Computer Physics Communications 128 (2000) 170–177

Computer Physics Communications

www.elsevier.nl/locate/cpc

Parallelization of SCF calculations within Q-Chem ☆

Thomas R. Furlani [a,*], Jing Kong [b], Peter M.W. Gill [c]

[a] *Center for Computational Research, University at Buffalo, Buffalo, NY 14260-1800, USA*
[b] *Q-Chem, Inc., Four Triangle Lane, Export, PA 15632, USA*
[c] *School of Chemistry, University of Nottingham, Nottingham NG7 2RD, UK*

Abstract

We have incorporated MPI based parallelism with *dynamic load balance* into the Hartree–Fock and DFT modules of Q-Chem. A series of benchmark calculations consisting of both *single point energy and gradient calculations* were carried out to gauge the performance of the parallel modules. Calculations were carried out on two different parallel computers, namely a shared memory Silicon Graphics Origin2000 and a distributed memory Cray T3E, to show the flexibility of the code and demonstrate the great utility of MPI. Scalability for the DFT and Hartree–Fock modules is demonstrated for up to 64 processors. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Parallel SCF; Parallel DFT; Q-Chem; Parallel Hartree–Fock

1. Introduction

Quantum chemical methods (i.e. those which explore and explain chemical phenomena directly from rigorous quantum physics) have established themselves as valuable tools in the arsenals of many chemists and biochemists. Indeed, the award of the 1998 Nobel Prize in Chemistry to Professors John Pople and Walter Kohn for their seminal contributions to the development of quantum chemistry serves to underscore the significance of quantum chemical methods for modeling molecular problems. The explosive growth in the application of such methods can be traced to remarkable advances in both computer hardware and software. In addition to the dramatic improvements in algorithms that have taken place in recent years, modern single-processor computers are now orders of magnitude faster than their predecessors of only a decade ago. Together, these improvements have allowed useful calculations on molecular systems containing tens to hundreds of atoms to be routinely carried out. It is widely recognized, however, that a significant further increase in the speed of modern supercomputers is more likely to come from increasing the number of processors than from increasing processor speeds. Even the remarkable advances of the past decade in the development of new computer hardware have not resulted in an order of magnitude improvement in CPU speed over that of the early CRAY with its 12 nanosecond clock. In fact, present technology is already close to fundamental limits. On the other hand, it is entirely feasible and cost effective to build machines with hundreds of inexpensive, air cooled scalar or vector processors that are connected together in a shared or distributed memory architecture, and which function autonomously and asynchronously. These parallel computers hold great promise for large scale scientific computing. Indeed, the availability of

☆ This paper is published as part of a thematic issue on Parallel Computing in Chemical Physics.

* Corresponding author. E-mail: furlani@ccr.buffalo.edu.

0010-4655/00/$ – see front matter
PII: S0010-4655(00)00059-X

commercial parallel computers in the late 1980's signaled the beginning of the long awaited "revolution" in scientific computing and spurred code development efforts in many scientific fields, including computational chemistry [1–14]. For example, quantum chemistry codes such as Q-Chem [14], GAMESS [9], Jaguar [8] and Gaussian [13] have been modified to function in a variety of parallel computing environments.

Here we report the performance of the density functional theory (DFT) and Hartree–Fock modules of the *ab initio* quantum chemistry code Q-Chem on both an SGI/CRAY T3E and an SGI Origin2000. These computers were selected because they represent the two most common parallel computer architectures in existence today, namely distributed memory (T3E) and shared memory (Origin2000). The parallel performance of Q-Chem, as measured by the ratio of the execution time on a single processor to that on multiple processors (speed-up), was studied for both *single point energy and gradient calculations* on up to 64 processors of each machine.

The following section contains a detailed description of Q-Chem, including the linear scaling DFT and Hartree–Fock modules. The scheme used for parallelization of Q-Chem is discussed in Section 3, including a description of parallel two-electron integral routines and the parallel DFT numerical integration scheme. Section 4 presents results from benchmark calculations and Section 5 summarizes our findings.

2. The Q-Chem package

Q-Chem 1.2 was released in March 1999. It is a general-purpose package for performing quantum chemical calculations using contracted Gaussian basis sets and is designed to be particularly effective for large systems. It is capable of ground-state calculations using density functional theory (DFT), Hartree–Fock (HF) and MP2 theory and excited-state calculations using CIS, CIS(D) and related methods. It can compute gradients at most levels of theory and analytic second derivatives at the HF and CIS levels. It contains a recent version of the Baker OPTIMIZE package for the exploration of potential energy surfaces.

