Semidirect MP2 Gradient Evaluation on Workstation Computers: The MPGRAD Program

Frank Haase[1,2] and Reinhart Ahlrichs[1]*

Universität Karlsruhe, [1]Lehrstuhl für Theoretische Chemie, Institut für Physikalische Chemie, D-7500 Karlsruhe, Germany, and [2]Max-Planck-Gesellschaft, Arbeitsgruppe Quantenchemie an der Humboldt Universität Berlin, D-1199 Berlin, Germany

Received 12 October 1992; accepted 27 February 1993

A semidirect implementation of the closed-shell MP2 gradient for efficient use on workstation computers is presented. The approach is based on the algorithm proposed by Frisch and coworkers but includes several modifications to reduce disk storage requirements and exploits nonabelian point group symmetry. The performance of the resulting program MPGRAD (BIOSYM Corp., San Diego, CA) is demonstrated in applications to the molecules $[AlSi(CH_3)_3]_4$ and ferrocene. The largest calculation involved 492 basis functions and was carried out on IBM RS/6000 workstations with memory sizes of 32 and 128 Mb. The ratio of CPU to wallclock time exceeds 90% in all typical applications.

INTRODUCTION

The direct self-consistent field (SCF) method has proven its efficiency[1,2] and has become standard especially for applications on workstation computers. Because a direct SCF calculation requires the storage of only a few matrices (of size N^2 with N as the number of basis functions), applications are basically CPU bound.

Attention has now focused on the extension of direct approaches to correlated methods. Because the majority of correlated methods are formulated in the molecular orbital (MO) basis, they require an integral transformation from the atomic orbital basis (AO) to the MO basis and hence need to process and store various types of MO integrals. Memory and disk storage requirements are therefore much larger than in SCF calculations, even for a second-order Møller–Plesset perturbation theory (MP2) treatment,[3] the simplest approach to treat electron correlation effects in a systematic manner. Because the computational requirements of correlated methods are at least N^5, correlated calculations on large molecules still represent a challenge. Direct approaches to correlated methods were initially suggested by Taylor[4] and later implemented for MP2 energy calculations by Head-Gordon et al.,[5] Sæbø et al.,[6] and in the TURBOMOLE program package.[7] Recently, direct MP2 energy calculations employing efficient integral prescreening techniques together with simple schemes to exploit symmetry of molecular systems involving more than 800 contracted basis functions were reported.[8]

The MP2 gradient expression and a conventional disk-based implementation was first reported by Pople et al.[9] A first direct approach to the MP2 gradient method was recently presented by Frisch and coworkers.[10] In a second article, the algorithm was extended to a semidirect approach that utilizes disk space at the expense of I/O.[11] Because external disks with more than 2-Gb capacity and access times of a few milliseconds at transfer rates of up to two Mb/s are now available at moderate prices, a flexible semidirect approach to the MP2 gradient method offers advantages that should be exploited. Together with tools for integral prescreening and exploitation of symmetry, this approach should allow applications to systems with several hundred basis functions.

It is our objective to demonstrate the performance of our semidirect MP2 gradient implementation for closed-shell systems on workstation computers. In the next section, we briefly describe the special features of our implementation, which is based on the semidirect approach proposed by Frisch et al.[11] but is further extended to reduce disk space requirements and in addition exploits nonabelian point group symmetry. Timings of two demonstrative applications to large and symmetric molecules are given in the last section.

IMPLEMENTATION

The design of the present algorithm was derived from the requirement that applications employing a few hundred basis functions could be carried out on workstations with 32–128 Mb memory and 1 Gb to

*Author to whom all correspondence should be addressed.

CCC 0192-8651/93/080907-06

a few Gb disk space. This led to the following specifications:

1. Main memory requirement should scale with N^2.
2. Disk space requirements should be adjustable.
3. I/O is admissible as long as this overlaps sufficiently with CPU.
4. Use of all discrete point group symmetries.

