Theor Chem Acc (2006) 115: 221–226
DOI 10.1007/s00214-005-0005-0

REGULAR ARTICLE

Patrizia Calaminici · Victor D. Domínguez-Soria
Gerald Geudtner · Elizabeth Hernández-Marín
Andreas M. Köster

Parallelization of three-center electron repulsion integrals

Received: 16 December 2004 / Accepted: 6 May 2005 / Published online: 6 December 2005

Abstract The parallelization of the three-center electron repulsion integrals arising from the variational fitting of the Coulomb potential is presented. A scheme for dynamical load balancing of the corresponding loop structure is discussed. The implementation in the density functional theory program deMon using the message passing interface is described. The efficiency of the parallelization is analyzed by selected benchmark calculations.

Keywords DFT · Parallelization · ERI · deMon · Load balancing

1 Introduction

Over the last decade the interest in density functional theory (DFT) methods has grown dramatically within computational chemistry. Numerous systematic studies have shown that DFT methods offer a promising alternative to the more traditional Hartree–Fock approach. Most interestingly, DFT methods include electron correlation at a formal cubic scaling. This is a fundamental difference to Hartree–Fock based methods, where the inclusion of correlation typically increases the computational effort considerably. The rate-limiting steps in today's DFT methods using localized atomic orbitals are the calculations of the two-electron repulsion integrals and the numerical integration of the exchange-correlation contribution. The use of auxiliary functions has a long history in DFT methods [1]. Many years ago Sambe and Felton [2] proposed the use of auxiliary functions in the framework of X_α implementations using the linear combination of Gaussian-type orbitals (LCGTO). Based on this early work Dunlap et al. [3,4] introduced the variational fitting of the Coulomb potential in order to avoid the evaluation of four-center integrals in DFT methods. More recently, the so-called resolution of the identity has gained attention [5]. The working formulas are identical to the variational fitting of the Coulomb potential without constraints. The variational fitting of the Coulomb potential reduces the formal scaling to $N^2 \times M$, where N is the number of basis functions and M the number of auxiliary functions. Usually the number of auxiliary functions is three to five times the number of basis functions. It is obvious that this fitting can improve the efficiency of DFT methods considerably. Moreover, it has been shown in the past that the variational approximation of the Coulomb potential is robust and within the intrinsic accuracy of LCGTO-DFT methods [6]. This also includes the availability of accurate gradients and second order properties [7, 8]. For the variational fitting of the Coulomb potential an approximated density is introduced. It is expanded in atom-centered auxiliary functions. The expansion coefficients are obtained variationally by the minimization of the following self-interaction error:

$$\mathcal{E}_2 = \iint \frac{|\,\rho(\mathbf{r}_1) - \tilde{\rho}(\mathbf{r}_1)\,||\,\rho(\mathbf{r}_2) - \tilde{\rho}(\mathbf{r}_2)\,|}{|\,\mathbf{r}_1 - \mathbf{r}_2\,|}\, d\mathbf{r}_1\, d\mathbf{r}_2 \qquad (1.1)$$

Here, $\rho(\mathbf{r})$ and $\tilde{\rho}(\mathbf{r})$ are the orbital and approximated densities, respectively. In most implementations primitive or contracted Cartesian Gaussian functions are used for the expansion of the approximated density. Therefore, three-center electron repulsion integrals (ERIs) over Cartesian Gaussian functions have to be evaluated. In earlier implementations, modified recurrence relations of Obara and Saika (OS) [9], which connect a given integral to others with lower angular momentum, have been used [10]. In a more recent implementation [11] a modified OS method [12] was combined with the method from McMurchie and Davidson (MD) [13] for the evaluation of the ERIs. In the MD method Hermite Gaussian functions [14] are used as intermediates to calculate the integrals over Cartesian Gaussian functions.

