MOLECULAR PHYSICS, 2002, VOL. 100, NO. 11, 1755–1761

Parallelization of analytical Hartree–Fock and density functional theory Hessian calculations. Part I: parallelization of coupled-perturbed Hartree–Fock equations

PRAKASHAN P. KORAMBATH[1], JING KONG[1]*, THOMAS R. FURLANI[2] and MARTIN HEAD-GORDON[3]

[1] Q-Chem Inc., Four Triangle Lane, Suite 160, Export, PA 15632, USA
[2] Center For Computational Research, State University of New York at Buffalo, NY 14260-1800, USA
[3] Department of Chemistry, University of California, Berkeley, CA 94720, USA

(*Received 20 August 2001; accepted 30 October 2001*)

Solving the coupled-perturbed Hartree–Fock (CPHF) equations is the most time consuming part in the analytical computation of second derivatives of the molecular energy with respect to the nuclei. This paper describes a unique parallelization approach for solving the CPHF equations. The computational load is divided by the nuclear perturbations and distributed evenly among the computing nodes. The parallel algorithm is scalable with respect to the size of the molecule, i.e. the larger the molecule, the greater the parallel speedup. The memory storage requirements are also distributed among the processors, with little communication among the processors. The method is implemented in the Q-Chem software package and its performance is discussed. This work represents the first step in a research project to parallelize analytical frequency calculations at Hartree–Fock and density functional theory levels.

1. Introduction

The second derivative of energy with respect to nuclear motion (nuclear Hessian) permits us to calculate efficiently a wide variety of molecular properties, including harmonic vibrational frequencies, equilibrium and transition structures and potential energy surfaces. The Hessian can be computed either analytically via the coupled perturbed Hartree–Fock equations (CPHF) [1] or numerically using a finite difference approach [2]. The finite difference approach is computationally less efficient and also suffers from the disadvantage that accuracy depends on the step size.

Even though the formula to compute analytical second derivatives of the ground state HF energy via the CPHF method [1] was developed a long time ago, its solution was not practical until Pople, Binkley and others in 1979 [3] devised a novel computational method based on an analytical iterative procedure. Early implementations of this method required the transformation of atomic orbital (AO) based electron repulsion integrals (ERI) to molecular orbital (MO) based integrals through a four-index transformation. Osamura *et al.* [4] suggested the solution of CPHF equation in the AO basis and Head-Gordon *et al.* later devised the so-called 'direct' approach that avoided the integral transformation, thereby allowing calculations for larger molecules and basis sets to be realized [5]. A similar formalism has been developed for excited states [6]. With the increase in popularity of density functional theory (DFT), a coupled perturbed Kohn–Sham scheme has also been implemented in several packages, including Gaussian [7], Cadpac [8] and Q-Chem [9].

Ab initio quantum chemistry calculations are computationally demanding because the conventional self-consistent field (SCF) energy and derivative calculations involve the evaluation of electron repulsion integrals (ERI), the cost of which for small molecules scales as $\mathcal{O}(N^4)$ where N is the size of the molecule, and scales as $\mathcal{O}(N^2)$ asymptotically. For this reason, calculations have long been limited to molecules with less than a few hundred atoms. The use of parallel processing has long been pursued as a viable approach to extend the application and scope of various computationally expensive *ab initio* models, because the evaluation of energy and integrals can be partitioned over the computing nodes with minimal communication overhead [10–18].

