Orbitalinvariant secondorder manybody perturbation theory on parallel computers: An approach for large molecules

David E. Bernholdt and Robert J. Harrison

Citation: The Journal of Chemical Physics **102**, 9582 (1995); doi: 10.1063/1.468774
View online: http://dx.doi.org/10.1063/1.468774
View Table of Contents: http://scitation.aip.org/content/aip/journal/jcp/102/24?ver=pdfcov
Published by the AIP Publishing

Orbital-invariant second-order many-body perturbation theory on parallel computers: An approach for large molecules

David E. Bernholdt and Robert J. Harrison[a)]
Environmental Molecular Sciences Laboratory, Pacific Northwest Laboratory, Richland, Washington 99352-0999

(Received 18 October 1994; accepted 23 March 1995)

The equations for the second-order many-body perturbation theory [MBPT(2)] energy are derived in an orbital-invariant representation, analogous to that obtained with the method of self-consistent electron pairs of Meyer. This formulation is well suited to take advantage of the localized nature of interactions in large chemical systems in order to reduce the computational effort required to study them. This formulation of the MBPT(2) method also lends itself to implementation on parallel computers. We describe a scalable implementation in which the key data are distributed across the parallel computer rather than being replicated. Portability to both shared- and distributed-memory computer architectures is provided through the use of a subroutine library implementing a ''global array'' programming model. We demonstrate that this approach is scalable even for relatively small chemical systems.

I. INTRODUCTION

Advances in computational power in recent years, coupled with algorithmic improvements, have enabled *ab initio* calculations on increasingly large chemical systems. However, the fundamental scaling properties of the electronic-structure methods, quartic in the basis set size for Hartree–Fock (HF) calculations, quintic for second-order many-body perturbation theory [MBPT(2)], and higher for higher orders of perturbation theory, coupled cluster, and configuration interaction methods, mean that interesting calculations can still quickly exceed computational resources. As a result, increasing effort is being applied to the development of algorithms that reduce these scalings to more manageable levels while retaining familiar and fairly well-understood properties of the underlying theoretical method.

At the same time, improvements in computational power are coming increasingly from the use of parallel computers. Indeed, it is widely recognized that massively parallel processors (MPPs) are the most likely route to teraflop (10^{12} floating point operations per second) computers and beyond. For parallel computing, a different sort of scaling is of concern. In order to use MPPs efficiently, it is necessary to employ algorithms that scale with the number of processors and the memory available. In other words, within limits governed by the problem size, the speedup gained by applying p processors to the calculation should be close to p.

As the simplest method to include dynamic correlation effects, MBPT(2) is very important. For many problems, it provides acceptable results for properties for which correlation is important, including reaction energetics, transition states, electronic and vibrational spectra, etc. Consequently, the MBPT(2) method has been the subject of numerous studies aimed at improving the computational scaling and developing efficient parallel implementations.

Most attempts at reducing the computational scaling of *ab initio* methods can be thought of as alternate (approximate) representations for the Gaussian product charge distributions $\phi_\mu\phi_\lambda$ in the two-electron integrals,

$$(\mu\lambda|\nu\sigma) = \int \phi_\mu(\mathbf{r}_1)\,\phi_\lambda(\mathbf{r}_1)\frac{1}{|\mathbf{r}_1-\mathbf{r}_2|} \times \phi_\nu(\mathbf{r}_2)\,\phi_\sigma(\mathbf{r}_2)\,d\tau_1\,d\tau_2. \quad (1)$$

The ''resolution of the identity MP2'' (RI-MP2) approach,[1] employs an auxiliary basis of Gaussian functions to represent the $\phi_\mu\phi_\lambda$ distribution, thereby reducing from four- to three-index integrals. This idea, which has also been applied to HF[2] and coupled cluster calculations,[3] reduces the cost of evaluating Coulomb-like interactions from $O(N^4)$ (for N basis functions) to $O(N^3)$, but exchangelike interactions are not improved. In fact for RI-MP2, they become $O(N^5)$, however, the computational effort shifts from integral evaluation and I/O (in disk-based codes), both of which become $O(N^3)$ in the RI scheme, to simple linear algebra operations (i.e., matrix multiplications). Since most machines handle straightforward linear algebra more efficiently than integral evaluation or I/O, the result is usually faster overall than the exact version.

The pseudospectral (PS) approach, developed by Freisner *et al.* in the context of the Hartree–Fock method,[4] uses a spatial grid to represent the charge distributions in order to evaluate the Coulomb contributions. This idea has been applied to a number of other electronic structure methods,[5,6] most recently by Martinez and Carter to MBPT(2) and MBPT(3).[7] The PS method changes an N^2 factor in the formal scaling of these methods to $N_{\mathrm{occ}}M$, N_{occ} being the number of occupied orbitals and M the size of the mesh. However, as Martinez and Carter point out, spatial locality of the atomic orbitals can be utilized by limiting the sum over grid points to those where the orbital of interest has a non-negligible contribution, which would result in an additional reduction in the scaling by an asymptotic factor of N.

Multipole summation techniques, including the fast multipole method (FMM),[8] were originally developed to sim-

a)The authors may be contacted by e-mail at de_bernholdt@pnl.gov and rj._harrison@pnl.gov.

plify evaluation of Coulomb interactions in molecular dynamics calculations, but have also been applied to *ab initio* electronic structure methods.[9–11] The essence of these methods is that distant charge distributions can be approximated as electrical multipoles, the interactions of which can then be calculated using classical electrostatics. In general, of course, only a subset of interactions will be suitable for the use of multipole methods, and the rest must be obtained through more conventional treatments.

Empirical observation[12,13] and more rigorous theoretical analysis[14] suggest that the number of non-negligible two-electron integrals over Gaussian-like atomic orbitals (AOs) has an asymptotic dependence of $O(N^2(\ln N)^2)$. Clearly screening integral blocks before evaluating them will provide some of the benefits of this reduced scaling, but the basis in which the interactions are represented can play an important role in determining how much of this reduction will be present in practical calculations. Specifically, correlated methods such as MBPT(2) are most commonly formulated in the basis of Hartree–Fock molecular orbitals (MOs) rather than atomic orbitals, which means that using sparsity in the MO integrals is more important than in the AO integrals. However, canonical HF orbitals tend to be delocalized, which increases the number of significant integrals over what would be observed in the AO basis.