Very recently, the use of primitive Hermite Gaussian functions for the expansion of the auxiliary function density has been proposed in deMon2k [15]. The use of primitive Hermite Gaussian auxiliary functions was first proposed by Mintmire et al. [16] for the calculation of periodic systems and has

P. Calaminici (✉) · V. D. Domínguez-Soria · G. Geudtner
E. Hernández-Marín · A. M. Köster
Departamento de Química, CINVESTAV,
Avenida Instituto Politécnico Nacional,
2508 A. P. 14-740, Mexico D.F. 07000, Mexico
E-mail: pcalamin@mail.cinvestav.mx

been implemented in the FILMS [16] and GTOFF [17] codes. Within deMon2k it was also shown that this auxiliary function density can be directly used for the calculation of the exchange-correlation potential [18]. Because the approximated density is a linear combination of auxiliary functions the density calculation and, thus, the exchange-correlation potential calculation at each grid point becomes linear. Instead, along with the orbital density, products of basis functions have to be evaluated. Obviously, this represents a considerable simplification of the grid work. However, the number of auxiliary functions is usually three to five times the number of basis functions. This could deteriorate the performance of the approximated density calculation on the grid because the calculation of the exponential function is computationally not negligible. Therefore, we are using in our implementation primitive Hermite Gaussian functions which are grouped together in sets, sharing the same exponents. As an example, a d auxiliary function set contains ten primitive Hermite Gaussians, one s, three p and six d functions, all with the same exponent. Thus, the number of exponents that have to be evaluated at each grid point is for an auxiliary function density of this structure much smaller than for the corresponding orbital density. Moreover, by using Hermite Gaussian auxiliary functions the Hermite polynomial recurrence relations [14] can be used for the function calculation on the grid. With this approach the computational demand for the numerical integration of the exchange-correlation energy and potential becomes secondary with respect to the calculation of the ERIs. Therefore, we decided to parallelize the calculation of these integrals. It should be noted that the same problem had already been tackled in other electronic structure codes like NWChem [19,20] and GAMESS–UK [21] following different strategies.

In this paper we describe the basic strategy for the parallelization of the three-center electron repulsion integral calculation using FORTRAN and the message passing interface (MPI). In the following section the strategy for the parallelization of the DFT code deMon2k [22] using MPI is discussed in general. In Sect. 3 the parallelization of the three-center electron repulsion is presented. Benchmark results are discussed in Sect. 4 and concluding remarks are drawn in the last section.

2 Parallelization strategy

The parallelized routines of a program usually have to exchange data. To handle this communication the widely used Message Passing Interface (MPI) is employed in deMon2k. Under MPI each parallel execution is called a task. It should be noted that such a task is not necessarily connected to its own node or processor. In principle several tasks may run on the same processor using MPI. However, for our discussion here we can assume that each task is connected to its own CPU.

The transfer of data between tasks is initiated inside a program by calls to MPI routines. The calls to these system routines would lead to an error in the link step during the generation of the serial deMon2k executable on a single processor computer because of the absence of the necessary libraries. Therefore, the CALL statements for the MPI routines are replaced by CALL statements of interface routines. Two sets of these interface routines exist within the deMon2k source code, one for the serial version and another for the parallel version. For the serial version these routines are dummy routines without any code to be executed. For the parallel version they include the calls to the MPI routines. This strategy has the advantage that only a few routines have to be exchanged in order to compile a parallel or serial version of deMon2k. Obviously, this simplifies code maintenance and ensures the integrity of both parallel and serial versions. It also allows an easy change of the messaging system, e.g. from MPI to shmem or parallel virtual machine (PVM).