Most of the parallelization efforts in quantum chemistry are at energy and gradient levels. In the parallel

* Author for correspondence. e-mail: jkong@q-chem.com

Molecular Physics ISSN 0026–8976 print/ISSN 1362–3028 online
http://www.tandf.co.uk/journals
DOI: 10.1080/00268970110109466

implementation in Q-Chem [10] and Gaussian [11], where the Prism [19] algorithm is used to evaluate ERIs, the two-electron integral evaluation is carried out in batches according to the angular momentum and contraction types, and the batches are distributed over the processors, and the partial Fock matrices are formed locally on each processor and summed globally to form the full Fock matrix once all the integrals are calculated. This procedure requires duplication of both the Fock matrix and the density matrix on every node. In NWChem, where the integral evaluation is blocked over the atoms [12], storage arrays are distributed using the global array (GA) tools [20], and the parallel implementation requires accumulation of small blocks of Fock matrix into the distributed matrix. The NWChem package also bypasses the repeated diagonalization of the Fock matrix by using the quadratically convergent SCF approach [21]. Colvin *et al.* [13] extended the parallelization to Fock matrix diagonalization and matrix–matrix multiplications, but at the cost of loss of permutation symmetry for two-electron integrals. In the implementation by Furlani and King [14], the integral evaluation, Fock matrix, and the density matrix steps are distributed. They communicate the density matrices well before they are required by a particular node, with the partial Fock matrix blocks subsequently communicated to the nodes which owns them. Dupuis *et al.* [22] have parallelized the integral evaluation part in the SCF and gradient calculation by distribution of the loop over shell blocks among available nodes, and suggested a similar procedure for the evaluation of analytical second derivative integrals.

The computational cost involved in the evaluation of the analytical Hessian of the Hartree–Fock energy, is far more expensive than that for the SCF energy and gradient calculations in terms of CPU and memory requirements. Conserving memory, while important, was not a critical issue in most of the parallel SCF energy and gradient schemes implemented to date. However, in Hessian calculations, one has to worry about distributing the memory requirements in addition to computational work because the memory usage is $\mathcal{O}(N^3)$, instead of $\mathcal{O}(N^2)$ for energy and gradient calculations. Thus, it is crucial for the success of any parallel CPHF scheme that the scheme is designed to take advantage of the total aggregate memory available on a parallel computer. Windus *et al.* [23] parallelized second derivative calculations by distributing the integral evaluation part while leaving the AO to MO transformation and CPHF iteration part to be carried out serially. The implementation reported by Sosa *et al.* [11] in the Gaussian package extended their SCF parallelization procedure for the integral derivative calculation. However, this implementation duplicates the memory requirement on all the nodes, and therefore limits the application of their algorithm to the maximum memory on a single node. The implementation by Fletcher *et al.* [24] involves parallel transformation of two-electron integrals to an MO basis. The CPHF step involving the contraction of MO integrals with density matrices is parallelized according to the distribution of the MO integrals. The trial vectors are globally summed at the end to solve the set of linear equations serially. This also involves duplication of memory.

In this paper, we present a simple method to parallelize the CPHF iteration that is scalable in CPU time as well as memory for analytical Hessian calculations. This effort is part of an on-going project to parallelize the calculation of the analytical nuclear Hessian at the SCF level for both the HF and the DFT methods, with the HF method being studied first.

2. Strategy of parallelization of CPHF equation

The general theory and equations for solving the CPHF equation through an analytical iterative procedure are well documented in the literature and will not be repeated here [3]. In an analytical Hessian calculation, most of the computational work is involved in solving the CPHF equation to obtain the derivatives of the density matrix or MO coefficients with respect to changes in the nuclear position. The remaining execution time is spent primarily on evaluation of the second derivatives of the ERIs.

The CPHF equation, which is solved iteratively, can be symbolically expressed by

$$(\mathbf{1} - \mathbf{A})\mathbf{B} - \mathbf{B}_0 = \mathbf{0}, \tag{1}$$

where $\mathbf{B}$ represents the derivatives of MOs with respect to the nuclear motions, $\mathbf{B}_0$ contains quantities that do not change during the iteration, and $\mathbf{A}$ is essentially the ERIs. The iteration starts with a set of guess MO derivatives to evaluate $\mathbf{B}$, and then proceeds to calculate $\mathbf{AB}$, which involves contraction of trial density matrix derivatives with the integrals $\mathbf{A}$, followed by the transformation of the resulting matrices from the AO basis to the MO basis. The new $\mathbf{AB}$ vector is then used to construct the trial $\mathbf{B}$ vector for the next iteration by solving a linear equation. This process repeats until $\mathbf{B}$ is converged. The convergence of $\mathbf{B}$ is accelerated as described in the appendix.