It would make more sense to utilize localized molecular orbitals which attempt to reduce the spatial extent of individual orbitals according to some numerical criteria. The usual post-SCF orbital localization procedures, such as Edmiston–Ruedenberg,[15] Foster–Boys,[16] or the more recent scheme of Pipek and Mezey,[17] involve unitary rotations among the occupied orbitals and virtual orbitals (without mixing the two blocks). Because of the invariance properties of the Hartree–Fock (HF) equations, these molecular orbitals (MOs) still satisfy the HF equations and yield the same SCF energy as the original, *canonical*, orbitals. However they do not diagonalize the Fock matrix; they are part of the larger class of *noncanonical* Hartree–Fock (ncHF) orbitals. The ''traditional'' formulation of the MBPT(2) equations (and most other correlated methods) cannot be used in this case, since the usual derivation assumes that the Fock matrix is diagonal in the molecular orbital representation.

The Laplace transformation technique of Almlöf and Häser[18–20] is one of two routes to an MBPT(2) implementation capable of using ncHF references. This approach applies a Laplace transformation to the traditional MBPT(2) in order to replace the denominators with a numerical quadrature problem. As a side effect, the resulting equations are invariant to unitary transformations, thus allowing the use of localized reference functions.

The other alternative is to develop the equations without the assumption of a diagonal Fock matrix. The ability to treat noncanonical reference functions arises naturally in methods derived in the coupled cluster framework, including MBPT(2), but this feature is rarely retained in published presentations. Such a formulation is implicit in the work of Pulay and Saebø[21] wherein the MBPT(2) energy and gradient were presented in the framework of the method of self-consistent electron pairs (SCEP).[22,23] SCEP and other formulations with the same invariance properties offer a significant advantage in the possible avenues to improving the computational scaling: The virtual orbitals can be represented by any nonorthogonal basis that spans the virtual space. Thus instead of HF molecular orbitals, which do not localize very well, it is possible to use the AO basis itself, greatly improving the spatial locality of the individual functions spanning the virtual space and thereby improving sparsity. This is the basis of the ''local correlation'' approach proposed by Saebø and Pulay,[24–26] and should produce an MBPT(2) method which scales asymptotically as $O(N^3)$.

In order to develop an MBPT(2) implementation capable of treating significantly larger chemical systems than are currently possible it is necessary to take advantage of both the improvements in computational scaling, as described above, and the most powerful computers available – massively parallel processors. There have been a number of parallel implementations of both the four-index transformation and the MBPT(2) energy, which have recently been reviewed by Harrison and Shepard.[27–43] In general, these efforts have been limited to relatively small parallel machines (generally $\leqslant 32$ nodes) and display reasonable parallel efficiencies, but scalability to massively parallel processors (MPPs) with hundreds or perhaps thousands of nodes has not been demonstrated. Moreover, all reported parallel MBPT(2) implementations employ the traditional formulation, which does not have the invariance properties required to make use of localized reference functions.

In this paper, we describe the parallel implementation of an MBPT(2) method designed to be scalable to the treatment of large molecules using MPPs. The working equations, which are equivalent to those obtained in the SCEP approach, are derived as transformations of the appropriate MO basis equations, a technique which can be applied to other MBPT and coupled cluster methods. We then discuss the parallelization strategy and present an example of the performance of this method.

II. ORBITAL-INVARIANT MBPT(2) EQUATIONS

The usual presentation of the method of self-consistent electron pairs obscures the relationship between the traditional MO-based equations and their orbital-invariant forms which are readily derived in the SCEP scheme. Perturbation theory through fifth order,[44] coupled cluster methods through full inclusion of quadruple excitations (CCSDTQ),[45] and a variety of related methods, have been presented in diagrammatic and MO-based algebraic forms. Because of this substantial existing work, and because the MO-based presentations are often more familiar, we prefer to think of the orbital invariant form of MBPT(2) and other methods in terms of transformations of the MO equations rather than use a completely separate derivation. Such a transformation is used by Pulay, Saebø, and Meyer,[23] but is not explicated. In this section, we detail the transformation of MO-based MBPT(2) equations to their orbital-invariant form.

We must start with equations applicable to an ncHF reference wave function. In the case of MBPT(2), these equations are readily obtained from either diagrammatic or algebraic methods. We choose to defer further discussion as to

their origin to the Appendix and simply state them here. The second-order contribution to the correlation energy is

$$E^{(2)}=\frac{1}{4}\sum_{i,j,a,b}[(ia|jb)-(ib|ja)]t_{ij}^{ab}. \tag{2}$$

We use the convention that Greek letters refer to atomic orbitals, $i,j,...$ refer to orbitals which are occupied in the Hartree–Fock reference wave function, and $a,b,...$ refer to unoccupied, or virtual MOs. Integrals are written in the Mulliken notation as defined in Eq. (1). The double excitation amplitudes, t, represent the amplitude for simultaneous excitation of an electron from orbital i to a and another from j to b and are determined by the equation

$$0=r_{ij}^{ab}=[(ia|jb)-(ib|ja)]+\sum_{e}(t_{ij}^{ae}f_{be}+f_{ae}t_{ij}^{eb}) - \sum_{m}(t_{im}^{ab}f_{mj}+f_{im}t_{mj}^{ab}),\quad \forall i,j,a,b. \tag{3}$$

Here, f is the Fock matrix of the Hartree–Fock reference function and the residual, r_{ij}^{ab}, is used in the iterative solution of these equations.

