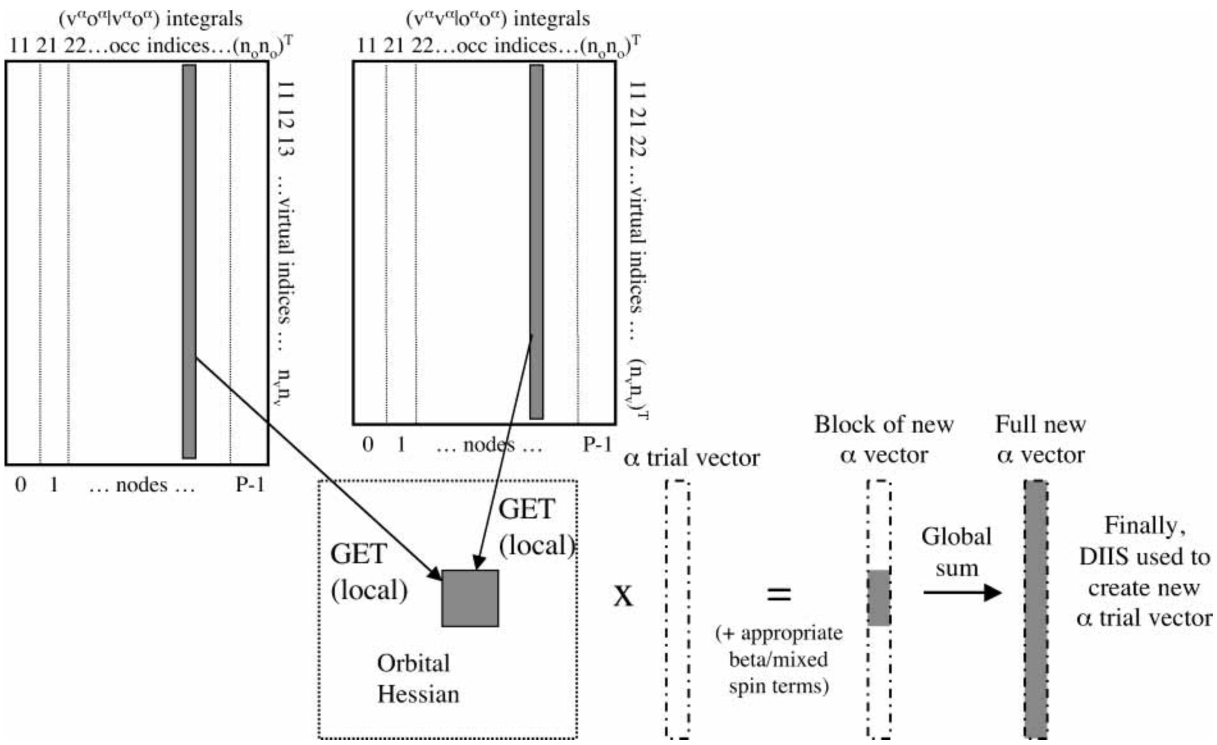
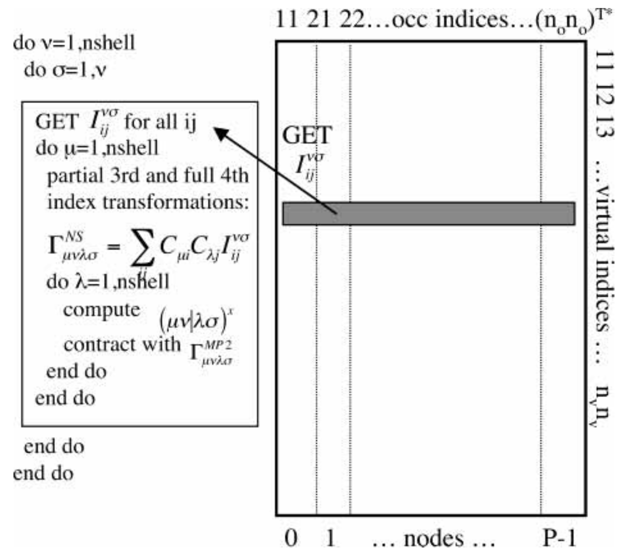

($v^\alpha o^\alpha|v^\alpha o^\alpha$) integrals
11 21 22…occ indices…(n_o n_o)^T
11 12 13 …virtual indices … n_v n_v
0 1 … nodes … P-1
($v^\alpha v^\alpha|o^\alpha o^\alpha$) integrals
11 21 22…occ indices…(n_o n_o)^T
11 21 22 …virtual indices … (n_v n_v)^T
0 1 … nodes … P-1
GET (local)
GET (local)
Orbital Hessian
X
α trial vector
=
(+ appropriate beta/mixed spin terms)
Block of new α vector
Global sum
Full new α vector
Finally, DIIS used to create new α trial vector

**Figure 3.** Solution of the Z-vector equations. Locally held integrals are used to construct a block of the orbital Hessian. Storage of the full orbital Hessian is avoided. Interprocessor communication is needed only for the global sum.



```
do ν=1,nshell
  do σ=1,ν
    GET $I_{ij}^{\nu\sigma}$ for all ij
    do μ=1,nshell
      partial 3rd and full 4th
      index transformations:
      $\Gamma_{\mu\nu\lambda\sigma}^{NS} = \sum_{ij} C_{\mu i} C_{\lambda j} I_{ij}^{\nu\sigma}$
      do λ=1,nshell
        compute $(\mu\nu|\lambda\sigma)^x$
        contract with $\Gamma_{\mu\nu\lambda\sigma}^{MP2}$
      end do
    end do
  end do
end do
```