The computing cost of each component of calculation is as follows. With modern algorithms such as PRISM [19], the AO integral evaluation scales asymptotically as $\mathcal{O}(N^2)$ with N being the number of atoms in the system. Accordingly, the evaluation of $\mathbf{A}$ scales as $\mathcal{O}(N^2)$. The $\mathbf{AB}$ step is the most expensive step, and scales as $\mathcal{O}(N^3)$. It is composed of contractions of $\mathbf{A}$ with $3 \times N$ matrices, with each of the $3 \times N$ matrices being the derivative of

the density matrix with respect to a particular nuclear motion. Each contraction is similar to the construction of a Fock matrix, which scales as $\mathcal{O}(N^2)$. The evaluation of the next trial vector **B** requires solving a set of linear equations of dimension $3 \times N$, which scales as $\mathcal{O}(N^3)$. The transformation of various matrices from the AO basis to the MO basis scales as $O(N^4)$, but usually takes a negligible amount of time unless the system is very large. There is also a small linear algebra part which scales as $\mathcal{O}(N^3)$. Thus the computational cost required to solve the CPHF equations can be expressed as $aN^2 + bN^3 + cN^3 + dN^4$.

The percentage of CPU time for each component of the CPHF equation as a function of the size of the molecule is contained in table 1. Listed in the table are the three most significant components in a CPHF iteration. As expected the largest component is the contraction of the density matrix with the ERIs. Even though the ERI component is the second largest contributor for small molecules, it becomes less prominent as the molecular size increases. The fourth column is comprised of the MO to AO contraction of the density matrix, the AO to MO transformation of the Coulomb and exchange integrals, and other linear algebra operations.

Based on results presented in table 1, it is obvious that the contraction of the density matrix with the ERIs along with the linear algebra portion of the CPHF iteration needs to be parallelized first. We propose the following simple parallel scheme. (1) The nuclear perturbations are divided evenly among the available computing nodes, and (2) the MO derivatives with respect to the nuclei are computed for the set of nuclei assigned to a given node. The distribution of perturbations can be either grouped by atoms or as an exact division of total perturbations. This is possible because equation (1) can be used to obtain the derivatives of the MOs with respect to any single perturbation. The only difference between the parallel solution and the serial implementation is that the trial vector for the next iteration cycle in the latter is obtained in the space of all the perturbations, whereas in this parallel scheme it is obtained in the space of a subset of the perturbations. While this may result in a slightly slower convergence, it will have no affect on the accuracy. It is important to note that this approach requires no communication overhead because the perturbations on each node are solved independently.

Table 1. CPU timings as percentages for different components of a CPHF iterative calculations using the 6-31G* basis set for a series of alkanes.

Molecule	Contraction	ERI	Linear algebra	Total CPHF
C_8H_{18}	90.35	4.93	4.72	100
$C_{16}H_{34}$	92.27	2.19	5.54	100
$C_{24}H_{50}$	92.06	1.35	6.58	100

In this parallelization scheme, both the contraction step and the linear algebra steps are parallelized, leaving evaluation of the ERIs ($\mathcal{O}(N^2)$) as the only replicated part. The algorithm is scalable as the size of the molecule (N) increases. On p number of processors, the cost for the contraction step AB will be as $\mathcal{O}(N^3/p)$ and the cost for the AO/MO transformation will be as $\mathcal{O}(N^4/p)$. The cost of the other linear algebra operations will also be $\mathcal{O}(N/p)^3$. Only evaluation of the ERIs remains replicated to eliminate the cost involved in communicating the integrals among the processors. The parallel algorithm is said to be scalable if the speedup approaches p as N/p becomes very large. As discussed above, the cost of significant components in solving the CPHF equation can be expressed as: $aN^2 + bN^3 + cN^3 + dN^4$, where a, b, c and d are scaling factors for the contribution from ERI, contraction, linear algebra and AO/MO transformations to the total time. Assuming the number of the parallel processors is p, the speedup is:

$$\text{Speedup} = \frac{(aN^2 + bN^3 + cN^3 + dN^4)}{\left(aN^2 + \frac{bN^3}{p} + c\left(\frac{N}{p}\right)^3 + d\frac{N^4}{p}\right)}. \quad (2)$$