Attempts in the early 1990s to apply traditional quantum chemistry packages to large molecular systems were largely disappointing because the computational costs of conventional ab initio algorithms scale steeply as the size N of the Gaussian basis set increases. For example, conventional DFT and HF calculations scale as N^2, MP2 as N^5, CCSD as N^6, and CCSD(T) and MP4 as N^7. The recognition that these formidable bottlenecks had to be overcome before quantum chemistry could become a useful tool for biochemical problems has fueled a major research effort within several groups and has led to a number of significant advances during the last few years. Because the present paper is concerned with the Q-Chem package, we do not attempt to review here the important contributions of the Almlof, Arias, Friesner, Parrinello, Payne, Scuseria, St. Amant and Yang groups but focus instead on the contributions from the Q-Chem collaborators.

In most DFT calculations, the exchange-correlation energy is expressed as an integral over the molecular volume and, because the integral is not generally analytically evaluable, three-dimensional quadrature is used [15]. In 1992, Johnson et al. developed a complete quadrature algorithm [16] whose cost scales only linearly with the number of grid points and this was implemented in Q-Chem the following year. The standard grid in Q-Chem 1.2 is the SG-1 grid [17] and consists of roughly 2500 points per atom. It combines Euler–Maclaurin radial quadrature with Lebedev angular quadrature and has been systematically "pruned" of unimportant points.

In 1994, White et al. published the first algorithm [18] that computes the Coulomb energy of an ensemble of M overlapping Gaussian charge distributions in work that scales only linearly with M. This scheme, which was dubbed the "Continuous Fast Multipole Method (CFMM)", was an important first step towards the removal of the so-called "Coulomb bottleneck" and enabled Q-Chem to perform a self-consistent DFT energy calculation on a segment of RNA with more than 12,000 basis functions using only a mid-sized workstation. Details of DFT calculations using the CFMM were reported [19,20] two years later.

The CFMM achieves its high computational efficiency by splitting the Coulomb problem into near-field and far-field parts. The former, which concerns the interactions of pairs of charge distributions that overlap strongly, is treated via two-electron integrals [21,22] or the J-matrix engine [23]. The lat-

ter, which concerns non-overlapping distributions, is treated via exact multipole expansions. Dombroski et al. have shown [24] that it is also useful to split the Coulomb operator itself into short- and long-range parts and Lee et al. have discovered the optimal partition [25]. Approaches based on such partitions are known as KWIK methods.

Encouraged by success with the Coulomb problem, the Q-Chem collaborators turned to the more difficult task of computing the Fock exchange energy. This is a non-classical problem and remains an active area of research. However, in 1997, Challacombe et al. reported a scheme [26,27] that leads to linear-scaling exchange calculations in insulating systems with fairly large HOMO-LUMO separations (bandgaps). Their scheme, which was termed ONX, was later improved and led to the LinK and ONX-2 methods. Although none of these is more efficient than conventional two-electron approaches when applied to delocalized molecules, they have allowed hybrid DFT calculations to be undertaken on insulating systems with several thousand basis functions.

3. Q-Chem parallelization overview

Message Passing Interface (MPI) [28] based parallelism with *dynamic load balance* has been incorporated into both the Hartree–Fock and DFT modules of Q-Chem. MPI, which was developed jointly by researchers from industry, government laboratories and universities, was selected because it is easy to use, widely accepted and supported throughout the scientific community, available on all major platforms (including both shared and distributed memory architectures), and functions on a heterogeneous cluster of workstations. Thus, MPI offers great flexibility in terms of computing environments.

3.1. Parallelization of the two-electron integral routines

The single most important aspect of implementing any SCF algorithm on a parallel computer is evaluation of the two-electron integrals (coulomb and exchange) since their computation is, by far, the dominant step in a sequential HF calculation and a substantial step in a DFT one. In Q-Chem, the two-electron integrals are evaluated according to the PRISM algorithm [21,22]. This algorithm divides the two-electron integrals into batches according to angular momentum (L) and contraction (K) types, rather than according to atom centers as is the more common approach, since it is the L and K characteristics which determine computational cost. Batch size can be adjusted according to the amount of available memory. Parallelization of this construct is achieved simply by distributing the batches over the processors. This can be done either statically or dynamically, with the latter being more desirable as discussed below. A sketch of the parallel PRISM loop structure is as follows:

```
Broadcast (sparse) density matrix and
  orbital shell-pair data
Loop over two-electron-integral type
  Loop over batches of two-electron
    integrals of current type
      Do the current batch on the next
        available CPU
  End batch loop
End loop over integral type
```