11 21 22…occ indices…$(n_o n_o)^{T*}$
GET $I_{ij}^{\nu\sigma}$
11 12 13 …virtual indices … $n_v n_v$
0 1 … nodes … P-1

**Figure 4.** Second half back transformation of the (vo|vo) integrals. This procedure is used in the creation of the nonseparable two-particle density matrix. Communication across the processors is required to retrieve the half-back-transformed integrals.

generate the $\alpha$ and $\beta$ product vectors. In this way, storage of the full orbital Hessian is avoided. Alpha and beta error vectors are calculated. A direct inversion in the iterative subspace (DIIS)[44,45] interpolation method is used to determine the trial vectors for the next iteration. Since previous trial and error vectors are needed for DIIS, these are stored in a distributed manner across the nodes. At convergence, the trial vectors are the virtual−occupied block of the $\alpha$ and $\beta$ one-particle density matrices. The (vv|oo) integrals are not needed further, so the memory for these is released.

**F. Completion of One-Particle Gradient.** After solution of the Z-vector equations, the one-particle density matrix is complete. The third term in the virtual−occupied block of the one-particle energy-weighted density matrix (eq A6) may now be calculated. In addition, the final contribution to the fourth and fifth terms of eq A4 can be evaluated using the (vo|oo) integrals. This contribution may be computed in a distributed fashion and globally summed to complete the energy-weighted density matrix. This concludes the use of the (vo|oo) integrals, so memory for these is released. Finally, the completed $\alpha$ and $\beta$ one-particle density matrices and energy-weighted density matrices are back transformed to the AO basis, added to their SCF counterparts (eqs 2 and 3), and contracted with the core Hamiltonian and overlap derivatives (eq 1) to yield the one-electron portion of the gradient

**G. Two-Particle Gradient.** The four-index back transformation of the amplitudes in the nonseparable two-particle density (eq A14) is similar in essence to the procedure described in section D. The process begins with the half back transformation described earlier to generate half-transformed integrals (Figure 2). A third 4-fold loop algorithm is required to generate the derivative ERIs. Inside the outer two loops, a DDI_GET operation necessitating communication between the processors is required to read in the half-transformed integrals with a given $\sigma,\nu$ pair (Figure 4). These half-transformed integrals are used to form the half-transformed amplitudes. Inside the third loop over shells, the half-transformed amplitudes are further back transformed to all AO indices in one range and to AO indices over the third-loop shell in the other range. This step is done locally to avoid further communication. However, since the nonseparable density is not symmetrized, the derivative ERIs must be calculated four times more than the minimal list. After the nonseparable density is calculated, it is combined with the separable density (eq A15), added to its SCF analog, and contracted with the derivative ERI's.

## III. Timings/Benchmarks

The parallel UMP2 gradient code described above has been implemented in GAMESS and has been benchmarked on a cluster system. The cluster under consideration is comprised of IBM p640 nodes connected by dual Gigabit Ethernet. Each p640 node has 4 Power3-II processors running at 375 MHz and contains 16 GB of memory. On 8 nodes, global memory of 128 GB is available.

The smallest molecule under consideration in the benchmarks is $Au_3H_4$. Two basis sets have been used with this molecule. The smaller of the two consists of the 6-31++G**[46−48] basis set on H and the Stevens−Basch−Krauss−Jasien−Cundari (SBKJC)[49−51] effective core potential basis set with f polarization functions and one diffuse sp function on Au, for a total of 170 spherical harmonic basis functions. The larger basis set consists of the aug-cc-pVTZ[52] basis set on H and the uncontracted SBKJC basis set with 3f2g polarization functions and one sp diffuse function on Au, for a total of 380 spherical harmonic basis functions. For both basis sets, there are 31 $\alpha$-occupied orbitals and 30 $\beta$-occupied orbitals for $Au_3H_4$. A UMP2 gradient calculation with the smaller basis set requires 4 Mwords of replicated memory and 116 Mwords of distributed memory, so it fits in the memory of a single processor on the IBM cluster. The calculation using the larger basis set requires 18 Mwords of replicated memory and 647 Mwords of distributed memory. This calculation requires the memory allotted to two processors on the IBM cluster.

A larger test case is $Au_3O_4$. The basis set used with this molecule consists of the aug-cc-pVTZ[52,53] basis set on O and the uncontracted SBKJC basis set with 3f2g polarization functions and one sp diffuse function on Au, for a total of 472 spherical harmonic basis functions. For this molecule, there are 45 $\alpha$-occupied orbitals and 44 $\beta$-occupied orbitals. A UMP2 gradient calculation on this molecule requires 41 Mwords of replicated memory and 2166 Mwords of distributed memory. This calculation requires the memory allotted to six processors on the IBM cluster.

Table 2 lists the average CPU time in seconds and the associated speedups and parallel efficiencies for a UMP2 gradient calculation on 1, 2, 4, 8, 16, and 32 processors for the three gold clusters benchmarked in this study. Since the second and third test cases require more than one processor on the IBM cluster, the single-processor time required for these calculations is determined from runs on two and eight processors, respectively, assuming ideal speedup. The speedups may be visualized graphically in Figure 5. A superlinear speedup is exhibited for the smallest test case running on two processors. This may be due to the increased cache capacity available on two processors. As the number of processors increases, the observed speedup is less than the ideal speedup. This may be due to serial portions of the code, load balancing issues, or the communication and synchronization costs. On larger numbers of processors, the larger test cases exhibit greater parallel efficiencies than the smaller test cases. This is expected, since the amount of parallel computation relative to communication and other costs is higher for larger runs.