This function will monolithically approach p as N becomes large, showing the scalability of the algorithm with respect to problem size.

The memory requirements for each of the CPHF components scale very similarly to that of the CPU time. The memory bottleneck in a CPHF calculation is the storage of the perturbed MOs and their contractions with the ERIs, which requires $\mathcal{O}(N^3)$ in memory. All other components require $\mathcal{O}(N^2)$ in memory. The cubic scaling of the memory has limited the application of analytical Hessian calculations to small to medium size molecules. Because the storage requirement is proportional to the number of perturbations, the memory requirement for our parallel algorithm will be N^3/p on each node as opposed to N^3 for the serial algorithm and other parallel algorithms [11, 24] .

It should be noted that the above discussion does not take into account the sparsity of the density matrix and its derivatives as in the linear scaling CPSCF algorithm [25]. Q-Chem has not included those algorithms in the analytical second derivative calculations, but the parallel scheme proposed here can be applied equally well.

4. Results and discussions

The above parallel algorithm is implemented in Q-Chem 1.2 [26]. Table 2 lists the average CPU time in seconds for a single CPHF iteration calculation on 8, 16, and 32 processors on an IBM SP machine at NERSC with two POWER 3 processors per node. The data shown in parantheses are the relative speedup with respect to the data in the previous column. Ideally, table 2 should have had the results from processors 1 to 32. Unfortunately, due to the CPU-hour per queue limitation on the IBM SP at NERSC we were not able to carry out our benchmark calculation on a single processor on this machine. Hence the actual speedup shown in table 2 on 16 and 32 processors is compared with respect to the time on 8 processors.

Table 3 compares the speedup obtained using our algorithm with the ideal speedup. The ideal speedup based on the serial content (evaluation of the ERIs) listed in table 1 is calculated using Amdahl's law. According to Amdahl's law, the best speedup one can achieve for a parallel calculation that contains a serial (replicated) component is given by

$$\text{Speedup} = \frac{1}{s + \dfrac{(1-s)}{p}}, \tag{3}$$

where s is the serial fraction. In column 4 of table 3 we scaled the time we obtained on 8 processors using the ideal ratio in column 3 to compute the time that would have been taken on one processor. (So our 1 PE time is actually obtained by multiplying our observed time on 8 PE in column 2 of table 2 with the ideal ratio in column 3 of table 3). We used this scaled 1 PE time to compute the observed speedup on 16 and 32 processors. The actual results compare favourably with what is expected ideally (table 3), demonstrating the scalability of the algorithm with respect to both molecular size and number of processors. As shown in table 1, the serial component attributable to evaluation of the ERIs can be as high as 5% or more for small molecules. Not surprisingly, parallel performance falls off more rapidly for the calculation with the greatest serial content.

We were able to time the algorithm for all the three test molecules starting from a single processor to 32 processors on an SMP cluster of 2 Intel Pentium III processors per node running the Linux operating system at the Center for Computational Research (CCR), Buffalo, facility. Figure 1 plots the speedup for three alkane molecules, C_8H_{18}, $C_{16}H_{34}$ and $C_{24}H_{50}$ on 1, 2, 4, 8, 16, and 32 processors using a 6-31G* basis set. The speedup data used to plot figure 1 are listed in table 4. For a given number of processors, the speedup increases as the size of the molecule increases, once again demonstrating good scalability with respect to the size of the problem. Also shown in figure 1 is a curve for the linear speedup to draw our attention to the apparent superscalability on the Linux clusters. The superscalability evident here is a benefit of the pentium based Linux PCs. Pentium based systems favour smaller

Table 2. Average CPU time in seconds for a single CPHF iteration calculation on 8, 16 and 32 processors for an alkane series using the 6-31G* basis set on IBM-SP[a] machine at NERSC.