By effectively sorting the integral classes by cost in advance and computing all integrals of the same cost together, it is possible to more equitably distribute the computational workload over all processors. However, experience has shown that even a careful (static) distribution of integral batches will not insure that a load balance is achieved, i.e. that all processors contribute equally to completing the assigned task. Rather a dynamic scheme, which assigns the integral batches to the processors on the fly is needed. This is even more important when running in heterogeneous computing environments such as a workstation cluster, where processor speeds and machine loads can vary greatly. In Q-Chem the dynamic load balance scheme is implemented as follows. Only a portion of the batches are assigned to the processors at the beginning of the SCF cycle. A large number of batches are held back and subsequently assigned to processors as they finish their current batch. The load balance routine in Q-Chem is based on the simple idea of a global shared counter (similar in spirit to the NXTVAL functionality in the TCGMSG toolset), and is currently implemented literally in this fashion, i.e. a single counter variable in shared memory that is accessed and incremented sequentially by the processors using

a lock on the counter memory. Thus processors that finish their assignment early are given additional work. This insures that all processors are busy computing integrals for the entire time integral evaluation takes place. This method is similar in spirit to that devised by Furlani and King in their implementation of a parallel direct SCF method for distributed memory parallel computers [10–12].

3.2. Parallelization of DFT numerical integration

We now discuss the parallelization of an important component of a large-scale DFT calculation, namely, the numerical integration of the exchange-correlation (XC) functional to give the XC energy and matrix elements. This is a significant computational step for large-scale DFT calculations; whereas long ago we developed linear-scaling methods for this step, our subsequent linear-cost methods for Coulomb and HF exchange are *analytic* rather than numerical, and thus the linear-cost XC quadrature can be the dominant step in a large calculation.

The XC quadrature is carried out on a superposition of atom-centered grids, with approximately 2500 points per atom representing a typical grid size. Since the grid size grows with the molecular size, parallelization over the grid is a good candidate for massive parallelism on large molecules. Thus, we have structured our serial algorithm to be driven over the atomic grids as follows:

```
Loop over atoms
  Construct quadrature grid for current atom
    and Becke weights [15]
  Loop over grid batches
    On the next available CPU:
      Evaluate basis functions, density and
        functional on the grid
      Evaluating contribution to XC energy
        and matrices
  End batch loop
End atom loop
```

As the atomic contributions are completely independent of each other, we have constructed a very efficient low-communication distributed-memory parallel algorithm. In this algorithm the sparse density matrix is broadcast at the start, the atomic grids are assigned in parallel with dynamic load balance, and the desired results (XC energy and matrices) are global-summed at the end. Note that the global sums may be carried out *after* the numerical integration has been performed due to the way the algorithm is structured, and thus the communication involved is *independent* of the size of the numerical grid, a desirable feature.

For systems smaller than 100 atoms or so, the straightforward approach of distributing the atoms over the processors is likely not to achieve good load balance (even with dynamic correction) due to the coarse grain. In this case the problem is easily remedied by sub-dividing the atomic quadrature grids into (again) independent batches of points and distributing these batches to the processors. Up to 10 parallel batches can be constructed from each atomic grid if necessary without sacrificing serial performance. This adds only minimal complication to the parallel algorithm.

4. Results from benchmark calculations

A series of benchmark calculations consisting of both *single point energy and gradient calculations* were carried out to gauge the performance of the parallel Hartree–Fock and DFT modules in Q-Chem. Although Q-Chem offers users a wide range of options from which to choose, geometry optimizations and single point energy calculations represent the most commonly utilized options. Efficient parallelization of the energy and the gradient of the energy with respect to the position of the atoms (which is needed for geometry optimization) is therefore critical. Calculations were carried out on two widely different parallel computers, namely a shared memory Silicon Graphics Origin2000 (250 MHz, R10000) and a distributed memory Cray T3E-900 (450 MHz), to show the flexibility of the code and to demonstrate the great utility of MPI. Before describing our results, we present a discussion of parallel performance in order to provide a context for the discussion that follows.