These goals can be achieved with the algorithm of Frisch et al.[11]—with some modifications—if combined with tools available within TURBOMOLE. To meet requirements 1 and 2, it is necessary to perform certain steps repeatedly. This especially requires a multiple-pass strategy as suggested in ref. 11: Occupied MOs are divided by the number of passes yielding batches of size B that are processed one batch at a time.

In the following, the formulation of the MP2 gradient as given by Handy et al.[12] will be adopted. Indices $i, j, k\ldots$ refer to occupied MOs; $a, b, c\ldots$ denote virtual MOs; $p, q, r\ldots$ signify both virtual and occupied MOs; and $\mu, \nu, \kappa\ldots$ are basis functions. In the restricted Hartree–Fock (RHF) closed-shell case, the MP2 gradient can be succinctly written as[12]

$$E^{(x)}_{\mathrm{MP2}} = \sum_{pq} \mathbf{Y}_{pq} h^x_{pq} + \sum_{pq} \mathbf{W}_{pq} S^x_{pq} + \sum_{pqrs} \Gamma_{pqrs} (pq|rs)^x \quad (1)$$

where (x) denotes differentiation with respect to a Cartesian coordinate. The matrix $\mathbf{Y}$ is the correlated density (also known as relaxed or response density) consisting of three blocks: the occupied–occupied block V_{ij}, the virtual–virtual block V_{ab}, and the occupied–virtual block Z_{ai}, which is the solution of the Z-vector equation.[13] The latter is a coupled perturbed Hartree-Fock (CPHF) type equation with the MP2 lagrangian L_{ai} as the corresponding right-hand side. $\mathbf{W}$ is an intermediate matrix similar to the energy-weighted density of the SCF gradient expression. The final term contains the two-particle density matrix Γ, which consists of a nonseparable part $\mathbf{\Gamma}^{\mathbf{NS}}$ and a separable part $\mathbf{\Gamma}^{\mathbf{S}}$ and is defined through

$$\sum_{pqrs} \Gamma_{pqrs} (pq|rs)^x = \underbrace{-2 \sum_{ijab} t^{ab}_{ij} (ia|jb)^x}_{\mathbf{\Gamma}^{\mathbf{NS}}} + \underbrace{\sum_{pq} \mathbf{Y}_{pq} \sum_{k} [2(pq|kk)^x - (pk|qk)^x]}_{\mathbf{\Gamma}^{\mathbf{S}}} \quad (2)$$

where

$$t^{ab}_{ij} = \frac{2(ia|jb) - (ib|ja)}{\epsilon_a + \epsilon_b - \epsilon_i - \epsilon_j} \quad (3)$$

An efficient way to obtain the MP2 gradient is to calculate the matrices $\mathbf{Y}$, $\mathbf{W}$ and the double-substitution amplitudes t^{ab}_{ij} in the MO representation, then transform them back to the AO basis, and finally contract them with derivative AO integrals. The present algorithm to evaluate these terms is sketched in Algorithm 1.

Algorithm 1. Structure of the present semidirect implementation. Loop over batches B.

(1) $(\nu\mu|\kappa\lambda) \rightarrow (\nu\mu|ia),(\nu\mu|ij)$ and write; all $i \in B$
  loop over $i \in B$
    (2) read $(\nu\mu|ia),(\nu\mu|ij) \rightarrow (jb|ia),(ab|ij),(ak|ij)$ and write
    (3) read $(ia|jb) \rightarrow t^{ab}_{ij}$ and write
    (4) increment V_{ij}, V_{ab}, $\mathbf{W}_{ij}$, $\mathbf{W}_{ab}$ and L_{ai}
    (5) read $(\nu\mu|ia) \rightarrow (cb|ia)$ and increment L_{ai} on the fly
    (6) read $t^{ab}_{ij} \rightarrow t^{a\kappa}_{i\lambda}$ and write
    (7) read $t^{a\kappa}_{i\lambda} \rightarrow \mathbf{\Gamma}^{\mathbf{NS}}_{\mu\nu,\kappa\lambda}$ and contract with $(\mu\nu|\kappa\lambda)^x$
  end loop i
end loop B
(8) complete L_{ai} and solve Z-vector equation for Z_{ai}
(9) contract $\mathbf{\Gamma}^{\mathbf{S}}_{\mu\nu,\kappa\lambda}$ with $(\mu\nu|\kappa\lambda)^x$ and evaluate $\mathbf{Y}_{\mu\nu} h^x_{\mu\nu}$, $\mathbf{W}_{\mu\nu}$ $S^x_{\mu\nu}$