Table 3 lists an assessment of the "percentage parallelism" in each calculation. These values are determined by assuming Amdahl scaling and fitting an equation of the form $t(P) = t_s +$

**TABLE 2: Wall Clock Time (s), Speedup, and Parallel Efficiency for UMP2 Gradient Step on IBM Cluster**

| *P* (no. processors) | $Au_3H_4$, $n = 170$ | | | $Au_3H_4$, $n = 380$ | | | $Au_3O_4$, $n = 472$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | wall time | speedup | efficiency | wall time | speedup | efficiency | wall time | speedup | efficiency |
| 1 | 2867 | 1.0 | 100.0 | 166482[a] | 1.0 | 100.0 | 665392[b] | 1.0 | 100.0 |
| 2 | 1412 | 2.0 | 101.5 | 83241 | 2.0 | 100.0 | 332696[b] | 2.0 | 100.0 |
| 4 | 771 | 3.7 | 93.0 | 42555 | 3.9 | 97.8 | 166348[b] | 4.0 | 100.0 |
| 8 | 596 | 4.8 | 60.1 | 22602 | 7.4 | 92.1 | 83174 | 8.0 | 100.0 |
| 16 | 295 | 9.7 | 60.7 | 11980 | 13.9 | 86.9 | 45138 | 14.7 | 92.1 |
| 32 | 231 | 12.4 | 38.8 | 7509 | 22.2 | 69.3 | 25205 | 26.4 | 82.5 |

[a] Value calculated from run using two processors, assuming ideal speedup. [b] Values calculated from run using eight processors, assuming ideal speedup.

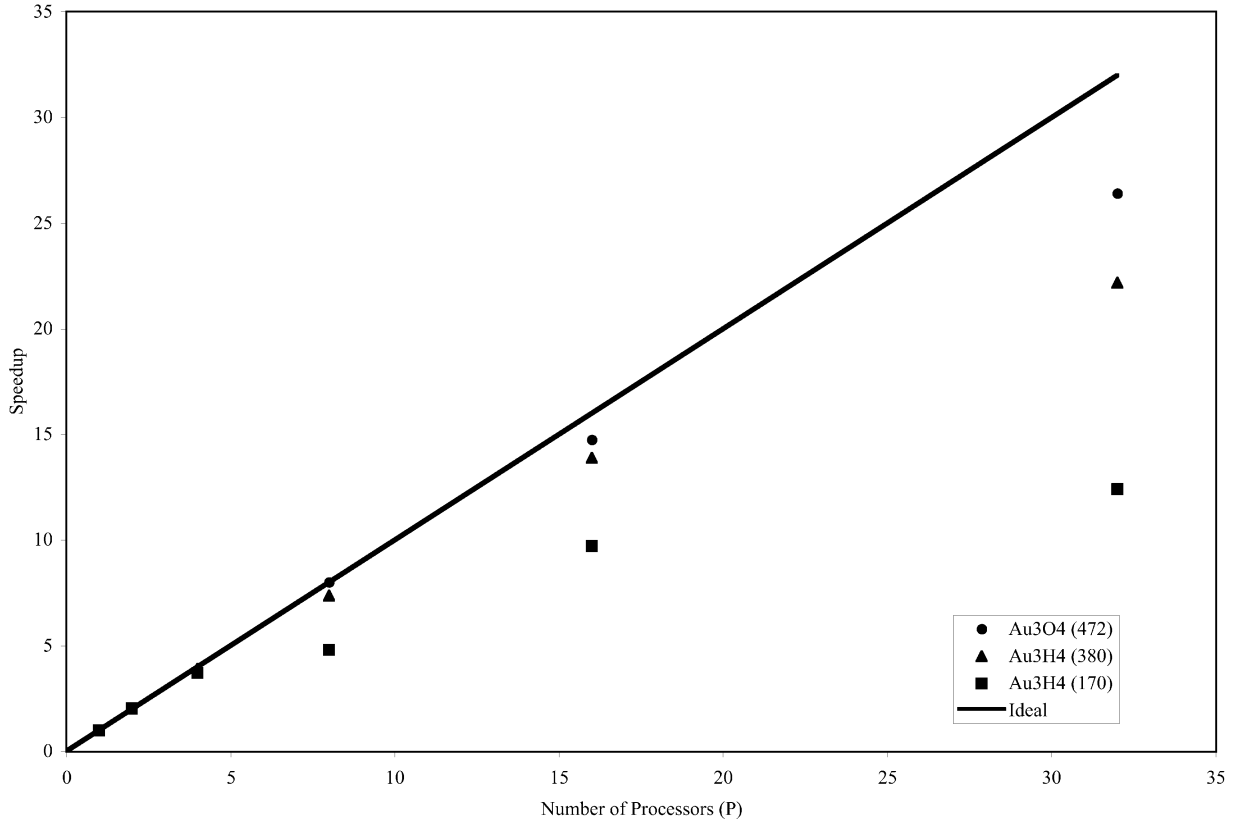


**Figure 5.** Speedup curves for three gold cluster molecules. The size of the basis set is listed in parentheses.

**TABLE 3: Serial Time, Parallel Time, and Parallel Percentage for UMP2 Gradient Calculation**

| | $Au_3H_4$ (170) | $Au_3H_4$ (380) | $Au_3O_4$ (472) |
|---|---|---|---|
| $t_s$ (s) | 150 | 3270 | 6187 |
| $t_p$ (s) | 2679 | 159396 | 616942 |
| parallel % | 93.5 | 95.7 | 92.7 |
| $R^2$ | 0.99542 | 0.99994 | 0.99986 |