3 Parallelization of three-center electron repulsion integrals

As already described in the introduction, the calculation of the three-center ERIs becomes the most computationally demanding step if the exchange-correlation potential is calculated with the auxiliary function density. In the direct self-consistent field (SCF) approach these integrals have to be calculated twice in each SCF step, once for the construction of the Kohn-Sham matrix elements,

$$K_{\mu\nu} = H_{\mu\nu} + \sum_{k} \langle\, \mu\nu \parallel k \,\rangle\, (\, x_k + z_k \,), \tag{3.1}$$

and again for the calculation of the fitting coefficients,

$$x_l = \sum_{k} G_{lk}\, J_k \quad , \quad J_k = \sum_{\mu,\nu} P_{\mu\nu}\, \langle\, \mu\nu \parallel k \,\rangle. \tag{3.2}$$

Here $H_{\mu\nu}$ and $P_{\mu\nu}$ denote elements of the core Hamiltonian and the density matrix. The fitting coefficients of the approximated density and the exchange-correlation potential are represented by x_k and z_k, respectively. To avoid unnecessary complication in the presentation we have dropped terms arising from the normalization of the approximated density (see [18] for more details). Here we will focus on the parallel computation of the ERIs $\langle\, \mu\nu \parallel k \,\rangle$. In this notation, μ and ν refer to contracted Cartesian Gaussian functions which represent the basis set of the calculation. The primitive Hermite Gaussian functions used for the auxiliary function expansion are denoted by k. The symbol $\parallel$ stands for the Coulomb operator $1/|\mathbf{r_1} - \mathbf{r_2}|$.

In order to achieve a good efficiency of the parallelization it is necessary to distribute the work, i.e. the calculation of the ERIs, homogeneously over the tasks. For this reason the parallelization of the innermost loop over the auxiliary functions would be too coarse to achieve a good load balancing of the tasks. It is, therefore, more efficient to parallelize over the shells or shell combinations. Here a shell defines a set of contracted basis functions that share the same exponents and contraction pattern, e.g. s, p or d orbitals. Each shell combination possesses a set of orbital products $\mu\nu$. The number of these products depend on the quantum number of the orbitals in the two shells. The different number of orbital products in the various shell combinations as well as the screening of ERIs based on the maximum overlap integrals of the shell combinations makes the workload balancing a formidable task. Moreover, the dynamical screening of shell combinations based on density matrix differences in the direct SCF procedure forbids any static workload balancing. This situation calls for a dynamical load balancing as described now.

At the beginning of the SCF, the active number of shell combinations N_{shell}^{act} is calculated, active here referring to all shell combinations that are not screened. The ERIs are then homogeneously distributed according to N_{shell}^{act} and the number of orbital products in each shell. The order of the active shell combination is determined by the ERI loop structure and for each CPU lower and upper limits within this, implicit lists are assigned. For each shell combination the inner auxiliary function loop is then processed within the assigned task. The computational time for the ERI calculation in the construction of the Kohn-Sham matrix and the Coulomb vector J for each task is then measured. Based on these time stamps a redistribution of the active number of shell combination for each task is performed according to the following formula:

$$\omega_I^{(n+1)} = \frac{\omega_I^{(n)}}{t_I^{(n)} \sum_J^N \frac{\omega_J^{(n)}}{t_J^{(n)}}} \tag{3.3}$$

Here $\omega_I^{(n)}$ represents the fractions of N_{shell}^{act} calculated by the task I in the nth SCF cycle divided by 100, $t_I^{(n)}$ denotes the corresponding time stamp for task I in the nth SCF cycle and N is the total number of available tasks in the calculation. After 3 SCF cycles a stable and very efficient load balancing is achieved. We refer to this approach as dynamical ERI load balancing. In order to be more explicit we now present an example for the dynamical load balancing from a calculation of the Coulomb vector J on four CPUs. As test system a $C_{72}H_{146}$ calculation with the DZVP basis and the VWN functional was used. In the first step N_{shell}^{act} was 28502. The first fractions were $\omega_1^{(1)} = 0.232$, $\omega_2^{(1)} = 0.237$, $\omega_3^{(1)} = 0.237$ and $\omega_4^{(1)} = 0.293$. This results in local N_{shell}^{act} ranges on the 1st CPU from 1 to 6,625, on the 2nd CPU from 6,626 to 13,382, on the 3rd CPU from 13,383 to 20,137 and on the 4th CPU from 20,138 to 28,502. The corresponding time stamps for the J vector calculation were $t_1 = 33.65$ s, $t_2 = 34.77$ s, $t_3 = 34.28$ s and $t_4 = 37.60$ s. From these timings the normalized time stamps