Steps 1 and 2, the transformation $(\nu\mu|ia) \rightarrow (cb|ia)$ in step 5, the solution of the Z-vector equation, and step 9 are based on the MP2, CPHF, and SCF gradient routines of the TURBOMOLE program with appropriate modifications. This applies especially to the symmetry handling and the use of integral prescreening techniques.[2,14]

To further reduce disk storage demands and hence I/O, the semidirect approach of Frisch et al.[11] was modified in two ways. The maximum disk requirement of their algorithm is BN^3 and appears at stage 2 of Algorithm 1. The three external integrals over MOs occupy by far the largest part of this disk space, that is, $\frac{1}{2}BV^3$ (V is the number of virtual MOs). Because they are needed only once, to increment the MP2 langrangian L_{ai}, they are generated on the fly and never stored in our algorithm. In contrast to ref. 11, the second half-transformation, to compute three external MO integrals (step 2), is deferred. Only when double-substitution amplitudes t^{ab}_{ij} are available, the second half-transformation is carried out together with the on-the-fly contraction of the resulting three externals with amplitudes (step 5). The basic loop structure for this task is given below ($\mathbf{C}$ is the MO matrix).

Algorithm 2. Loop structure for step 5 of Algorithm 1.

```
loop i ∈ B, all a
    read (μν|ia) all μ, ν
    read t^ba_ji all j, b
        loop b
            loop (νμ) pairs
                (μb|ia) ← (μb|ia) + C_νb (μν|ia)
                (νb|ia) ← (νb|ia) + C_μb (μν|ia)
            end loop (νμ) pairs
            loop c ≤ b
                loop μ
                    (cb|ia) ← (cb|ia) + C_μc (μb|ia)
                end loop μ
                loop k
                    L(c, k) ← L(c, k) − 4t^ba_ki (cb|ia)
                end loop k
            end loop c
        end loop b
end loop i,a
```

In this way, the maximum disk requirement of the present algorithm is shifted to stage 1 of Algorithm 1 and is reduced by a factor of two ($\frac{1}{2}BN^3$) compared to the original algorithm[11] at the only extra cost of reading the half-transformed integrals of type $(ia|\mu\nu)$ twice and an additional reading of the amplitudes t_{ij}^{ab}. But, this is not a serious disadvantage because I/O sufficiently overlaps with the N^5 arithmetic of the transformation. Besides this reduction due to the above-mentioned modification, a further saving in disk space occurs by using the full symmetry point group, as will be explained below.

The second modification concerns the back transformation of the double-substitution amplitudes from the MO to AO basis (step 6 of Algorithm 1), yielding the nonseparable part Γ^{NS} of the MP2 two-particle density, which is subsequently contracted with derivative integrals (step 7). We organized this back transformation in two half-transformations similar to the integral transformation.[7] The second half-transformation is absorbed into the subsequent contraction of the two-particle density with derivative integrals. This leads to an additional N^5 step at this stage compared to the algorithm of ref. 11 but has the following advantage. It requires only a single transposition of an array of size $\frac{1}{2}BVN^2$ (half-transformed amplitudes $t_{i\lambda}^{a\kappa}$). In the N^2 algorithm of ref. 11, an additional reordering of an array of size $\frac{1}{2}BN^3$ is necessary, which causes additional I/O.