$t_p/P$ to the series of wall clock times $t(P)$ on $P$ processors for a given calculation, as discussed in ref 3. This yields $t_s$, the amount of time spent on serial portions of the calculation, and $t_p$, the total amount of time spent on parallel sections. The parallel percentage represents the percentage of the calculation that is scalable. Unlike parallel efficiencies, which only compare two calculations, this model provides a way to assess the scalability over a range of processors. It does not consider effects due to load balancing or communication latencies, but these effects are usually small. Given the small number of data points in the analysis, it is not meaningful to quantitatively distinguish the three parallel % values in the table. However, the $R^2$ values suggest that the parallel percentage of the UMP2 gradient calculation is over 90%.

## IV. Conclusions

A scalable parallel gradient algorithm for UMP2 using the DDI is presented. The purpose of this algorithm is to increase the speed for a given UMP2 gradient calculation as well as to permit the evaluation of larger jobs. The transformed molecular orbital integrals with two virtual indices or fewer are distributed across the nodes. Data of the order $n^2$ or less is replicated across the nodes. The algorithm has been designed to use locally held data wherever possible, and sections where communication is required have been identified. Redundant computation of two-electron integrals and derivative integrals is used in order to reduce communication and required memory costs.

Benchmark calculations have been run on a series of three gold cluster molecules using an IBM cluster, and the calculations scale well. The percentage parallelism for the UMP2 gradient calculation is above 90%.

**Acknowledgment.** The authors would like to thank Graham Fletcher, Mike Schmidt, and Ricky Kendall for helpful discussions. C.M.A. is grateful to the National Science Foundation for providing a Graduate Research Fellowship. The authors also acknowledge the US-DOD Common High Performance Computing Software Support Initiative administered by the Air Force

Office of Scientific Research. The authors would also like to thank IBM for a Shared University Research award, which provided a 128-processor cluster on which the benchmark calculations were run.

## Appendix. Density Matrix Expressions

In general, indices *p*, *q*, *r*, and *s* represent any molecular orbital. Occupied orbitals are indexed by *i*, *j*, and *k*; virtual orbitals are indexed by *a*, *b*, and *c*; core orbitals are indexed by *K*. The one-particle α−α density matrix expressions for the core−active, active−active, and virtual−virtual blocks of $P^{(2)}$ in the MO basis are

$$P^{(2)}_{K^\alpha i^\alpha} = P^{(2)}_{i^\alpha K^\alpha} = \frac{1}{(\epsilon_i{}^\alpha - \epsilon_K{}^\alpha)} \left[ \sum_{j^\alpha}^{\text{act}^\alpha} \sum_{a^\alpha b^\alpha}^{\text{virt}^\alpha} \frac{[(i^\alpha a^\alpha|j^\alpha b^\alpha) - (i^\alpha b^\alpha|j^\alpha a^\alpha)]}{D^{a^\alpha b^\alpha}_{i^\alpha j^\alpha}} (K^\alpha a^\alpha|j^\alpha b^\alpha) + \sum_{j^\beta}^{\text{act}^\beta} \sum_{a^\alpha}^{\text{virt}^\alpha} \sum_{b^\beta}^{\text{virt}^\beta} \frac{(i^\alpha a^\alpha|j^\beta b^\beta)}{D^{a^\alpha b^\beta}_{i^\alpha j^\beta}} (K^\alpha a^\alpha|j^\beta b^\beta) \right] \quad \text{(A1)}$$

$$P^{(2)}_{i^\alpha j^\alpha} = - \sum_{k^\alpha}^{\text{act}^\alpha} \sum_{a^\alpha}^{\text{virt}^\alpha} \sum_{b^\alpha}^{\text{virt}^\alpha} \frac{[(i^\alpha a^\alpha|k^\alpha b^\alpha) - (i^\alpha b^\alpha|k^\alpha a^\alpha)]}{D^{a^\alpha b^\alpha}_{i^\alpha k^\alpha}} \frac{(j^\alpha a^\alpha|k^\alpha b^\alpha)}{D^{a^\alpha b^\alpha}_{j^\alpha k^\alpha}} - \sum_{k^\beta}^{\text{act}^\beta} \sum_{a^\alpha}^{\text{vact}^\alpha} \sum_{b^\beta}^{\text{virt}^\beta} \frac{[(i^\alpha a^\alpha|k^\beta b^\beta)]}{D^{a^\alpha b^\beta}_{i^\alpha k^\beta}} \frac{[(j^\alpha a^\alpha|k^\beta b^\beta)]}{D^{a^\alpha b^\beta}_{j^\alpha k^\beta}} \quad \text{(A2)}$$

$$P^{(2)}_{a^\alpha b^\alpha} = \sum_{i^\alpha}^{\text{act}^\alpha} \sum_{j^\alpha}^{\text{act}^\alpha} \sum_{c^\alpha}^{\text{virt}^\alpha} \frac{[(i^\alpha a^\alpha|j^\alpha c^\alpha) - (i^\alpha c^\alpha|j^\alpha a^\alpha)]}{D^{a^\alpha c^\alpha}_{i^\alpha j^\alpha}} \frac{(i^\alpha b^\alpha|j^\alpha c^\alpha)}{D^{b^\alpha c^\alpha}_{i^\alpha j^\alpha}} + \sum_{i^\alpha}^{\text{act}^\alpha} \sum_{j^\beta}^{\text{act}^\beta} \sum_{c^\beta}^{\text{virt}^\beta} \frac{(i^\alpha a^\alpha|j^\beta c^\beta)}{D^{a^\alpha c^\beta}_{i^\alpha j^\beta}} \frac{(i^\alpha b^\alpha|j^\beta c^\beta)}{D^{b^\alpha c^\beta}_{i^\alpha j^\beta}} \quad \text{(A3)}$$