$$t_{I,norm}^{(n)} = \frac{t_I^{(n)}}{\sum_J^N t_J^{(n)}} \tag{3.4}$$

are obtained as $t_{1,norm} = 0.240$, $t_{2,norm} = 0.248$, $t_{3,norm} = 0.244$, and $t_{4,norm} = 0.268$. With these time stamps and the above formula for $\omega_I^{(n+1)}$ new fractions for the distribution of N_{shell}^{act} in the 2nd SCF cycle were calculated. The results were $\omega_1^{(2)} = 0.243$, $\omega_2^{(2)} = 0.240$, $\omega_3^{(2)} = 0.243$ and $\omega_4^{(2)} = 0.274$. In the 2nd SCF cycle N_{shell}^{act} is 47,086. The number is much larger then in the 1st SCF cycle because the tight-binding density matrix used to start the SCF procedure is much sparser as the now formed ab-initio density matrix. The distribution of this number of active shell combinations according to the $\omega_I^{(2)}$ values leads to the following $t_{I,norm}$ values: $t_{1,norm} = 0.257$, $t_{2,norm} = 0.253$, $t_{3,norm} = 0.257$ and $t_{4,norm} = 0.233$. The corresponding time stamps lead to new $\omega_I^{(3)}$ for the 3rd SCF cycle. In Table 1 the complete set of different values for the first six SCF cycles are shown. One can see that in the 3rd SCF cycle the $t_{1,norm}$ values are about 0.25, which means that each of the four CPUs is doing $1/4$ of the work for the calculation of the ERIs even if the N_{shell}^{act} value changes in every SCF cycle. Because of the similar loop structure for the calculation of the ERIs for the Kohn-Sham (KS) matrix construction a very similar scheme for parallelization of this calculation is used. The same scheme is also applied for the parallelization of the Coulomb integral gradients.

4 Benchmark results

As already mentioned we used the optimization of the $C_{72}H_{146}$ Alkene for benchmark calculations. In this system the time for the calculation of parts related to the three center integrals, e.g. the construction of the KS matrix, the Coulomb vector J and the Coulomb gradients, represent about 71% of the total CPU time on one processor. The parts related to the calculation of the exchange correlation energy are only about 10% of the total CPU time. The system was calculated on a cluster with Intel® Xeon™ 2.4 GHz CPUs on 1, 2, 4 and 6 CPUs. The results of the efficiency of the parallelized version for the different number of CPUs are shown in Fig. 1. The efficiency E_N is defined as the quotient of the calculation time on one CPU, t_1, and the product of the calculation time on N CPUs, t_N, with the number, N, of used CPUs.

$$E_N = \frac{t_1}{N t_N} \tag{4.1}$$

Therefore the efficiency E_N multiplied by the number of CPUs gives the speedup. The curves in Fig. 1 show for the Coulomb vector J and for the Coulomb gradients a very good efficiency. The data used for the calculation of the efficiency values shown in the figure are timings which are the sum over all steps in all SCF cycles in the calculations. They include the time for the calculation of the integrals, the distribution of the different intervals to the CPUs, the refinement of these intervals and the time for gathering the results. In the case of the Coulomb gradients the used data are the sum of the timings of the Coulomb gradient calculation over all optimization steps. With six CPUs the efficiency for these parts of the code is over 90%, which represents a speedup of about 5.5. The efficiency for the calculation of the ERIs for construction of the KS matrix is less good. Nevertheless, a speedup