In the following, we comment briefly on I/O, which may have a poor performance on workstations, especially for direct access. The wallclock time of a job can then be much larger than the CPU time. Because out-of-core sort steps are unavoidable in algorithms with N^2 memory requirement, the efficiency of I/O is of great importance. In our implementation, I/O is strictly N^4 dependent. When sorting arrays, no second copy or indices must be held on disk. The data that have to be sorted are written once and then read several times if necessary in a certain order (like in the case of the exchange-type MO integrals or double-substitution amplitudes).

Besides the usual double buffering of all I/O operations, some help comes from the dynamic memory allocation used throughout in the program. It immediately frees previously allocated memory if it is no longer in use. The main memory thus made available to the operating system (UNIX) is used as (additional) internal buffer that facilitates overlapping of CPU and channel activities. Finally, for the largest files—the half-transformed integrals and amplitudes—the I/O is coded in C, which is slightly faster than FORTRAN.

The scheme implemented for exploiting molecular point group symmetry is based on the formalism used throughout the TURBOMOLE program system.[7] It is due to Häser and is outlined elsewhere in detail.[15] The algorithm applies projection operators that project quadruples of MO onto their totally symmetric part and in this way allows to identify those MO integrals or amplitudes that are zero or equal to others for symmetry reasons. Further, the factorization of such projectors provides a means of packing and unpacking the above-mentioned quantities, leading to a symmetry speed-up and a reduction of disk space. For quantities in the MO basis, the disk requirement is reduced by a factor of g^{-1} if g is the order of the molecular point group. However, for half-transformed quantities like the integrals $(\nu\mu|ia)$ and amplitudes $t_{i\lambda}^{a\kappa}$ the saving factor is g^{-1} *at best* and may deteriorate if atoms are located on symmetry elements as in planar and linear molecules.

After the submission of the article, a revision of the present algorithm was completed concerning the solution of the Z-vector equation (step 8 of Algorithm 1). Although the original MO-based algorithm is fast, it requires the evaluation and storage of an additional type of half-transformed integrals $(\nu\mu|ij)$ in stage 1 and Coulomb-type MO integrals $(ab|ij)$ in stage 2. The algorithm was substituted by the direct inversion of iterative subspace (DIIS) method.[9] The Z-vector equation is now solved directly in the AO basis,[16] similar to a direct SCF calculation. While the overall computation time is increased marginally, disk storage requirements are reduced significantly for larger molecules at least (see Table III).

REPRESENTATIVE TIMINGS

In Table I, timings and disk space requirements for two symmetric molecules are listed that illustrate the savings due to symmetry. The CPU times for

Table I. Performance of MP2 gradient calculations using different point groups. Times refer to an IBM RS6000/320H unless otherwise indicated.

	$(AlH)_6$			$N_4(CH_2)_6$		
Number of basis functions[a]	138			210		
Point group	D_{3d}	D_3	C_{2h}	T_d	D_{2d}	D_2
CPU time[b] (min)	45	80	130[c]	154	373	688 (737)[d]
CPU/wallclock (%)	90	90	94	92	93	96
Disk space (Mb)	64	103	180[c]	166	353	670 (286)[d]

[a]A basis set of SVP type[17] was employed.

[b]All calculations performed with a single pass if not otherwise indicated.

[c]For comparison, Gaussian 92[18] calculations were carried out under the same conditions (semidirect N^2 algorithm, 20-Mb memory, full correlation, same symmetry and number of passes). The following keywords were specified: %MEM = 2621440; MP2 = (SEMIDIRECT,FULL, MAXDISK = 227840000); SCF = (DIRECT) GEN FORCE. The CPU time of the Gaussian 92 MP2 gradient modules (links 906, 1002, 701, 702, 703, 716) was 317 min and the disk requirement (rwf-file) was 1259 Mb.

[d]CPU time and disk requirement of an MPGRAD calculation on an IBM RS6000/350 with three passes given in parentheses. The same calculation with the Gaussian 92 MP2 gradient required 1456 min CPU time and 1735-Mb disk space.