We now recast the equations in the form of coupled matrix equations. The amplitudes are collected into a set of matrices, $\{\mathbf{t}_{ij}\}$ which are labeled by virtual orbitals a and b in their row and column dimensions. Similarly, the integrals are collected as a set of exchange operators, $\mathbf{K}_{ij}^{ab}=(ia|jb)$. The MBPT(2) energy is now represented as

$$E^{(2)}=\frac{1}{4}\sum_{i,j}[\mathbf{t}_{ij}\cdot(\mathbf{K}_{ij}-\mathbf{K}_{ij}^{\dagger})], \tag{4}$$

and the amplitude equation becomes

$$0=\mathbf{r}_{ij}=(\mathbf{K}_{ij}-\mathbf{K}_{ij}^{\dagger})+\mathbf{F}\mathbf{t}_{ij}+\mathbf{t}_{ij}\mathbf{F} - \sum_{m}f_{im}\mathbf{t}_{mj}-\sum_{m}\mathbf{t}_{im}f_{mj},\forall i,j. \tag{5}$$

Note that the final four terms can all be cast as matrix multiplications.

Any transformation $\mathbf{C}$ of the MOs which satisfies the orthonormality constraint $\mathbf{C}^{\dagger}\mathbf{S}\mathbf{C}=\mathbf{1}$ for some metric $\mathbf{S}$ may be used to transform the excitation space. Without loss of generality, we can therefore choose the AO basis to represent the excitation space. Orthogonality of the occupied and excitation spaces can be guaranteed by appropriate use of projection operators during solution of the amplitude equations.[22] The transformation rules for the Fock matrix and exchange operators,

$$\mathbf{F}^{\mathrm{MO}}=\mathbf{C}^{\dagger}\mathbf{F}^{\mathrm{AO}}\mathbf{C}, \tag{6}$$

$$\mathbf{K}_{ij}^{\mathrm{MO}}=\mathbf{C}^{\dagger}\mathbf{K}_{ij}^{\mathrm{AO}}\mathbf{C}, \tag{7}$$

can be inserted into Eq. (3), giving

$$0=\mathbf{r}_{ij}^{\mathrm{MO}}=\mathbf{C}^{\dagger}\mathbf{K}_{ij}^{\mathrm{AO}}\mathbf{C}-\mathbf{C}^{\dagger}\mathbf{K}_{ij}^{\mathrm{AO}\dagger}\mathbf{C}+\mathbf{C}^{\dagger}\mathbf{F}^{\mathrm{AO}}\mathbf{C}\mathbf{t}_{ij}^{\mathrm{MO}} +\mathbf{t}_{ij}^{\mathrm{MO}}\mathbf{C}^{\dagger}\mathbf{F}^{\mathrm{AO}}\mathbf{C}-f_{im}\mathbf{t}_{mj}^{\mathrm{MO}}-\mathbf{t}_{im}^{\mathrm{MO}}f_{mj}. \tag{8}$$

Using the orthogonality condition, $\mathbf{C}^{\dagger}\mathbf{S}\mathbf{C}=\mathbf{1}$, to eliminate the $\mathbf{C}^{\dagger}$ and $\mathbf{C}$ surrounding the first four terms of the right-hand side leaves

$$0=\mathbf{S}\mathbf{C}\mathbf{r}_{ij}^{\mathrm{MO}}\mathbf{C}^{\dagger}\mathbf{S}=\mathbf{K}_{ij}^{\mathrm{AO}}-\mathbf{K}_{ij}^{\mathrm{AO}\dagger}+\mathbf{F}^{\mathrm{AO}}\mathbf{C}\mathbf{t}_{ij}^{\mathrm{MO}}\mathbf{C}^{\dagger}\mathbf{S} +\mathbf{S}\mathbf{C}\mathbf{t}_{ij}^{\mathrm{MO}}\mathbf{C}^{\dagger}\mathbf{F}^{\mathrm{AO}}-f_{im}\mathbf{S}\mathbf{C}\mathbf{t}_{mj}^{\mathrm{MO}}\mathbf{C}^{\dagger}\mathbf{S} -\mathbf{S}\mathbf{C}\mathbf{t}_{im}^{\mathrm{MO}}\mathbf{C}^{\dagger}\mathbf{S}f_{mj}, \tag{9}$$

from which we can identify the transformation rules for the amplitudes and residuals,

$$\mathbf{t}_{ij}^{\mathrm{AO}}=\mathbf{C}\mathbf{t}_{ij}^{\mathrm{MO}}\mathbf{C}^{\dagger}, \tag{10}$$

$$\mathbf{r}_{ij}^{\mathrm{AO}}=\mathbf{C}\mathbf{r}_{ij}^{\mathrm{MO}}\mathbf{C}^{\dagger} \tag{11}$$

giving a final result of

$$0=\mathbf{S}\mathbf{r}_{ij}^{\mathrm{AO}}\mathbf{S}=\mathbf{K}_{ij}^{\mathrm{AO}}-\mathbf{K}_{ij}^{\mathrm{AO}\dagger}+\mathbf{F}^{\mathrm{AO}}\mathbf{t}_{ij}^{\mathrm{AO}}\mathbf{S}+\mathbf{S}\mathbf{t}_{ij}^{\mathrm{AO}}\mathbf{F}^{\mathrm{AO}} -\sum_{m}f_{im}\mathbf{S}\mathbf{t}_{mj}^{\mathrm{AO}}\mathbf{S}-\sum_{m}\mathbf{S}\mathbf{t}_{im}^{\mathrm{AO}}\mathbf{S}f_{mj}\forall i,j. \tag{12}$$

Inserting the transformation rules into the energy equation gives

$$E^{(2)}=\tfrac{1}{4}\sum_{i,j}[\mathbf{t}_{ij}^{\mathrm{AO}}\cdot(\mathbf{K}_{ij}^{\mathrm{AO}}-\mathbf{K}_{ij}^{\mathrm{AO}\dagger})]. \tag{13}$$

Notice that all two-index quantities are in the AO basis except for f_{im} and f_{mj} in the last two terms of Eq. (12), which remain in the MO basis. The four-index quantities, t, r, and K are in a mixed representation, with AOs for the correlating space and (localized) occupied molecular orbitals.