	Processors		
Molecule	8	16[b]	32[c]
C_8H_{18}	134	90 (1.49)	63 (1.42)
$C_{16}H_{34}$	1236	770 (1.60)	531 (1.45)
$C_{24}H_{50}$	4557	2687 (1.69)	1691 (1.59)

[a] Each node on this machine has two POWER 3 processors with 200 MHz clock speed and 1024 MB memory. Each processor has an L1 instruction cache of 32 MB, L1 data cache of 64 KB and L2 cache of 4096 KB.
[b] The data shown in parantheses are relative speedup with respect to 8 PE.
[c] The data shown in parantheses are relative speedup with respect to 16 PE.

Table 3. Comparison of the actual speedup of CPHF computation of an alkane series using the 6-31G* basis set on an IBM SP[a] machine at NERSC with ideal speedup expected from Amdahl's law.

		8PE		16 PE		32 PE	
Molecule	Serial	Ideal	Scaled[b]	Ideal	Observed	Ideal	Observed
C_8H_{18}	4.93%	5.94	5.94	9.19	8.84	12.65	12.62
$C_{16}H_{34}$	2.19%	6.93	6.93	12.04	11.12	19.06	16.13
$C_{24}H_{50}$	1.35%	7.30	7.30	13.30	12.38	22.55	19.67

[a] See table 2 for machine details.
[b] The observed result is scaled to match the ideal value because we could not compute the CPU time on a single processor due to time limit per queue on the NERSC machine.

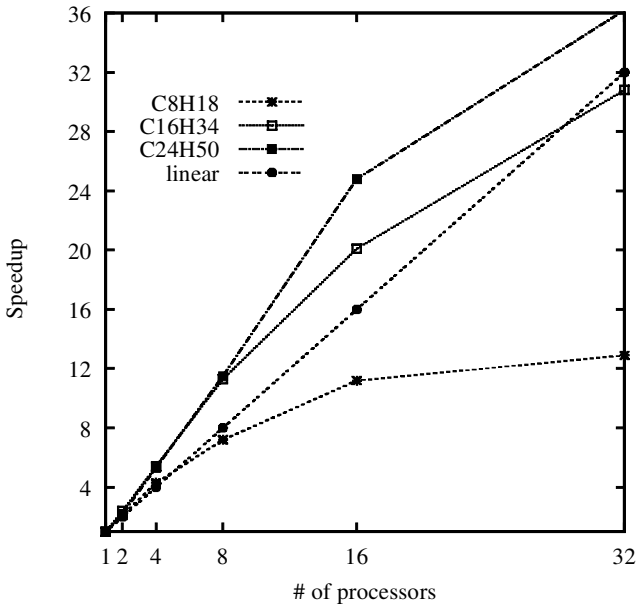

Figure 1. Average CPU time speedup for CPHF calculation of an alkane series using the 6-31G* basis set.

Table 4. Average speedup for the CPHF calculation on 1, 2, 4, 8, 16 and 32 processors for a series of alkanes using the 6-31G* basis set on a Intel Linux PC[a] cluster at CCR, Buffalo.

	Processors					
Molecule	1	2	4	8	16	32
C_8H_{18}	1	2.1	4.3	7.2	11.2	12.9
$C_{16}H_{34}$	1	2.4	5.4	11.3	20.1	30.8
$C_{24}H_{50}$	1	2.2	5.3	11.5	24.8	36.2

[a] Computed on an SMP cluster of 2 Intel Pentium III 1 GHz processors and 1024 MB memory per node running the Linux operating system at the Center for Computational Research (CCR), at Buffalo. Each processor has an instruction cache size of 16 kbytes, data cache size of 16 kbytes and secondary cache size of 256 kbytes.

data sets in memory intensive *ab initio* calculations, due to slower cache.