The performance of a parallel program, which is usually measured by how much execution time decreases as one increases the number of processors, is dependent on several factors including, problem size and architecture. For example, for a fixed problem size, parallel performance eventually falls off as one increases the number of processors due to the fact

that there is a finite amount of computational work to distribute. A useful measure of performance is the speed-up that an algorithm achieves as the number of processors is increased. Speed-up (S) is defined as the ratio of the execution time of the serial program (t_s) to the execution time of the parallel program (t_p) on a given number of processors.

$$S = t_s/t_p.$$

In an ideal parallel algorithm, the speed-up increases by a factor of 2 every time the number of processors is doubled.

Here we report execution time and speed-up versus number of processors for Hartree–Fock and DFT calculations. For most Hartree–Fock and DFT calculations, evaluation of the two-electron integrals accounts for 90–99% of the execution time. The majority of the remaining time is spent in linear algebra (matrix diagonalization and multiplication). Since our initial parallelization focused only on parallelization of the two-electron integral code and not on linear algebra, the best we can hope to achieve in terms of speed-up is a factor of 10–100. Eventually as one decreases the time spent in the parallel portion of the code by increasing the number of processors, the sequential portion of the calculation (in this case matrix diagonalization and multiplication) will begin to dominate. Note that parallelization of the linear algebra portions of Q-Chem is underway and will be reported on in a later paper.

Results for single point energy and gradient calculations for α-pinene ($C_{10}H_{16}$, 6–311$G(df, p)$) on the Cray T3E and Origin2000 are contained in Tables 1–5. The first 3 of these tables report timings on the T3E while the last 2 tables contain results from calculations carried out on the Origin2000. The SCF convergence criteria was 10^{-4} DIIS error for all the single-point calculations, and 10^{-8} for all the gradient calculations. Please note that, due to time limits on the batch queues for the CRAY T3E at NERSC, we were unable to run any of the T3E calculations on fewer than 2 processors. The execution times for calculations carried out on a single T3E processor were therefore obtained by multiplying the execution time for a calculation run on two processors by a factor of 2.0 (this assumes perfect speed-up). On the SGI Origin2000 at the University at Buffalo's Center for Computational Research (www.ccr.buffalo.edu) this construct was not necessary as we were able to run each benchmark on a single processor. Thus the execution times reported for the Origin2000 were all obtained directly from calculations.

Table 1
Execution time and speed-up for Hartree–Fock single-point calculations of α-pinene (6–311$G(df, p)$, 346 basis functions) on a T3E-900 (450 MHz). Timings are for SCF portion of calculation

Number of processors	CPU time (sec)	Speed-up
1	6880.	1.0
2	3440.	2.0
8	933.	7.4
16	521.	13.2
32	322.	21.4
64	253.	27.1

Table 1 shows the performance for the SCF portion of a single point Hartree–Fock calculation for α-pinene carried out on the Cray T3E (346 basis functions). The speed-up is excellent on 16 processors and still good on 32-processors, even for this relatively modest size calculation. It does, however, begin to fall off sharply after about 32 processors. As discussed above, this is attributable to the non-parallelized portion of the calculation, namely matrix diagonalization and multiplication. Results for computation of the Hartree–Fock *gradient* for α-pinene are contained in Table 2. Here the speed-up is considerably better than that for the single point calculation with a speed-up of 42 on 64 processors (there are no matrix diagonalizations involved in the gradient calculations). Table 3 shows the result for a single-point DFT calculation on α-pinene using a B3LYP functional (T3E). Although the performance is slightly better than that for the HF single point energy calculation, it still shows a significant fall off after 32 processors.

We hasten to point out that the observed fall-offs are not unexpected. In the initial parallelization scheme we addressed the most time consuming part of HF and DFT calculations, namely evaluation of the two-electron integrals. We realized that success in this phase of the calculation would necessitate our subsequently addressing the remaining non-parallel components, namely the linear algebra. Work in this area is underway.

Table 4 contains results for a single point DFT (B3LYP) calculation for α-pinene on an Origin2000

Table 2
Execution time and speed-up for Hartree–Fock gradient calculations of α-pinene (6–311$G(df, p)$, 346 basis functions) on a Cray T3E-900 (450 MHz). Timings are for SCF portion of calculation

Number of processors	CPU time (sec)	Speed-up
1	8294.	1.0
2	4147.	2.0
8	1071.	7.7
16	557.	14.9
32	310.	26.8
64	200.	41.5

Table 3
Execution time and speed-up for DFT calculations of α-pinene (B3LYP, 6–311$G(df, p)$, 346 basis functions) on a Cray T3E-900 (450 MHz). Timings are for SCF portion of calculation

Number of processors	CPU time (sec)	Speed-up
1	10512.	1
2	5256.	2
8	1427.	7.4
16	793.	13.3
32	488.	21.5
64	362.	29.0