III. PARALLELIZATION STRATEGY

While there have been a number of efforts towards parallel MBPT(2) using the traditional formulation,[32,36,39–43] this is the first attempt to parallelize the orbital invariant equations. As such, its implementation is guided by a number of basic requirements. Foremost among them is the requirement that the code be as simple and as modular as possible to facilitate its use as a testbed for new algorithms—including, for example, developing an efficient parallel implementation of the local correlation approximation.

Few MPP installations provide I/O capability which is useful to a scalable program. In addition, there is little consensus yet on the design or programming semantics of parallel file systems. Therefore the portability of programs which do parallel file I/O is problematical at this stage. Consequently this implementation avoids I/O, storing $\mathbf{K}$, $\mathbf{t}$, and $\mathbf{r}$ in main memory. Some of the impact of this requirement and prospects for easing it are discussed below.

A third important requirement is portability of the program to a variety of MPPs with differing architectures. The two main architectures are distributed and shared memory, according to whether each CPU has direct access to only a segment of the machine's memory or to all of it. A third type

of machine has recently emerged which combines the two: the underlying architecture is distributed memory, but through the use of additional hardware (as in the Kendall Square Research products) and/or software (as in the Cray T3D) the machine presents a shared-memory environment to the user. The concept of nonuniform memory access costs (NUMA) can be used to provide a portable programming model for all three categories of machines, as well as for sequential workstations. The NUMA idea is already present in modern computer workstations, which have a hierarchy of memory with different access times: registers, cache, physical memory, virtual memory. On parallel computers, there is simply another level in the hierarchy: off-processor physical memory.

In order to provide a portable shared NUMA memory programming environment, we have designed and implemented a subroutine library to support one-sided access to distributed one- and two-dimensional arrays.[46,47] These ''global array tools'' have been implemented on top of native message-passing facilities on Intel parallel computers, the IBM SPX, on networks of workstations using a portable message-passing facility (TCGMSG[48,49]), and on shared-memory multiprocessors, such as the Kendall Square Research KSR/Series, using standard UNIX shared memory facilities. These tools have been used quite successfully in our fully distributed parallel SCF,[50,51] DFT,[52] and RI-MP2[53] implementations, and are also being used in parallelization of the COLUMBUS program package,[54] and a coupled cluster code.[55] When combined with algorithms *designed* with NUMA in mind, the global array tools so far appear to offer a portable, easy-to-use environment for the implementation of parallel electronic structure methods.

The two dominant computational steps of the orbital invariant MBPT(2) are construction of the exchange operators **K** and evaluation of the residual of the amplitude equation **r**. We will discuss these two sections in detail, followed by a description of the remaining steps: updating the amplitudes and evaluation of the energy.

A. Construction of the exchange operators

The exchange operators are constructed from the equation

$$K_{ij}^{\lambda\sigma}=\sum_{\mu,\nu} C_{\mu i}^{\dagger}(\mu\lambda|\nu\sigma)C_{\nu j}, \tag{14}$$

which is essentially one half of the four-index transformation required for MO-based post-SCF methods. Many researchers have proposed parallel four-index transformation algorithms.[28–38] However, many of these algorithms are targeted for nonhierarchical shared memory architectures and do not adapt very well to the blocking necessary for good performance with NUMA programming models. Transformations targeted at distributed memory architectures also suffer a number of problems, in particular substantial memory requirements on each node and multiple passes over the integrals.

For the current version of the code we have implemented an algorithm which, while certainly not optimal, is conceptually quite simple, making it fairly easy to implement, and which can be improved upon in the future.

The primary concerns in the design of this algorithm were to avoid the need to replicate integral evaluation on every node, since this is inherently nonscalable; and at the same time to minimize interprocessor communications. We adopt a data distribution for $K_{ij}^{\lambda\sigma}$ which can be characterized as ''all ij for some $\lambda\sigma$''—each processor has a rectangular panel of the $\lambda\sigma$ matrix, and all ij for that panel. With this distribution, all of the integrals which contribute to each processor's panel are known. Due to permutational symmetry, each integral contributes to up to eight panels (assuming only the unique $\mathbf{K}_{i\geqslant j}$ are evaluated; four panels if all ij are evaluated, but at the cost of twice the memory), so without any sharing of integrals or interprocessor communication, we can construct **K** with, in effect, eight passes over the integrals.

Integral evaluation is performed over shell blocks of basis functions. Each processor loops through the canonical list of shell quartets $[(\alpha\beta|\gamma\delta)$ with $\alpha\geqslant\beta$, $\gamma\geqslant\delta$, and $\alpha\beta\geqslant\gamma\delta]$ and evaluates those which contribute to its panel. Each batch of integrals is then transformed to give **K**.

The work of the transformation is broken down into two quarter transformations. In order to account for all of the permutational symmetries, the first quarter transformation is done twice, with four second quarter transformations following from each. (Of course only those transformations which actually contribute to the processor's panel are performed.)

Although in the interest of simplicity the current code does not include any sparsity checking, this will be part of our investigation of controllable local correlation methods based on the ideas of Saebø and Pulay.[24–26]

The optimal data distribution for construction of the exchange operators is not necessarily the best one for their use. As explained below, the evaluation of the residual and the energy is done using an ''all $\lambda\sigma$ for some ij'' distribution—each $\mathbf{K}_{ij}$ matrix is a block, and each node has one or more blocks. So the final step in the exchange operator construction involves a redistribution of data, with each processor sending its $\lambda\sigma$ panel for each ij to the target processor holding that $\mathbf{K}_{ij}$. This ''personalized all-to-all'' communication can be easily implemented using the global array tools by having each node put the data that it owns initially into the appropriate panels of the global **K** data structure. This step involves the movement of $O(N^4)$ data [compared to $O(N^5)$ computation], and requires additional storage of the same size as **K**.