As mentioned earlier, our parallelization scheme maintains all aspects of the original CPHF implementation in Q-Chem except that the perturbed MOs are obtained in a subspace of perturbations instead of the whole space as in the serial code. Although this change will not affect the final result, as the CPHF equation is valid for each single perturbation, it may result in slower convergence, since each subspace has fewer degrees of freedom. One might be concerned that the number of CPHF iterations for a particular molecule will increase with an increase in the number of processors, since there will be a corresponding decrease in the size of the subspace on each processor. However, from the calculations we have carried out to date, we have found that this is not the case. For example, the alkane C_8H_{18} has only 2 or 3 perturbations on each node when run on 32 processors, yet the number of iterations is still between 6 and 8 for each node (in the serial code there are 6 iterations). The slight increase in number of iterations will not have an adverse effect on the parallel performance, since most of the MO perturbations are already converged in the first 5 or 6 cycles, leaving only a small number of perturbations to be calculated in the extra iterations. One drawback of this algorithm is that the number of perturbations may not be divisible of the number of processors, which could result in load imbalance. However, this effect diminishes as the size of the molecule increases. In this parallel CPHF implementation we did not take advantage of the effect of symmetry, which can significantly reduce the number of perturbation to be solved.

Our approach is different from the implementation of Sosa *et al.* [11]. even though both implementations are based on the direct CPHF method. Sosa *et al.* parallelized the computation of the derivative of the Fock matrix in a similar way to their SCF parallelization. The CPHF equation will not scale well in their approach because only the AO integral calculation and the contraction of the ERI with the density matrix scheme is parallelized, leaving behind a serial component that increases with the molecular size. The difference in scalability between the two methods is evident from table 1. We parallelize both columns 2 and 4 in table 1, which together account for more than 98% of the CPHF time for large molecules. Sosa *et al.* parallelized columns 2 and 3 in table 1, which account for 94% of the total CPU time. If we apply Amdahl's law to compute the best possible speedup using the data for $C_{24}H_{50}$ from the bottom row in table 1, the serial component in our algorithm is 1.4% which gives a speedup of 13 on 16 processors, whereas the serial component in Sosa *et al.*'s algorithm is 6.6%, which gives their algorithm a speedup of only 8 on 16 processors. Thus it is clear that one must pay careful attention to the serial component of a calculation when designing the parallel algorithm. Another important difference between the two is the amount of inter-processor communication. There is no inter-processor communication in our approach. However, in the Sosa *et al.* implementation, communication between the processors is necessary. The implementation by Fletcher *et al.* [24] is different from our method as Fletcher *et al.*'s process involves the transformation of AO integrals to MO integrals. Also the linear algebra part shown in column 4 of table 1 is done serially in their method, similar to Sosa *et al.*'s implementation.

Table 5. Memory usage (in megabytes) of CPHF iterative calculation for a series of alkanes using the 6-31G* basis set on 1 and 2 processors.

	Processors		
Molecule	1	2	Ratio
C_8H_{18}	42.5	22.4	1.90
$C_{16}H_{34}$	282.9	145.2	1.95
$C_{24}H_{50}$	880.5	448.2	1.96

Another important advantage of the parallel implementation presented here is that the $\mathcal{O}(N^3)$ memory requirement is distributed over the processors. In table 5 lists the memory requirements in megabytes on one and two processors for the CPHF iterative part of the calculation. As expected, the memory usage scales very well with respect to molecular size. Good memory scaling is important because poor memory scaling would have limited our application to the maximum available memory on a single node. Thus, this algorithm permits us to do calculations on large molecules by taking advantage of the total aggregate memory of a parallel distributed memory computer. On slow cache systems like Pentium based PCs, the smaller problem size on each processor also accelerates the calculation, as shown by the superlinear scaling in figure 1. The implementation by Sosa *et al.* and Fletcher *et al.* duplicates the memory requirement on each node, and therefore is limited by the memory available on a single node.