$(AlH)_6$ scale as 1.6 and 2.9 if symmetry is increased from C_{2h} to D_3 and D_{3d}, in close agreement with the ratio of the number of symmetry operations (1.5 and 3, respectively). In the case of $N_4(CH_2)_6$, the situation is less satisfactory but the values of 1.8 and 4.5 do not differ much from the corresponding theoretical factors 2 and 6. The savings in disk space are similar to those in CPU times. It should be mentioned that for those terms of the gradient expression that are fully evaluated in the MO representation (perturbed density Y) symmetry savings by a factor of $1/g^2$ occur. But, this hardly affects the total CPU time because in the dominating steps (steps 1, 2, 5, and 7 of Algorithm 1) the symmetry speed-up scales usu-1) the symmetry speed-up scales usually as $1/g$.

A sensitive indicator for the efficiency of the I/O implementation is the ratio of the CPU to wallclock time of a calculation. As can be seen from Table I, it exceeds 90% in all cases. This shows that already for such small calculations (note that I/O scales as N^4 and arithmetic as N^5) the program performs well on a workstation computer.

In the footnotes of Table I, we also document the performance of Gaussian 92[18] in comparison to MPGRAD as requested by one of the referees. In either case, the semidirect N^2 algorithm (which would be selected automatically under the given memory restrictions) required about twice the CPU time although a low (abelian) symmetry group was used. Disk space requirements were roughly six times larger for the two test cases considered.

MP2 gradient calculations on $[AlSi(CH_3)_3]_4$ and ferrocene were carried out to assess the performance of MPGRAD for large calculations. The technical data are summarized in Table II. The former molecule consists of 56 atoms, is highly symmetric (point group T_d), and thus should provide a good test case, especially for the efficiency of the symmetry scheme implemented. A salient feature of the semidirect algorithm is its capability of adapting to the respective resources of a computer. This has been carried to the extreme with the first calculation in Table II, where only 32 Mb memory and a mass storage capacity of no more than 1.3 Gb were made available. Three passes and hence three integral and four derivative evaluations were necessary under these conditions. The main memory requirement was 45 Mb and therefore significant paging to virtual memory was to be expected. However, the CPU performance of 83% is satisfactory under the given circumstances.

Table II. Technical data of MP2 gradient calculations on $[AlSi(CH_3)_3]_4$ and ferrocene.

	$[AlSi(CH_3)_3]_4$[a]	$Fe(C_5H_5)_2$[b]
Number of basis functions	492[c]	483[d]
Point group	T_d	D_{5h}
Number of passes	3	2
CPU time (min)	5061	3342
CPU/wallclock (%)	83	92
Disk space (Mb)	1221	1235

[a]Timings refer to an IBM RS6000/320H (32-Mb memory).
[b]Timings refer to an IBM RS6000/550 (128-Mb memory).
[c]A basis set of SVP type[17] was employed.
[d]Basis set of Fe, [8*s*7*p*4*d*2*f*]; C, [6*s*4*p*3*d*]; H, [3*s*2*p*].

As a second application, an MP2 gradient calculation on ferrocene was performed. This molecule was recently investigated by Park and Almlöf[19] using the modified coupled-pair functional (MCPF) and MP2 methods. Because the authors had to resort to single-point calculations on the iron—cyclopentadienyl distance and employed small basis sets, it was our goal to show that the analytic MP2 optimization with a sufficiently large basis set is no longer prohibitive with the present program. A detailed investigation of ferrocene on the MP2 level including the full optimization will be presented in a separate article. Because of the memory requirement of 94 Mb, we used a 128-Mb memory workstation. Thus problems due to paging were avoided and it was possible to assess the I/O performance. The CPU/wallclock time ratio exceeded 90%, as for the smaller calculations, although a much larger data transfer to disk was necessary.