B. Evaluation of the residual

The other computationally significant step is evaluation of the residual **r**, which is computed using a modified form of Eq. (12) discussed in more detail below.

Using the same ''all ij for some $\lambda\sigma$'' data distribution in which the exchange operators are evaluated, the third and fourth terms, constituting most of the work, would be distributed matrix multiplications, and the final two terms could be implemented either as local matrix multiplications or local matrix scale-and-sum (DAXPY[56,57]) operations. Distributed matrix multiplications, however, require on the order of 400

$\times 400$ element panels on each processor in order to obtain parallel efficiencies in the neighborhood of 80% on the Intel Touchstone Delta.[58,59] This is impractical on most current MPPs since each processor would have to hold (1/2) $N_{\mathrm{occ}}(N_{\mathrm{occ}}+1)$ such panels.

By distributing each $\mathbf{K}_{ij}$ over a few processors in order to maintain the larger block sizes required by the distributed matrix multiplication, and by having only a few $\mathbf{K}_{ij}$ on each group of processors, it might be possible to employ a distributed matrix multiplication effectively. However, this approach would make communications necessary to evaluate the final two terms of Eq. (12), and when using sparsity of the amplitudes, care would have to be taken that the nonzero blocks of the $\mathbf{t}_{ij}$ on each processor group resulted in balanced storage and computation requirements across the group. This could be accomplished by careful assignment of the $\mathbf{t}_{ij}$ to processor groups. The increased complexity of this approach, and the substantial extension that would be required to the current version of the global array tools, led us to choose a simpler algorithm for the present implementation; but this approach may prove useful in the future.

In the current code, we have chosen instead the ''all $\lambda\sigma$ for some ij'' distribution for the evaluation of Eq. (12). That is, the exchange operators, amplitudes, and residuals are arranged as $\mathbf{K}_{ij}$, $\mathbf{t}_{ij}$, and $\mathbf{r}_{ij}$ matrices, with one or more matrices of each object being stored on each node. With this distribution, the third and fourth terms are local matrix multiplications (as long as the Fock and overlap matrices are available on each node), and the final two terms involve a series of matrix DAXPYs with amplitudes from other nodes. Handled in this way, the communication volume is roughly $N_{\mathrm{occ}}^3 N^2$ words in N_{occ}^3 messages while the overall computational requirement for the residual evaluation is $O(N_{\mathrm{occ}}^2 N^3)$. This is not a favorable ratio of computation to communication time. In addition, while it would be possible to overlap the communication with the computation to some extent, the DAXPY operation is only N^2 floating point operations, which would mask only a fraction of the communication time.

Observe that by switching from a distribution represented by matrices $\mathbf{t}_{ij}$ to one represented by $\mathbf{t}^{\lambda\sigma}$ would allow the last two terms of Eq. (12) to be expressed as local matrix multiplications, $f\mathbf{t}^{\lambda\sigma}$ and $\mathbf{t}^{\lambda\sigma}f$. However, this data distribution is impractical for the rest of Eq. (12), as it would recast the third and fourth terms as multiple nonlocal matrix DAXPY operations. But an algorithm that evaluates the final terms of the residual equation, as

(1) redistribute $\mathbf{t}_{ij}$ and $\mathbf{r}_{ij}$ to $\mathbf{t}^{\lambda\sigma}$ and $\mathbf{r}^{\lambda\sigma}$,
(2) perform the matrix multiplications $\mathbf{r}^{\lambda\sigma} \leftarrow -f\mathbf{t}^{\lambda\sigma} - \mathbf{t}^{\lambda\sigma}f$,
(3) redistribute back to $\mathbf{t}_{ij}$ and $\mathbf{r}_{ij}$,

moves roughly $(3/2)N_{\mathrm{occ}}^2 N^2$ words of data in less than $(3/2)N_{\mathrm{occ}}^2$ messages (assuming the number of processors is $\lesssim(1/2)N_{\mathrm{occ}}^2$). This is better than the DAXPY algorithm, but additional storage of the size of $\mathbf{r}$ or $\mathbf{t}$ is required for the redistribution. We have therefore retained both algorithms in the present code, with the ability to switch between them as appropriate.

C. Amplitude update and energy evaluation

The remaining work in solving the MBPT(2) problem involves updating the amplitudes from the residual and finally, evaluating the energy. These steps are quite straightforward with the data distribution used to evaluate the residuals, each processor operating on that data which actually resides in its local memory.

The MBPT(2) energy, Eq. (13), is equally simple to evaluate. Each ij pair contributes an amount given by the sum of the elementwise product (matrix dot product) of the integrals with the amplitudes. Once again, all the required quantities are local, allowing each node to compute a partial sum of the pair energies for its pairs without any communication. The total energy is obtained by combining the partial sums from each processor in a global sum operation.

Since the energy itself costs only $O(N^4)$, it can be evaluated with each iteration of the amplitude equation at negligible cost (assuming the exchange operators are available). But this is not necessary, as convergence can be determined directly from the changes in the amplitudes from one iteration to the next.

IV. RESULTS AND DISCUSSION

These equations and algorithms have been implemented in a portable fashion utilizing our global array tools, described above, and broadcast and global sum functions from the TCGMSG message-passing toolkit. The most experience with this implementation has been gained on the Intel Touchstone Delta, a prototype for the Intel Paragon.

The Delta is a 16×32 mesh of Intel i860 processors (40 MHz), with additional Intel 386 nodes at the edges of the mesh to provide I/O, networking, interactive access, and other service functions to the system. The compute nodes have 16 Mbytes of memory each, of which about 9 MBytes is available for use within the MBPT(2) code after the operating system and the application itself are accounted for. Communication performance using the global array tools is about 8 Mbyte/s bandwidth with latencies on the order of 300 μs.