The one-particle α−α energy-weighted density matrix expressions for the occupied−occupied, virtual−virtual, and virtual−occupied blocks of $W^{(2)}$ are

$$W^{(2)}_{i^\alpha j^\alpha} = -N_{i^\alpha j^\alpha} \sum_{k^\alpha}^{\text{act}^\alpha} \sum_{a^\alpha b^\alpha}^{\text{virt}^\alpha} \frac{[(i^\alpha a^\alpha|k^\alpha b^\alpha) - (i^\alpha b^\alpha|k^\alpha a^\alpha)]}{D^{a^\alpha b^\alpha}_{i^\alpha k^\alpha}} (j^\alpha a^\alpha|k^\alpha b^\alpha) - N_{i^\alpha j^\alpha} \sum_{k^\beta}^{\text{act}^\beta} \sum_{a^\alpha}^{\text{virt}^\alpha} \sum_{b^\beta}^{\text{virt}^\beta} \frac{(i^\alpha a^\alpha|k^\beta b^\beta)}{D^{a^\alpha b^\beta}_{i^\alpha k^\beta}} (j^\alpha a^\alpha|k^\beta b^\beta) - \frac{1}{2} P^{(2)}_{i^\alpha j^\alpha} (\epsilon_i{}^\alpha + \epsilon_j{}^\alpha) - \sum_{p^\alpha q^\alpha}^{\text{all}^\alpha} P^{(2)}_{p^\alpha q^\alpha} [(p^\alpha q^\alpha|i^\alpha j^\alpha) - (p^\alpha i^\alpha|q^\alpha j^\alpha)] - \sum_{p^\beta q^\beta}^{\text{all}^\beta} P^{(2)}_{p^\beta q^\beta} (p^\beta q^\beta|i^\alpha j^\alpha) \quad \text{(A4)}$$

where

$$N_{i^\alpha j^\alpha} \equiv \begin{cases} 0, \text{ for both } i,j \in \text{core} \\ 1, \text{ otherwise} \end{cases}$$

$$W^{(2)}_{a^\alpha b^\alpha} = - \sum_{i^\alpha j^\alpha}^{\text{act}^\alpha} \sum_{c^\alpha}^{\text{virt}^\alpha} \frac{[(i^\alpha a^\alpha|j^\alpha c^\alpha) - (i^\alpha c^\alpha|j^\alpha a^\alpha)]}{D^{a^\alpha c^\alpha}_{i^\alpha j^\alpha}} (i^\alpha b^\alpha|j^\alpha c^\alpha) - \sum_{i^\alpha}^{\text{act}^\alpha} \sum_{j^\beta}^{\text{act}^\beta} \sum_{c^\beta}^{\text{virt}^\beta} \frac{(i^\alpha a^\alpha|j^\beta c^\beta)}{D^{a^\alpha c^\beta}_{i^\alpha j^\beta}} (i^\alpha b^\alpha|j^\beta c^\beta) - \frac{1}{2} P^{(2)}_{a^\alpha b^\alpha} (\epsilon_a{}^\alpha + \epsilon_b{}^\alpha) \quad \text{(A5)}$$

$$W^{(2)}_{a^\alpha i^\alpha} = -2 \sum_{j^\alpha k^\alpha}^{\text{act}^\alpha} \sum_{b^\alpha}^{\text{virt}^\alpha} \frac{[(j^\alpha a^\alpha|k^\alpha b^\alpha) - (j^\alpha b^\alpha|k^\alpha a^\alpha)]}{D^{a^\alpha b^\alpha}_{j^\alpha k^\alpha}} (i^\alpha j^\alpha|b^\alpha k^\alpha) - 2 \sum_{j^\alpha}^{\text{act}^\alpha} \sum_{k^\beta}^{\text{act}^\beta} \sum_{b^\beta}^{\text{virt}^\beta} \frac{(j^\alpha a^\alpha|k^\beta b^\beta)}{D^{a^\alpha b^\beta}_{j^\alpha k^\beta}} (i^\alpha j^\alpha|b^\beta k^\beta) - P^{(2)}_{a^\alpha i^\alpha} \epsilon_i{}^\alpha \quad \text{(A6)}$$