In a series of calculations with different numbers of passes, we wanted to examine what one has to pay for a significant reduction of the disk storage requirement. From inspection of Algorithm 1, it follows that each additional pass through all N^5-dependent terms inevitably leads to an additional integral and derivative evaluation. Because in the present implementation the integral and derivative evaluation scales with the order of the molecular point group, two cases were considered: one with high symmetry and one with moderate symmetry. Again, $[AlSi(CH_3)_3]_4$ in T_d symmetry was chosen together with the hexamethylenetetramine molecule in D_2 symmetry. To get a more detailed insight into the individual steps of the algorithm, Table III contains additional timings of all steps of Algorithm 1. Let us first consider the data for the calculation on $[AlSi(CH_3)_3]_4$. Comparing the total CPU times of the three different calculations, we see a surprisingly small increase when increasing the number of passes. A calculation with three passes (two passes) requires 66% (44%) less disk space but only 3.4% (2.3%) more CPU time than a run with a single pass. This additional expense is mostly caused by the integral derivative generation, while additional integral computations are less expensive. This is due to the high symmetry, which leads to a speed-up of the integral and derivative computation by a factor of 24. Thus, these two tasks have only a minor influence on the total CPU time. The situation changes for lower symmetry. In the case of $N_4(CH_2)_6$, the savings in disk space when increasing the number of passes from one to eight (four) are 68% (58%), but now one must accept an increase in CPU time of 188% (80%). All that can be gained by symmetry is a factor of 2 (if point group D_2 is imposed), and therefore integral

Table III. Timings (min) of individual steps of MP2 gradient calculations on $[AlSi(CH_3)_3]_4$ and $N_4(CH_2)_6$ with different numbers of passes.

	$[AlSi(CH_3)_3]_4$[a]			$N_4(CH_2)_6$[b]		
Number of basis functions[c]	492			210		
Point group	T_d			D_2		
Number of passes	1	2	3	1	4	8
First half-transformation, step 1	174	194	210	97	216	379
Second half-transformation, step 2	654	644	630	31	31	31
Increment **Y, W, L,** steps 3 and 4	163	164	160	17	17	18
Increment **L** with $(cb\|ia)$, step 5	897	892	878	67	69	69
t_{ij}^{ab} backtransformation, step 6	322	320	317	35	36	36
$\Gamma^{NS}_{\mu\nu,\kappa\lambda}$, $(\mu\nu\|\kappa\lambda)^r$ evaluation, step 7	273	332	382	289	707	1280
Solution of Z-vector equation, step 8	62	63	62	10	11	11
$\Gamma^{S}_{\mu\nu,\kappa\lambda}$, $(\mu\nu\|\kappa\lambda)^r$ evaluation, step 9	35	35	35	107	111	119
Total CPU time	2643	2703	2732	688	1233	1978
CPU/wallclock (%)	93	95	92	96	92	91
Disk space (Mb)	3598	2007	1221	670	283	215
Disk space (Mb)[d]	2680	1357	912	620	246	180

[a]Timings refer to an IBM RS6000/550 (128-Mb memory).
[b]Timings refer to an IBM RS6000/320H (32-Mb memory).
[c]A basis set of SVP type[17] was employed.
[d]Disk space requirement of the current version as explained in text.

and derivative computation becomes much more expensive and, further, more passes are necessary to achieve a disk space reduction comparable to the high-symmetry calculations. Besides this, steps 1 and 7 of Algorithm 1 are now dominant because hexamethylenetetramine has only 38 occupied MOs (108 occupied MOs were involved in the $[AlSi(CH_3)_3]_4$ calculation), which considerably reduces the computational effort for N^5-dependent steps.

CONCLUSION

We have shown that a semidirect implementation of the MP2 gradient method together with an efficient scheme to exploit molecular symmetry and techniques for integral prescreening provides a powerful tool in examining structures of large and symmetric molecules at the MP2 level of theory. Applications employing a few hundred basis functions are in this way feasible even on computers with limited resources. In cases of high symmetry, the extra expense in CPU time to drastically reduce the disk storage requirements of a calculation is small. For cases with lower symmetry, this is no longer true. However, although such calculations may require CPU times of several days the CPU/wallclock ratio still exceeds 90%.