Table 1 Number of active shells N_{shell}^{act} and their distribution ω_1, ω_2, ω_3 and ω_4. According to the dynamical ERI load balancing for the first six SCF cycles of $C_{72}H_{146}$. The corresponding time steps [s] and normalized time steps are also given

	SCF cycle					
	1	2	3	4	5	6
N_{shell}^{act}	28502	47086	45482	44144	43222	42930
ω_1	0.23244	0.24284	0.23554	0.23743	0.23641	0.23576
ω_2	0.23707	0.23970	0.23627	0.23666	0.23507	0.23626
ω_3	0.23700	0.24305	0.23529	0.23605	0.23533	0.23661
ω_4	0.29349	0.27441	0.29290	0.28985	0.29318	0.29137
t_1	33.65	60.80	56.95	56.41	55.21	54.42
t_2	34.77	59.83	57.31	56.55	54.78	54.57
t_3	34.28	60.92	57.22	56.34	54.76	54.75
t_4	37.60	55.25	58.01	55.53	55.40	54.48
$t_{1,norm}$	0.23984	0.25676	0.24816	0.25090	0.25078	0.24938
$t_{2,norm}$	0.24783	0.25266	0.24973	0.25152	0.24883	0.25007
$t_{3,norm}$	0.24433	0.25726	0.24934	0.25059	0.24874	0.25089
$t_{4,norm}$	0.26800	0.23332	0.25278	0.24699	0.25165	0.24966

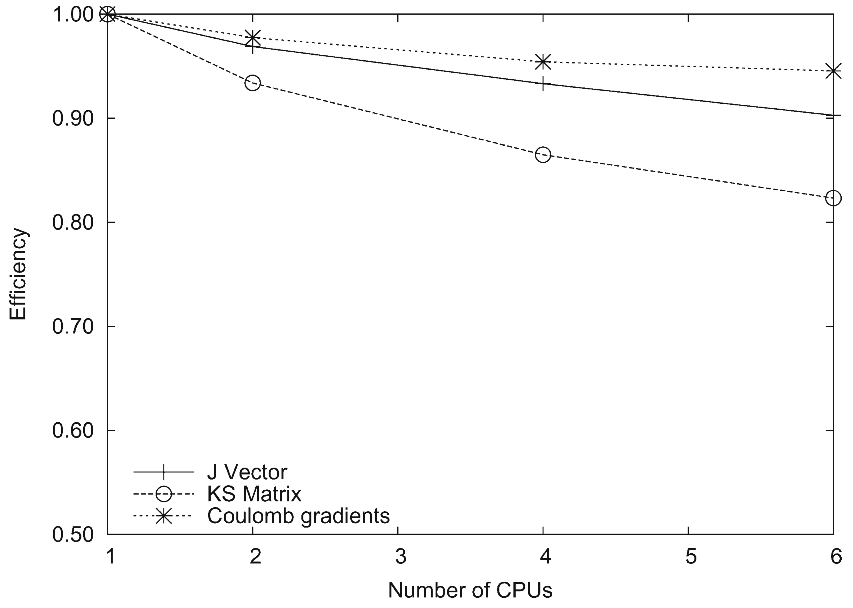

Fig. 1 Efficiency of the parallelized ERI Calculations for the Coulomb vector construction (*J* Vector), the Kohn-Sham (KS) matrix and the Coulomb gradients depending on the number of CPUs for the optimization of the $C_{72}H_{146}$ system

with six CPUs of about five is reached. Because of different screening techniques [15] used for the ERIs in the construction of the KS matrix it is more difficult for the dynamic load balancing mechanism to achieve an even workload here. This is due to other parts in the program that are less efficiently parallelized. Still an overall speedup of four for six CPUs is achieved. The efficiency values for larger number of CPUs will decrease because the parts each CPU has to calculate will become smaller and smaller. As a result, the dynamic load balancing becomes less efficient. As the system size increases an opposite effect is observed. This means that the efficiency of the parallelization increases with system size for a fixed number of CPUs.