In Fig. 1 we display the parallel speedup obtained with the C_4H_{10} molecule in a DZP-quality basis. This is a relatively small system, having only 118 basis functions and 17 occupied orbitals. Since for most ''reasonable'' parallel algorithms, parallel efficiency increases as the size of the problem increases (due to surface area to volume effects), an example of moderate size, such as the one we have chosen, represents a tougher test of the scalability of the method than a larger molecule would. This calculation required 2700 s to run on 16 nodes of the Delta. However, the current version of the code was implemented to study the parallelization of the method and, as discussed above, we have deferred consideration of a number of issues that would impact the execution time of the code, but which would have little impact on the parallel performance, the focus of this paper. We therefore consider not the absolute run time, but rather the improvement in that time as more processors are applied to the problem.

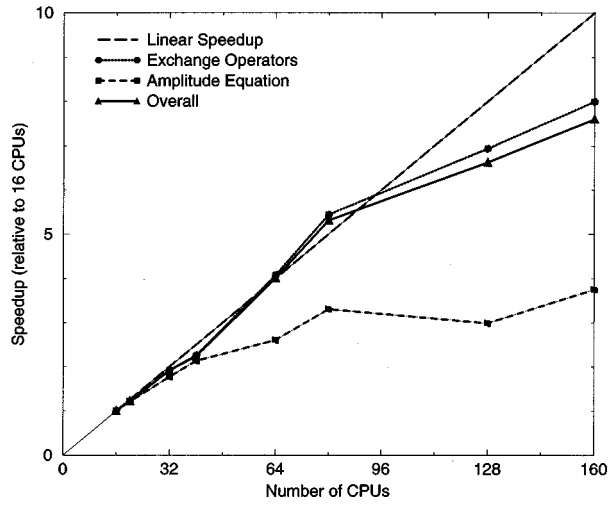

FIG. 1. Parallel speedups obtained for C_4H_{10} on the Intel Touchstone Delta, measured relative to the 16 CPU timings.

The dominant computational step in the calculation is the evaluation of the exchange operators **K**. Speedups for this step are represented by the dotted line in Fig. 1. Scaling is extremely good in this stage of the calculation because the redistribution of the computed exchange operator is the only communication involved, and it constitutes only 0.1%–0.4% of the time required for this stage (varying with the number of processors). The remainder of this stage is trivially parallel, and the small parallel performance losses are due mainly to load imbalance (different blocks of integrals requiring different effort to evaluate) and unfavorable distribution of the initial data. The latter problem arises from the fact that we are currently using a regular blocked distribution of the $\mathbf{K}_{ij}$ elements to the processors, while the shell block boundaries governing the integral evaluation are not regular. We have experimented with irregular distributions along shell block boundaries and found that this problem is easily fixed, but straightforward algorithms to compute the irregular distributions frequently do not make use of all the available processors. Since this problem would probably be obviated by a more sophisticated transformation algorithm, as discussed in Sec. III A, we have not invested the time necessary to find a general solution to it.

As more sophisticated integral evaluation and transformation procedures are incorporated into the code, the computational effort in this step will be reduced. At the same time, by taking advantage of sparsity, the amount of data to be redistributed will also be reduced. As the code matures therefore, we expect that this stage of the calculation will continue to scale well with the number of processors, while at the same time improving significantly in actual time to solution.

After the exchange operator construction, the other important step is solution of the amplitude equations, represented by the dashed line in Fig. 1. The parallel performance for this step is not as good as that observed for the exchange operator construction, but this step is not as time consuming (less than 5% of the overall time in this case), so it does not have too much impact on the overall performance. The amplitude evaluation differs from the exchange matrix construction step in that the redistributions required by the amplitude evaluation algorithm constitute a significant portion of the time required for the step. Although the formal costs are $O(N^5)$ for computation and $O(N^4)$ for communication, the relatively slow communications on the Delta result in comparable timings for the two components in the systems we studied. As the problem size we are able to treat increases, or on MPPs with better communication performance relative to their computational ability, the communication costs will represent a smaller fraction of the cost of this step, resulting in better parallel scalability and the ability to effectively use larger MPPs for a given problem.

Since the MBPT(2) problem involves $O(N^5)$ computation and $O(N^4)$ data, we cannot expect parallel performance to be as good as that of the Hartree–Fock problem,[51] which has $O(N^4)$ computation and $O(N^2)$ data. We consider the overall performance obtained in our parallel MBPT(2), represented by the solid line in Fig. 1 to be quite good, and we believe that we will be able to maintain this level of performance while incorporating more efficient integral evaluation techniques and sparsity. This also bodes well for massively parallel implementation of higher-level methods, such as MBPT(3), SDQ-MBPT(4), and CCSD, for which the computational cost is $O(N^6)$, but the amount of data remains $O(N^4)$.

V. CONCLUSIONS

We have presented an alternative derivation of the MBPT(2) energy in an orbital-invariant form; the same techniques used in this paper can be applied to other MBPT and coupled cluster methods to obtain orbital invariant versions.

The dominant computational steps, exchange operator construction (a half-transformation of the two-electron integrals), and evaluation of the residual of the amplitude equation, were the primary focus of parallelization. Once reasonable algorithms and data distributions were chosen, the remaining computational steps were readily parallelized. Our approach involves the use of explicit redistribution of data between the exchange operator construction step and the amplitude evaluation steps—and, optionally, another explicit redistribution within the amplitude evaluation step. The explicit communication steps make it difficult to hide communication costs by overlapping them with computation, but the overall communication volume is significantly lower than for alternative algorithms that intermix computation and communication.

Preliminary results presented here are perhaps the first demonstration that MBPT(2) methods can be implemented in a fashion which is scalable to MPPs having several hundred processors. Theoretical considerations indicate that it should be possible to utilize $(1/2)N_{\mathrm{occ}}^2$ processors efficiently. Our decision to avoid I/O, which is generally extremely slow on current MPPs, results in substantial memory requirements, and this limits our ability to run larger systems on the Intel Touchstone Delta. More modern machines, (e.g., Intel Paragon, Kendall Square Research KSR/Series, and IBM SPX)

which offer both more memory per node and virtual memory facilities, can handle larger problems.