The authors thank J. Gauss and M. Häser for several valuable hints given during the development of the program. They are grateful to H. Horn, C. Kölmel, and H. Weiss for many helpful discussions. Financial support by the Stipendienstiftung of the Fonds der Chemischen Industrie via a postdoctorate stipendium for one of us (F.H.) is gratefully acknowledged. Finally, F.H. thanks the TURBOMOLE team for the hospitality and exuberant but stimulating atmosphere during his stay in Karlsruhe.

REFERENCES

1. J. Almlöf, K. Fægri Jr., and K. Korsell, *J. Comp. Chem.*, **3,** 385 (1982); D. Cremer and J. Gauss, *J. Comp. Chem.*, **7,** 274 (1986).
2. M. Häser and R. Ahlrichs, *J. Comp. Chem.*, **10,** 104 (1989).
3. C. Møller and M.S. Plesset, *Phys. Rev.*, **46,** 618 (1934).
4. P.R. Taylor, *Int. J. Quantum Chem.*, **31,** 521 (1987).
5. M. Head-Gordon, J.A. Pople, and M.J. Frisch, *Chem. Phys. Lett.*, **153,** 281 (1988).
6. S. Sæbø and J. Almlöf, *Chem. Phys. Lett.*, **154,** 83 (1989).
7. R. Ahlrichs, M. Bär, M. Häser, H. Horn, and C. Kölmel, *Chem. Phys. Lett.*, **162,** 165 (1989).
8. V. Parasuk, J. Almlöf, and M.W. Feyereisen, *J. Am. Chem. Soc.*, **113,** 1049 (1991); M. Häser, J. Almlöf, and G.E. Scuseria, *Chem. Phys. Lett.*, **181,** 497 (1991); G.E. Scuseria, *Chem. Phys. Lett.*, **176,** 423 (1991).
9. J.A. Pople, R. Krishnan, H.B. Schlegel, and J.S. Binkley, *Int. J. Quantum Chem. Symp.*, **13,** 225 (1979).
10. M.J. Frisch, M. Head-Gordon, and J.A. Pople, *Chem. Phys. Lett.*, **166,** 275 (1990).
11. M.J. Frisch, M. Head-Gordon, and J.A. Pople, *Chem. Phys. Lett.*, **166,** 281 (1990).
12. N.C. Handy, R.D. Amos, J.F. Gaw, J.E. Rice, and E.D. Simandiras, *Chem. Phys. Lett.*, **120,** 151 (1985); E.D. Simandiras, R.D. Amos, and N.C. Handy, *Chem. Phys. Lett.*, **114,** 9 (1987).
13. N.C. Handy and H.F. Schaefer, *J. Chem. Phys.*, **81,** 5031 (1984).
14. H. Horn, H. Weiss, M. Häser, M. Ehrig, and R. Ahlrichs, *J. Comp. Chem.*, **12,** 1058 (1991).
15. M. Häser, Dissertation, Universität Karlsruhe, 1990; M. Häser, J. Almlöf, and M.W. Feyereisen, *Theor. Chim. Acta*, **79,** 115 (1991); M. Häser, *J. Chem. Phys.*, **95,** 8259 (1991).

16. Y. Osamura, Y. Yamaguchi, and H.F. Schaefer III, *J. Chem. Phys.*, **77,** 383 (1982).
17. A. Schäfer, H. Horn, and R. Ahlrichs, *J. Chem. Phys.*, **97,** 2571 (1992).
18. M.J. Frisch, G.W. Trucks, M. Head-Gordon, P.M.W. Gill, M.W. Wong, J.B. Foresman, B.G. Johnson, H.B. Schlegel, M.A. Robb, E.S. Replogle, R. Gomperts, J.L. Andres, K. Raghavachari, J.S. Binkley, C. Gonzalez, R.L. Martin, D.J. Fox, D.J. Defrees, J. Baker, J.J.P. Stewart, and J.A. Pople, Gaussian 92, revision C, Gaussian, Inc., Pittsburgh, PA, 1992.
19. C. Park and J. Almlöf, *J. Chem. Phys.*, **95,** 1829 (1991).