The virtual−occupied block of $P^{(2)}$ is obtained from the iterative solution of the Z-vector equations due to Handy and Schaefer[6]

$$\sum_{b^\alpha}^{\text{virt}^\alpha} \sum_{j^\alpha}^{\text{occ}^\alpha} \{A_{a^\alpha i^\alpha b^\alpha j^\alpha} + \delta_{ab}\delta_{ij}(\epsilon_b{}^\alpha - \epsilon_j{}^\alpha)\} P^{(2)}_{b^\alpha j^\alpha} + \sum_{b^\beta}^{\text{virt}^\beta} \sum_{j^\beta}^{\text{occ}^\beta} \{A_{a^\alpha i^\alpha b^\beta j^\beta}\} P^{(2)}_{b^\beta j^\beta} = -L_{a^\alpha i^\alpha} \quad \text{(A7)}$$

$$\sum_{b^\beta}^{\text{virt}^\beta} \sum_{j^\beta}^{\text{occ}^\beta} \{A_{a^\beta i^\beta b^\beta j^\beta} + \delta_{ab}\delta_{ij}(\epsilon_b^\beta - \epsilon_j^\beta)\} P^{(2)}_{b^\beta j^\beta} + \sum_{b^\alpha}^{\text{virt}^\alpha} \sum_{j^\alpha}^{\text{occ}^\alpha} \{A_{a^\beta i^\beta b^\alpha j^\alpha}\} P^{(2)}_{b^\alpha j^\alpha} = -L_{a^\beta i^\beta} \quad \text{(A8)}$$

where the orbital Hessian is found by

$$A_{p^\alpha q^\alpha r^\alpha s^\alpha} = 2(p^\alpha q^\alpha|r^\alpha s^\alpha) - (p^\alpha r^\alpha|q^\alpha s^\alpha) - (p^\alpha s^\alpha|q^\alpha r^\alpha) \quad \text{(A9)}$$

$$A_{p^\alpha q^\alpha r^\beta s^\beta} = 2(p^\alpha q^\alpha|r^\beta s^\beta) \quad \text{(A10)}$$

$$A_{p^\beta q^\beta r^\alpha s^\alpha} = 2(p^\beta q^\beta|r^\alpha s^\alpha) \quad \text{(A11)}$$

$$A_{p^\beta q^\beta r^\beta s^\beta} = 2(p^\beta q^\beta|r^\beta s^\beta) - (p^\beta r^\beta|q^\beta s^\beta) - (p^\beta s^\beta|q^\beta r^\beta) \quad \text{(A12)}$$

and the α−α Lagrangian is given by

$$L_{a^\alpha i^\alpha} = \sum_{j^\alpha k^\alpha}^{\text{occ}^\alpha} P^{(2)}_{j^\alpha k^\alpha} A_{a^\alpha i^\alpha j^\alpha k^\alpha} + \sum_{j^\beta k^\beta}^{\text{occ}^\beta} P^{(2)}_{j^\beta k^\beta} A_{a^\alpha i^\alpha j^\beta k^\beta} + \sum_{b^\alpha c^\alpha}^{\text{virt}^\alpha} P^{(2)}_{b^\alpha c^\alpha} A_{a^\alpha i^\alpha b^\alpha c^\alpha} + \sum_{b^\beta c^\beta}^{\text{virt}^\beta} P^{(2)}_{b^\beta c^\beta} A_{a^\alpha i^\alpha b^\beta c^\beta} - 2N_a{}^\alpha \sum_{j^\alpha k^\alpha}^{\text{act}^\alpha} \sum_{b^\alpha}^{\text{virt}^\alpha} \frac{[(j^\alpha a^\alpha|k^\alpha b^\alpha) - (j^\alpha b^\alpha|k^\alpha a^\alpha)]}{D^{a^\alpha b^\alpha}_{j^\alpha k^\alpha}} (i^\alpha j^\alpha|b^\alpha k^\alpha) - 2N_a{}^\alpha \sum_{j^\alpha}^{\text{act}^\alpha} \sum_{k^\beta}^{\text{act}^\beta} \sum_{b^\beta}^{\text{virt}^\beta} \frac{(j^\alpha a^\alpha|k^\beta b^\beta)}{D^{a^\alpha b^\beta}_{j^\alpha k^\beta}} (i^\alpha j^\alpha|b^\beta k^\beta) + 2N_i{}^\alpha \sum_{j^\alpha}^{\text{act}^\alpha} \sum_{b^\alpha c^\alpha}^{\text{virt}^\alpha} \frac{[(i^\alpha b^\alpha|j^\alpha c^\alpha) - (i^\alpha c^\alpha|j^\alpha b^\alpha)]}{D^{b^\alpha c^\alpha}_{i^\alpha j^\alpha}} (a^\alpha b^\alpha|j^\alpha c^\alpha) + 2N_i{}^\alpha \sum_{j^\beta}^{\text{act}^\beta} \sum_{b^\alpha}^{\text{virt}^\alpha} \sum_{c^\beta}^{\text{virt}^\beta} \frac{(i^\alpha b^\alpha|j^\beta c^\beta)}{D^{b^\alpha c^\beta}_{i^\alpha j^\beta}} (a^\alpha b^\alpha|j^\beta c^\beta) \quad \text{(A13)}$$

where

$$N_i^{\alpha} \equiv \begin{cases} 0, \text{ for } i = \text{core} \\ 1, \text{ for } i = \text{active} \end{cases}$$