Having established that orbital-invariant MBPT(2) can indeed be implemented efficiently in parallel, we plan to proceed with the next stage of the project, which is to reduce the computational scaling for large molecules while maintaining parallel efficiency.

ACKNOWLEDGMENTS

This work benefited greatly from discussions with Rick Kendall, Martyn Guest, and Rik Littlefield (Pacific Northwest Laboratory), Alistair Rendell (Daresbury Laboratory, UK), and Hans Lischka (University of Vienna, Austria). The work of Jarek Nieplocha and Greg Thomas (PNL) on implementing some of the software tools underlying the MBPT(2) program is also very much appreciated. D.E.B. would also like to thank the Dépertement de Physique of the Université de Sherbrooke, Québec, Canada, and in particular Jan Musfeldt, for their hospitality during several visits in the course of this work. This research we performed in part using the CSCC Intel Touchstone Delta operated by Caltech on behalf of the Concurrent Supercomputing Consortium. Access to this facility was provided by the Pacific Northwest Laboratory. Support for this work was provided by the High Performance Computing and Communications Program of the Office of Scientific Computing, U. S. Department of Energy under Contract No. DE-AC-6-76RLO-1830 with Battelle Memorial Institute which operates the Pacific Northwest laboratory, and Contract No. DE-FG06-89ER-75522 with Associated Western Universities, Inc.—Northwest Division.

APPENDIX: MO-BASED MBPT(2) EQUATIONS FOR NONCANONICAL HARTREE–FOCK REFERENCE WAVE FUNCTIONS

Although the MO-based ncHF-MBPT(2) equations are easily derived using either diagrammatic or algebraic methods, they do not appear to have been previously published in the open literature. We therefore take this opportunity to briefly discuss their derivation.

A convenient starting place is the doubles amplitude equation of the CCSDT method, as presented by Lauderdale *et al.*,[60] from which, for our purposes, we can immediately eliminate the single and triple excitation amplitude contributions, leaving

$$\langle {}^{ab}_{ij}|W+fT_2+WT_2+\tfrac{1}{2}WT_2^2|0\rangle_C=0. \tag{A1}$$

In this notation, f represents the one-body component of the Hamiltonian (the Fock operator) and W the two-body term (the perturbation). Note that Lauderdale and co-workers were interested in non-Hartree–Fock references, but for the ncHF case, f is of course block diagonal; so their distinction between the off-diagonal f_{ov} component and the block diagonal (occupied–occupied and virtual–virtual) component f_D is not necessary here.

The second-order energy is determined by the first-order wave function, and therefore the first-order approximation to T_2 is required. W is first-order in correlation, while the block diagonal f is taken as zeroth order. Thus we can easily obtain the operator form of the first-order amplitudes, which we denote $T_2^{(1)}$, by retaining only those terms in Eq. (A1) which are first order in the perturbation,

$$\langle {}^{ab}_{ij}|W+fT_2^{(1)}|0\rangle_C=0. \tag{A2}$$

Evaluating this expression using Wick's theorem (see, for example, Ref. 44) results in Eq. (3), above. The energy, Eq. (2), is identical for canonical or noncanonical HF references.

[1] M. Feyereisen, G. Fitzgerald, and A. Komornicki, Chem. Phys. Lett. **208**, 359 (1993).
[2] O. Vahtras, J. Almlöf, and M. W. Feyereisen, Chem. Phys. Lett. **213**, 514 (1993).
[3] A. P. Rendell and T. J. Lee, J. Chem. Phys. **101**, 400 (1994).
[4] M. N. Ringnalda, M. Belhadj, and R. A. Friesner, J. Chem. Phys. **93**, 3397 (1990).
[5] J.-M. Langlois, R. P. Muller, T. R. Coley, W. A. Goddard III, M. N. Ringnalda, Y. Won, and R. A. Friesner, J. Chem. Phys. **92**, 7488 (1990).
[6] T. J. Martinez, A. Mehta, and E. A. Carter, J. Chem. Phys. **97**, 1876 (1992).
[7] T. J. Martinez and E. A. Carter, J. Chem. Phys. **100**, 3631 (1994).
[8] L. F. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems*, ACM Distinguished Dissertations (MIT, Cambridge, MA, 1988).
[9] V. R. Saunders, C. Freyria-Fava, R. Dovesi, L. Salasco, and C. Roetti, Mol. Phys. **77**, 629 (1992).
[10] I. Panas and J. Almlöf, Int. J. Quantum Chem. **42**, 1073 (1992).
[11] C. A. White, M. Head-Gordon, B. G. Johnson, and P. M. W. Gill, The continuous fast multipole method, Contributed Talk, 15th West Coast Theoretical Chemistry Conference, Sandia National Laboratory, Livermore, California, 1994.
[12] E. Clementi, G. Corngiu, and S. Chakravorty, in *Modern Techniques in Computational Chemistry: MOTECC–90*, edited by E. Clementi (ESCOM Science, Leiden, 1990), Chap. 7, pp. 343–434.
[13] J. Almlöf (private communication).
[14] V. Dyczmons, Theor. Chim. Acta **28**, 307 (1973).
[15] C. Edmiston and K. Ruedenberg, Rev. Mod. Phys. **35**, 457 (1963).
[16] S. F. Boys, in *Quantum Theory of Atoms, Molecules, and the Solid State*, edited by P.-O. Löwdin (Academic, New York, 1966), pp. 253–262.
[17] J. Pipek and P. G. Mezey, J. Chem. Phys. **90**, 4916 (1989).
[18] J. Almlöf, Chem. Phys. Lett. **181**, 319 (1991).
[19] M. Häser and J. Almlöf, J. Chem. Phys. **96**, 489 (1992).
[20] M. Häser, Theor. Chim. Acta **87**, 147 (1993).
[21] P. Pulay and S. Saebø, Theor. Chim. Acta **69**, 357 (1986).
[22] W. Meyer, J. Chem. Phys. **64**, 2901 (1976).
[23] P. Pulay, S. Saebø, and W. Meyer, J. Chem. Phys. **81**, 1901 (1984).
[24] P. Pulay, Chem. Phys. Lett. **100**, 151 (1983).
[25] S. Saebø and P. Pulay, Chem. Phys. Lett. **113**, 13 (1985).
[26] S. Saebø and P. Pulay, Annu. Rev. Phys. Chem. **44**, 213 (1993).
[27] R. J. Harrison and R. Shepard, Annu. Rev. Phys. Chem. **45**, 623 (1994).
[28] E. Clementi, S. Chin, and D. Logan, Lecture Notes in Chemistry **44**, 130 (1986).
[29] R. A. Whiteside, J. S. Binkley, M. E. Colvin, and H. F. Schaefer III, J. Chem. Phys. **86**, 2185 (1987).
[30] J. N. Hurley, D. L. Huestis, and W. A. Goddard III, J. Phys. Chem. **92**, 4880 (1988).
[31] C. W. Bauschlicher, Jr., Theor. Chim. Acta **76**, 187 (1989).
[32] J. D. Watts and M. Dupuis, Vector and parallel implementations of a fourth-order Moller–Plesset perturbation theory (MP4) program, Technical Report No. KGN-197, IBM Corporation, Data Systems Division, Kingston, NY 12401, 1989.
[33] L. A. Covick and K. M. Sando, J. Comput. Chem. **11**, 1151 (1990).
[34] R. Wiest, J. Demuynck, M. Bénard, M. Rohmer, and R. Ernenwein, Comput. Phys. Commun. **62**, 107 (1991).
[35] R. J. Harrison, Technical Report No. CCSF-14-92, Caltech Concurrent Supercomputing Facility, Pasadena, CA, 1992.
[36] A. C. Limaye and S. R. Gadre, J. Chem. Phys. **100**, 1303 (1994).
[37] M. Dupuis, S. Chin, and A. Marquez, in *Relativistic and Electronic Correlation Effects in Molecules and Clusters*, edited by G. L. Malli (Plenum, New York, 1994), Vol. 318, pp. 315–338.
[38] T. L. Windus, M. E. Schmidt, and M. S. Gordon, Theor. Chim. Acta **89**, 77 (1994).
[39] J. D. Watts and M. Dupuis, J. Comput. Chem. **9**, 158 (1988).