We also benchmarked two other systems. These are fullerene systems with 100 and 240 carbon atoms. Different to the $C_{72}H_{146}$ molecule we used here up to 10 CPUs for the calculations. The efficiency results for the C_{100} system are shown in Fig. 2 and for the C_{240} system in Fig. 3. The results are similar to the $C_{72}H_{146}$ molecule. One can also see that for the fullerenes the efficiency for the calculation with six CPU is slightly better then for the alkane chain. This is due to the larger system size. Different from the alkane chain, the fullerenes show a less continuous behavior in efficiency with respect to the number of CPUs. The main reason for this different behaviour is the cluster load. Because the cluster is equipped with dual processor boards a bottleneck in

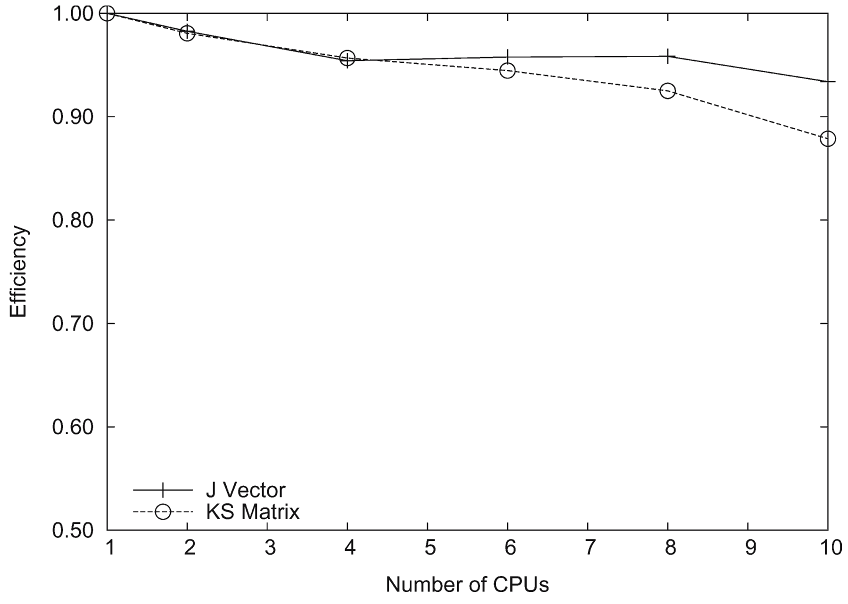

Fig. 2 Efficiency of the parallelized ERI Calculations for the Coulomb vector construction (*J* Vector) and the Kohn-Sham (KS) matrix depending on the number of CPUs for the C_{100} system