Expressions for the $\beta-\beta$ matrices are direct analogs of the $\alpha-\alpha$ expressions. The two-particle density matrices are given by

$$\Gamma^{\text{NS}}_{\mu\nu\lambda\sigma} = \sum_{i^\alpha j^\alpha}^{\text{act}^\alpha} \sum_{a^\alpha b^\alpha}^{\text{virt}^\alpha} \frac{[(i^\alpha a^\alpha|j^\alpha b^\alpha) - (i^\alpha b^\alpha|j^\alpha a^\alpha)]}{\epsilon_i^{\ \alpha} + \epsilon_j^{\ \alpha} - \epsilon_a^{\ \alpha} - \epsilon_b^{\ \alpha}} C^\alpha_{\mu i} C^\alpha_{\nu a} C^\alpha_{\lambda j} C^\alpha_{\sigma b} + 2 \sum_{i^\alpha}^{\text{act}^\alpha} \sum_{j^\beta}^{\text{act}^\beta} \sum_{a^\alpha}^{\text{virt}^\alpha} \sum_{b^\beta}^{\text{virt}^\beta} \frac{(i^\alpha a^\alpha|j^\beta b^\beta)}{\epsilon_i^{\ \alpha} + \epsilon_j^{\ \beta} - \epsilon_a^{\ \alpha} - \epsilon_b^{\ \beta}} C^\alpha_{\mu i} C^\alpha_{\nu a} C^\beta_{\lambda j} C^\beta_{\sigma b} + \sum_{i^\beta j^\beta}^{\text{act}^\beta} \sum_{a^\beta b^\beta}^{\text{virt}^\beta} \frac{[(i^\beta a^\beta|j^\beta b^\beta) - (i^\beta b^\beta|j^\beta a^\beta)]}{\epsilon_i^{\ \beta} + \epsilon_j^{\ \beta} - \epsilon_a^{\ \beta} - \epsilon_b^{\ \beta}} C^\beta_{\mu i} C^\beta_{\nu a} C^\beta_{\lambda j} C^\beta_{\sigma b} \quad \text{(A14)}$$

and

$$\Gamma^{\text{S}}_{\mu\nu\lambda\sigma} = P^{(2)}_{\mu\nu}(\alpha\alpha)P^{\alpha\text{SCF}}_{\lambda\sigma} - P^{(2)}_{\mu\lambda}(\alpha\alpha)P^{\alpha\text{SCF}}_{\nu\sigma} + P^{(2)}_{\mu\nu}(\alpha\alpha)P^{\beta\text{SCF}}_{\lambda\sigma} + P^{(2)}_{\mu\nu}(\beta\beta)P^{\beta\text{SCF}}_{\lambda\sigma} - P^{(2)}_{\mu\lambda}(\beta\beta)P^{\beta\text{SCF}}_{\nu\sigma} + P^{(2)}_{\mu\nu}(\beta\beta)P^{\alpha\text{SCF}}_{\lambda\sigma} \quad \text{(A15)}$$