4. Conclusion

We have demonstrated effective parallelization of the CPHF calculation by distributing the most time consuming steps, namely the contraction of the density matrix with ERIs and linear algebraic transformations in the iterative direct CPHF calculation over the available processors. This implementation is scalable in both memory and computing time and has no communication overhead. The only replicated part is the evaluation of ERIs, which is repeated in every processor. This replicated part is a small percentage of the total workload, and decreases with the size of the molecule, thereby ensuring the scalability of the algorithm. Calculations on linear alkane chain molecules clearly show good performance.

The authors would like to acknowledge fruitful discussion with Dr Holger Dachsel at ZAM, Juelich, Germany. We also acknowledge generous computer time at NERSC computing facility and the computing facility at the CCR, SUNY, Buffalo. This work was supported by the NIH SBIR Grant No GM 58295-02. MHG also acknowledges support from National Science Foundation (CHE-9981997).

Appendix

The trial vector **B** is obtained in a more or less similar way to that described in [3]. The linear algebraic equation Eq. (1) can be written as

$$\mathbf{B} = \sum_{n=0}^{\infty} \mathbf{A}^n \mathbf{B}_0 \qquad \text{(A 1)}$$

Instead of solving the power series to arrive at **B**, which has slow convergence, an iterative process is used. Starting with a trial vector $\mathbf{V}_0$, $\mathbf{AV}_0$ is obtained through the AO integral computation process and a new vector $\mathbf{V}_1$ is constructed using the following formula, so that $\mathbf{V}_1$ is orthogonal to $\mathbf{V}_0$:

$$\mathbf{V}_{n+1} = \mathbf{A}\mathbf{V}_n - \sum_{i=0}^{n} \frac{\langle \mathbf{V}_i | \mathbf{A} | \mathbf{V}_n \rangle}{\langle \mathbf{V}_i | \mathbf{V}_i \rangle} \mathbf{V}_i \qquad \text{(A 2)}$$

Equation (1) is solved in the space of all $\mathbf{V}_0, \mathbf{V}_1, \ldots,$ etc., and the new trial solution is compared with the previous one. If the difference is significant, then $\mathbf{AV}_{n+1}$ is calculated, and the whole process repeated, until the trial solution does not change anymore. This process is valid for a single perturbation, but in Q-Chem the next trial solution is determined in the space of all perturbations, which provides more degrees of freedom and therefore may result in faster convergence. The parallel algorithm proposed here uses a subset of the total perturbations. In practice, we have found that all the perturbations converge in 6 to 8 iterations, regardless of the size of the subset.

References

[1] Gerrat, J., and Mills, I. M., 1968, *J. chem. Phys.*, **49,** 1719.
[2] Pulay, P., 1969, *Molec. Phys.*, **17,** 197.
[3] Pople, J. A., Krishnan, R., Schelegel, H. B., and Binkley, J. S., 1979, *Intl J. Quantum. Chem. Symp.*, **13,** 225.
[4] Osamura, Y., Yamaguchi, Y., and Schaefer III, H. F., 1982, *J. chem. Phys.*, **77,** 383.
[5] Frisch, M. J., Head-Gordon, M., and Pople, J. A., 1990, *Chem. Phys.*, **141,** 189.
[6] Maurice, D., and Head-Gordon, M., 1999, *Molec. Phys.*, **96,** 1533.
[7] Johnson, B. G., and Frisch, M. J., 1994, *J. chem. Phys.*, **100,** 7429.
[8] Handy, N. C., Tozer, D. J., Laming, G. L., Murray, C. W., and Amos, R. D. 1993, *Israel J. Chem.*, **33,** 331.
[9] Kong, J., White, C. A., Krylov, A. I., Sherrill, D., Adamson, R. D., Furlani, T. F., Lee, M. S., Lee, A. M., Gwaltney, S. R., Adams, T. R., Ochsenfeld, C., Gilbert, A. T. B., Kedziora, G. S., Rassolv, V. A., Maurice, D. R., Nair, N., Shao, Y., Besley, N. A., Maslen, P. E., Dombroski, J. P., Dachsel, H., Zhang, W., Korambath, P. P., Baker, J., Byrd,