[40] D. Moncrieff, D. J. Baker, and S. Wilson, Comput. Phys. Commun. **55**, 31 (1989).
[41] M. E. Colvin, R. A. Whiteside, and H. F. Schaefer III, in *Concurrent Computation in Chemical Calculations*, edited by S. Wilson (Plenum, New York, 1989), pp. 167–237.
[42] M. E. Colvin, Ph.D. thesis, Department of Chemistry, University of California, Berkely, California, 1986.
[43] S. Wilson, in *Concurrent Computation in Chemical Calculations*, edited by S. Wilson (Plenum, New York, 1989), pp. 1–61.
[44] S. A. Kucharski and R. J. Bartlett, Adv. Quantum Chem. **18**, 281 (1986).
[45] S. A. Kucharski and R. J. Bartlett, Theor. Chim. Acta **80**, 387 (1991).
[46] J. Nieplocha, R. J. Harrison, and R. J. Littlefield, in *Supercomputing'94* (Institute of Electrical and Electronics Engineers and Association for Computing Machinery IEEE Computer Society, Los Alamitos, 1994).
[47] J. Nieplocha and R. J. Harrison, Global array tools library, available for anonymous FTP from ftp.pnl.gov in the pub/global directory.
[48] R. J. Harrison, Int. J. Quantum Chem. **40**, 847 (1991).
[49] R. J. Harrison, TCGMSG, available for anonymous FTP from ftp.tcg.anl.gov in the pub/tcgmsg directory.
[50] I. T. Foster, J. Tilson, A. F. Wagner, R. Shepard, R. J. Harrison, R. A. Kendall, and R. J. Littlefield, J. Comput. Chem. (to be published).
[51] R. J. Harrison, M. F. Guest, R. A. Kendall, D. E. Bernholdt, A. T. Wong, M. Stave, J. Anchell, A. Hess, R. Littlefield, G. I. Fann, J. Nieplocha, G. S. Thomas, D. Elwood, J. Tilson, R. L. Shepard, A. F. Wagner, I. T. Foster, E. Lusk, and R. Stevens, J. Comput. Chem. (to be published).
[52] J. L. Anchell, E. Aprá, A. C. Hess, J. A. Nichols, M. S. Stave, and H. L. Taylor (unpublished work).
[53] D. E. Bernholdt and R. J. Harrison (in preparation).
[54] H. Lischka, H. Dachsel, R. Shepard, and R. J. Harrison, in *Parallel Computing in Computational Chemistry*, edited by T. G. Mattson (American Chemical Society, Washington, DC, 1994).
[55] A. P. Rendell (private communication).
[56] C. Lawson, R. Hanson, D. Kincaid, and F. Krogh, ACM Trans. Math. Software **5**, 308 (1979).
[57] C. Lawson, R. Hanson, D. Kincaid, and F. Krogh, ACM Trans. Math. Software **5**, 324 (1979).
[58] S. Huss-Lederman, E. M. Jacobson, A. Tsao, and G. Zhang, Matrix multiplication on the Intel Touchstone Delta, Technical Report. SRC-TR-93-101, Supercomputing Research Center, 17100 Science Drive, Bowie, MD 20715-4300, 1993. This paper is PRISM Working Note No. 11, available via anonymous FTP to ftp.super.org in the directory pub/prism.
[59] J. Choi, J. J. Dongarra, and D. W. Walker, Technical Report No. ORNL/TM-12252, Oak Ridge National Laboratory, Oak Ridge, TN 37831 (1993), available on the Internet using the URL http://www.netlib.org/scalapack/pumma.ps.
[60] W. J. Lauderdale, J. F. Stanton, J. Gauss, J. D. Watts, and R. J. Bartlett, Chem. Phys. Lett. **187**, 21 (1991).