## References and Notes

(1) Whiteside, R. A.; Binkley, J. S.; Colvin, M. E.; Schaefer, H. F., III. *J. Chem. Phys.* **1987**, *86*, 2185−2193.
(2) Møller, C.; Plesset, M. S. *Phys. Rev.* **1934**, *46*, 618−622.
(3) Pople, J. A.; Krishnan, R.; Schlegel, H. B.; Binkley, J. S. *Int. J. Quantum Chem. Symp.* **1979**, *13*, 225.
(4) Handy, N. C.; Amos, R. D.; Gaw, J. F.; Rice, J. E.; Simandiras, E. D. *Chem. Phys. Lett.* **1985**, *120*, 151.
(5) Pulay, P.; Saebø, S. *Theor. Chim. Acta* **1986**, *69*, 357.
(6) Handy, N. C.; Schaefer, H. F. *J. Chem. Phys.* **1984**, *81*.
(7) Lee, T. J.; Racine, S. C.; Rice, J. E.; Rendell, A. P. *Mol. Phys.* **1995**, *85*, 561.
(8) Frisch, M. J.; Head-Gordon, M.; Pople, J. A. *Chem. Phys. Lett.* **1990**, *166*, 275.
(9) Frisch, M. J.; Head-Gordon, M.; Pople, J. A. *Chem. Phys. Lett.* **1990**, *166*, 281.
(10) Head-Gordon, M. *Mol. Phys.* **1999**, *96*, 673.
(11) Fletcher, G. D.; Schmidt, M. W.; Gordon, M. S. In *Advances in Chemical Physics*; Prigogine, I., Rice, S. A., Eds.; John Wiley & Sons: New York, 1999; Vol. 110, pp 267−294.
(12) Fletcher, G. D.; Schmidt, M. W.; Bode, B. M.; Gordon, M. S. *Comput. Phys. Commun.* **2000**, *128*, 190−200.
(13) Nieplocha, J.; Harrison, R. J.; Littlefield, R. J. In *Proceedings of Supercomputing 1994*; IEEE Computer Society Press: Washington, DC, 1994; pp 340−349.
(14) Alexeev, Y.; Kendall, R. A.; Gordon, M. S. *Comput. Phys. Commun.* **2002**, *143*, 69−82.
(15) Alexeev, Y.; Schmidt, M. W.; Windus, T. L.; Gordon, M. S. Manuscript in preparation.
(16) Fletcher, G. D.; Schmidt, M. W.; Gordon, M. S. Manuscript in preparation.
(17) Schmidt, M. W.; Baldridge, K. K.; Boatz, J. A.; Elbert, S. T.; Gordon, M. S.; Jensen, J. H.; Koseki, S.; Matsunaga, N.; Nguyen, K. A.; Su, S.; Windus, T. L.; Dupuis, M.; Montgomery, J. A., Jr. *J. Comput. Chem.* **1993**, *14*, 1347−1363.
(18) Umeda, H.; Koseki, S.; Nagashima, U.; Schmidt, M. W. *J. Comput. Chem.* **2001**, *22*, 1243−1251.
(19) Webb, S. P.; Petrich, J. W.; Gordon, M. S. Manuscript in preparation.
(20) Gan, Z.; Alexeev, Y.; Kendall, R. A.; Gordon, M. S. *J. Chem. Phys.* **2003**, *119*, 47−59.
(21) Colvin, M. E.; Janssen, C. L.; Whiteside, R. A.; Tong, C. H. *Theor. Chim. Acta* **1993**, *84*, 301−314.
(22) Furlani, T. R.; King, H. F. *J. Comput. Chem.* **1995**, *16*, 91−104.
(23) Foster, I. T.; Tilson, J. L.; Wagner, A. F.; Shepard, R. L.; Harrison, R. J.; Kendall, R. A.; Littlefield, R. J. *J. Comput. Chem.* **1996**, *17*, 109−123.
(24) Harrison, R. J.; Guest, M. F.; Kendall, R. A.; Bernholdt, D. E.; Wong, A. T.; Stave, M.; Anchell, J. L.; Hess, A. C.; Littlefield, R. J.; Fann, G. L.; Wagner, A. F.; Foster, I. T.; Lusk, E.; Stevens, R. *J. Comput. Chem.* **1996**, *17*, 124−132.
(25) Fruchtl, H.; Kendall, R. A.; Harrison, R. J.; Dyall, K. G. *Int. J. Quantum Chem.* **1997**, *64*, 63−69.
(26) Furlani, T. R.; King, H. F. In *Quantum Mechanical Simulation Methods for Studying Biological Systems*; Springer-Verlag: Spencer, France, 1996.
(27) Furlani, T. R.; Kong, J.; Gill, P. M. W. *Comput. Phys. Commun.* **2000**, *128*, 170−177.
(28) Marquez, A. M.; Oviedo, J.; Sanz, J. F.; Dupuis, M. *J. Comput. Chem.* **1997**, *18*, 159−168.
(29) Korambath, P. P.; Kong, J.; Furlani, T. R.; Head-Gordon, M. *Mol. Phys.* **2002**, *100*, 1755−1761.
(30) Limaye, A. C.; Gadre, S. R. *J. Chem. Phys.* **1994**, *100*, 1303.
(31) Marquez, A. M.; Dupuis, M. *J. Comput. Chem.* **1995**, *16*, 395−404.
(32) Nielsen, I. M. B.; Seidl, E. T. *J. Comput. Chem.* **1995**, *16*, 1301−1313.
(33) Wong, A. T.; Harrison, R. J.; Rendell, A. P. *Theor. Chim. Acta* **1996**, *93*, 317−321.
(34) Bernholdt, D. E.; Harrison, R. J. *J. Chem. Phys.* **1995**, *102*, 9582−9589.
(35) Bernholdt, D. E. *Chem. Phys. Lett.* **1996**, *250*, 477−484.
(36) Schütz, M.; Lindh, R. *Theor. Chim. Acta* **1997**, *95*, 13−34.
(37) Nielsen, I. M. B. *Chem. Phys. Lett.* **1996**, *255*, 210−216.
(38) Fletcher, G. D.; Rendell, A. P.; Sherwood, P. *Mol. Phys.* **1997**, *91*, 431−438.
(39) Daschel, H.; Lishka, H.; Shepard, R.; Nieplocha, J.; Harrison, R. J. *J. Comput. Chem.* **1997**, *18*, 430−448.
(40) Dobbyn, A. J.; Knowles, P. J.; Harrison, R. J. *J. Comput. Chem.* **1998**, *19*, 1215−1228.
(41) Rendell, A. P.; Guest, M. F.; Kendall, R. A. *J. Comput. Chem.* **1993**, *14*, 1429−1439.
(42) Kobayashi, R.; Rendell, A. P. *Chem. Phys. Lett.* **1997**, *265*, 1−11.
(43) Aikens, C. M.; Webb, S. P.; Bell, R. L.; Fletcher, G. D.; Schmidt, M. W.; Gordon, M. S. *Theor. Chem. Acc.* **2003**, *110*, 233.
(44) Pulay, P. *Chem. Phys. Lett.* **1980**, *73*, 393−398.
(45) Pulay, P. *J. Comput. Chem.* **1982**, *3*, 556−560.
(46) Ditchfield, R.; Hehre, W. J.; Pople, J. A. *J. Chem. Phys.* **1971**, *54*, 724−728.
(47) Hariharan, P. C.; Pople, J. A. *Theor. Chim. Acta* **1973**, *28*, 213−222.
(48) Clark, T.; Chandrasekhar, J.; Spitznagel, G. W.; Schleyer, P. v. R. *J. Comput. Chem.* **1983**, *4*, 294−301.
(49) Stevens, W. J.; Basch, H.; Krauss, M. *J. Chem. Phys.* **1984**, *81*, 6026.
(50) Stevens, W. J.; Basch, H.; Krauss, M.; Jaisen, P. *Can. J. Chem.* **1992**, *70*, 612.
(51) Cundari, T. R.; Stevens, W. J. *J. Chem. Phys.* **1993**, *98*, 5555.
(52) Dunning Jr., T. H. *J. Chem. Phys.* **1989**, *90*, 1007.
(53) Kendall, R. A.; Dunning Jr., T. H.; Harrison, R. J. *J. Chem. Phys.* **1992**, *96*, 6769.