E. F. C., Voorhis, T. V., Oumi, M., Hirata, S., Hsu, C. P., Ishikawa, N., Florian, J., Warshel, A., Johnson, B. G., Gill, P. M. W., Head-Gordon, M., and Pople, J. A., 2000, Q-Chem 2.0: a high-performance *ab initio* electronic structure program package, *J. Comput. Chem.*, **21,** 1532.

[10] Furlani, T. R., Kong, J., and Gill, P. M. W., 2000, *Comput. Phys. Commun.*, **128,** 170.

[11] Sosa, C. P., Ochterski, J., Carpenter, J., and Frisch, M. J., 1998. *J. Comput. Chem.*, **19,** 1053.

[12] Bernholdt, D. E., Apra, E., Fruchtl, H. A., Guest, M. F., Harrison, R. J., Kendall, R. A., Kutteh, R. A., Long, X., Nicholas, J. B., Nichols, J. A., Taylor, H. L., Wong, A. T., Fann, G. I., Littlefield, R. J., and Nieplocha., J., 1995, *Intl J. Quantum Chem. Symp.*, **29,** 475.

[13] Colvin, M. E., Janssen, C. L., Whiteside, R. A., and Tong, C. H., 1993, *Theoret. Chim. Acta*, **84,** 301.

[14] Furlani, T. R., and King, H. F., 1995, *J. Comput. Chem.*, **16,** 91.

[15] Harrison, R. J., and Shepard, R., 1994, *Ann. Rev. phys. Chem.*, **45,** 623.

[16] Guest, M. F., Harrison, R. J., van Lenthe, J. H., and van Corler, L. C. H., 1987, *Theoret. Chim. Acta*, **71,** 117.

[17] Bernholdt, D. E., and Harrison, R. J., 1996, *Chem. Phys. Lett.*, **250,** 477.

[18] Dachsel, H., Lischka, H., Shepard, R., Nieplocha, J., and Harrison, R., 1997, *J. Comput. Chem.*, **18,** 430.

[19] Gill, P. M. W., 1994, *Adv. Quantum Chem.*, **25,** 141.

[20] Nieplocha, J., Harrison, R. J., and Littlefield, R. J., 1994, Global Arrays: A portable shared memory programming model for distributed computers (Washington, DC: IEEE Computer Society Press).

[21] Bacskay, G. B., 1982, *Chem. Phys.*, **61,** 385.

[22] Dupuis, M., and Watt, J. D., 1987, *Theoret. Chim. Acta*, **71,** 91.

[23] Windus, T. L., Schmidt, M. W., and Gordon, M. S., 1993, *Chem. Phys. Lett.*, **216,** 375.

[24] Fletcher, G. D., Rendell, A. P., and Sherwood, P., 1997, *Molec. Phys.*, **91,** 431.

[25] Ochsenfeld, C., and Head-Gordon, M., 1997, *Chem. Phys. Lett.*, **270,** 399.

[26] White, C. A., Kong, J., Maurice, D. R., Adams, T. R., Baker, J., Challacombe, M., Schwegler, E., Dombroski, J. P., Ochsenfeld, C., Oumi, M., Furlani, T. R., Florian, J., Adamson, R. D., Nair, N., Lee, A. M., Ishikawa, N., Graham, R. L., Warshel, A., Johnson, B. G., Gill, P. M. W., and Head-Gordon, M., 1998, Q-Chem, Version 1.2 (Pittsburgh: PA: Q-Chem, Inc.).