Table 4
Execution time and speed-up for DFT calculations of α-pinene (B3LYP, 6–311$G(df, p)$, 346 basis functions) on an SGI Origin2000 (R10000, 250 MHz). Timings are for SCF portion of calculation

Number of processors	CPU time (sec)	Speed-up
1	6041.	1.0
2	3343.	1.8
8	822.	7.3
16	443.	13.6
32	251.	24.1
64	162.	37.3

Table 5
Execution time and speed-up for DFT gradient calculations of α-pinene (B3LYP, 6–311$G(df, p)$, 346 basis functions) on an Origin2000 (R10000, 250 MHz). Timings are for SCF portion of calculation

Number of processors	CPU time (sec)	Speed-up
1	6795.	1.0
2	3816.	1.8
8	944.	7.2
16	503.	13.5
32	272.	25.0
64	136.	50.0

as opposed to the T3E. The Origin2000 results are similar to the T3E results (Table 3), although the performance on 32 and especially 64 processors is significantly better on the Origin2000. In the latter case, the speedup is a factor of 29 on the T3E and a factor of 37 on the Origin2000. Finally, in Table 5 we report the results for computation of the *gradient* for α-pinene using density functional theory on the Origin2000. The results are similar to that obtained on the T3E using Hartree–Fock theory (Table 2). The only significant difference being for 64 processors where the Origin2000 outperformed the T3E in terms of speed-up. Since the T3E calculation is based on HF theory and the Origin2000 calculation on DFT theory, it is not possible to attribute the difference in speed-up solely to machine architecture, as was possible when comparing the single point DFT calculations on the two platforms (Table 3 versus Table 4).

We wish to point out that the timing results presented here compare favorably, both in terms of execution time and speed-up, with those reported by Sosa and coworkers [13] for Gaussian-94 software on a T3E. In terms of single processor performance for a 6–311$G(df, p)$ SP calculation for α-pinene, Gaussian reports an execution time of 8994 seconds on a T3E-900. The corresponding time for the same calculation using Q-Chem is 6880 seconds. For the DFT calculations the times are 13,044 seconds for Gaussian and 10,512 seconds for Q-Chem. In all cases the speed-up achieved with Q-Chem is equivalent to that obtained by Gaussian on the T3E [13].

5. Conclusion

We have successfully incorporated MPI based parallelism with *dynamic load balance* into both the

Hartree–Fock and DFT modules of Q-Chem. A series of benchmark calculations consisting of both *single point energy and gradient calculations* were carried out to gauge the performance of the parallel modules. Although Q-Chem offers users a wide range of options from which to choose, geometry optimizations and single point energy calculations represent the most commonly utilized options. Accordingly, efficient parallelization of these two procedures has the potential to have a significant impact on a large number of end-users. Calculations were carried out on two widely different parallel computers, namely a shared memory Silicon Graphics Origin2000 and a distributed memory Cray T3E, to show the flexibility of the code and demonstrate the great utility of MPI. Benchmark calculations on both the SGI and T3E clearly demonstrate the utility of parallel processing for quantum chemical calculations as implemented in Q-Chem. Results on both the SGI Origin2000 and CRAY T3E are very good. For example, speed-ups of a factor of 24 and 22 were obtained on 32 processors of an Origin2000 and CRAY T3E, respectively. Even though performance falls off as the number of processors is increased (for a fixed problem size), the observed decreases in execution time are significant. For example, a geometry optimization requiring 2 weeks on a single processor high-performance workstation (which is not an uncommon occurrence) could easily be run overnight on 16 processors. Further work is underway to parallelize the remaining sequential portions of the code (primarily the linear algebra routines) to further improve performance.

Acknowledgement

TRF gratefully acknowledges support from NSF grants DBI9871132 and ATM971338. JK and TRF also acknowledge support from NIH grant GM58295-01. Access to the Origin2000 at the University at Buffalo's Center for Computational Research and the T3E at NERSC is also greatly appreciated.

References

[1] T.G. Mattson (Ed.), Parallel computing in computational chemistry, ACS Symp. Ser. 592 (1995).
[2] R. Weist, J. Demuynck, M. Bernard, M. Rohmer, R. Emenwein, Comput. Phys. Commun. 62 (1991) 107–124.
[3] M.E. Colvin, R.A. Whitside, H.F. Schaefer III, in: Methods in Quantum Chemistry, Vol. 3, S. Wilson (Ed.) (Plenum Press, NY, 1989) p. 167.
[4] M.W. Feyereisen, R.A. Kendall, Theor. Chim. Acta 84 (1993) 289.
[5] M.E. Colvin, C.L. Janssen, R.A. Whiteside, C.H. Tong, Theor. Chim. Acta 84 (1993) 301.
[6] R.J. Harrison, R. Shepard, Annu. Rev. Phys. Chem. 45 (1994) 623.
[7] R.J. Harrison, M.F. Guest, R.A. Kendall, D.E. Bernholdt, A.T. Wong, M. Stave, J.L. Anchell, A.C. Hess, R.J. Littlefield, G.L. Fann, J. Nieplocha, G.S. Thomas, D. Elwood, J. Tilson, R.L. Shepard, A.F. Wagner, I.T. Foster, E. Lusk, R. Stevens, J. Comput. Chem. 16 (1996) 124.
[8] D. Chasman, M.D. Beachy, L. Wang, R.A. Friesner, J. Comput. Chem. 19 (1998) 1017.
[9] M.W. Schmidt, K.K. Baldridge, J.A. Boatz, S.T. Elbert, M.S. Gordon, J.H. Jensen, S. Koseki, N. Matsunaga, K. Nguyen, S. Su, T.L. Windus, M. Dupuis, J.A. Montgomery, Jr., J. Comput. Chem. 14 (1993) 1347.
[10] T.R. Furlani, H.F. King, J. Comput. Chem. 16 (1995) 91.
[11] T.R. Furlani, H.F. King, in: Quantum mechanical Simulation Methods for Studying Biological Systems, D. Biscout, M.F. Field (Eds.) (Springer, France, 1996) p. 271.
[12] J. Gao, T.R. Furlani, IEEE Comput. Sci. Engrg. 2 (1995) 24.
[13] C.P. Sosa, J. Ochterski, J. Carpenter, M.J. Frisch, J. Comput. Chem. 19 (1998) 1053.
[14] C.A. White, J. Kong, D.R. Maurice, T.R. Adams, J. Baker, M. Challacombe, E. Schwegler, J.P. Dombroski, C. Ochsenfeld, M. Oumi, T.R. Furlani, J. Florian, R.D. Adamson, N. Nair, A.M. Lee, N. Ishikawa, R.L. Graham, A. Warshel, B.G. Johnson, P.M.W. Gil, M. Head-Gordon, Q-Chem, Version 1.2 (Q-Chem, Inc., Pittsburgh, PA, 1998).
[15] A.D. Becke, J. Chem. Phys. 88 (1988) 2547.
[16] B.G. Johnson, Ph.D. Thesis, Carnegie Mellon University (1993).
[17] P.M.W. Gill, B.G. Johnson, J.A. Pople, Chem. Phys. Lett. 209 (1993) 506.
[18] C.A. White, B.G. Johnson, P.M.W. Gill, M. Head-Gordon, Chem. Phys. Lett. 230 (1994) 8.
[19] C.A. White, B.G. Johnson, P.M.W. Gill, M. Head-Gordon, Chem. Phys. Lett. 253 (1996) 268.
[20] B.G. Johnson, C.A. White, Q.-M. Zhang, B. Chen, R.L. Graham, P.M.W. Gill, M. Head-Gordon, Advances in methodologies for linear-scaling density functional calculations, in: Recent Developments in Density Functional Theory, Vol. 4, J.M. Seminario (Ed.) (Elsevier Science B.V., Amsterdam, 1996) p. 441.
[21] P.M.W. Gill, Adv. Quantum Chem. 25 (1994) 141.
[22] T.R. Adams, R.D. Adamson, P.M.W. Gill, J. Chem. Phys. 107 (1997) 124.
[23] C.A. White, M. Head-Gordon, J. Chem. Phys. 104 (1996) 2620.
[24] J.P. Dombroski, S.W. Taylor, P.M.W. Gill, J. Phys. Chem. 100 (1996) 6272.
[25] A.M. Lee, S.W. Taylor, J.P. Dombroski, P.M.W. Gill, Phys. Rev. A 55 (1997) 3233.

[26] M. Challacombe, E. Schwegler, J. Chem. Phys. 106 (1997) 5526.
[27] E. Schwegler, M. Challacombe, M. Head-Gordon, J. Chem. Phys. 106 (1997) 9703.
[28] The MPI Forum, Proceedings of Supercomputing '93, IEEE Computer Society, Los Alamitos, CA, 1991, p. 878.