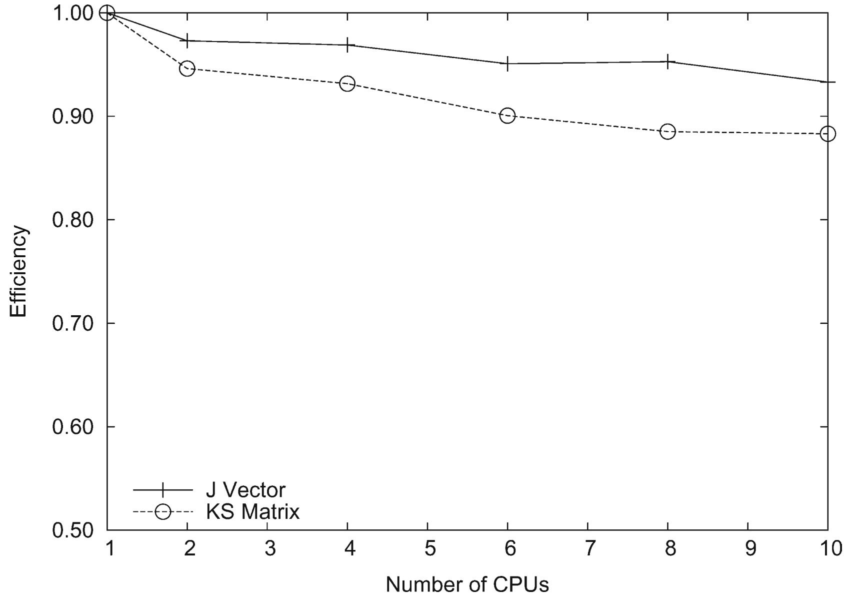

Fig. 3 Efficiency of the parallelized ERI Calculations for the Coulomb vector construction (*J* Vector) and the Kohn-Sham (KS) matrix depending on the number of CPUs for the C_{240} system

the memory access occurs when both CPUs are used at the same time. Of course, this bottleneck influences the overall calculation time. Thus, Figs. 2 and 3 show the behavior of the parallelized deMon2k code in a real production environment.

5 Conclusions

An efficient algorithm for the parallelization of the calculation of the three-center electron repulsion integrals in the framework of the DFT code deMon2k is presented. This algorithm has the characteristic to adapt to the changing conditions during the SCF cycles. It is also well suited for heterogeneous computing environments. This is achieved by a dynamical load balancing. Because similar situations also occur in the calculation of the exchange correlation potential this algorithm can be used here too. Implementation requires only a few routines to be changed in order to compile a parallel or a serial version of the code used. The efficiency of the parts of the program in which this algorithm is used is, with six CPUs, about 85% or higher.

Acknowledgements Financial support from the CONACYT projects 36037-E and 40379-F is gratefully acknowledged.

References

1. Baerends EJ, Ellis DE, Ros P (1973) Chem Phys 2:41
2. Sambe H, Felton RH (1975) J Chem Phys 62:1122
3. Dunlap BI, Connolly JWD, Sabin JR (1979) J Chem Phys 71:4993
4. Mintmire JW, Dunlap BI (1982) Phys Rev A 25:88
5. Vahtras O, Almlöf J, Feyereisen MW (1993) Chem Phys Lett 213:514
6. Dunlap BI (2000) J Mol Struct (THEOCHEM) 529:37
7. Dunlap BI, Andzelm J (1992) Phys Rev A 45:81
8. Komornicki A, Fitzgerald G (1993) J Chem Phys 98:1398
9. Obara S, Saika A (1986) J Chem Phys 84:3963
10. Andzelm J, Wimmer E (1992) J Chem Phys 96:1280
11. Köster AM (1996) J Chem Phys 104:4114
12. Head-Gordon M, Pople JA (1988) J Chem Phys 89:5777
13. McMurchie LE, Davidson ER (1978) J Comput Phys 26:218
14. Saunders VR (1983) In: Methods in Computational Physics, (eds) Diercksen GHF, Wilson S Reidel, Dordrecht
15. Köster AM (2003) J Chem Phys 118:9943
16. Mintmire JW, Sabin JR, Trickey SB (1982) Phys Rev B 26:1743
17. Boettger JC, Trickey SB (1996) Phys Rev B 53:3007
18. Köster AM, Reveles JU, del Campo JM (2004) J Chem Phys 121:3417
19. Apra E, Windus TL, Straatsma TP, Bylaska EJ, de Jong W, Hirata S, Valiev M, Hackler M, Pollack L, Kowalski K, Harrison R, Dupuis M, Smith DMA, Nieplocha J, Tipparaju V, Krishnan M, Auer AA, Brown E, Cisneros G, Fann G, Fruchtl H, Garza J, Hirao K, Kendall R, Nichols J, Tsemekhman K, Wolinski K, Anchell J, Bernholdt D, Borowski P, Clark T, Clerc D, Dachsel H, Deegan M, Dyall K, Elwood D, Glendening E, Gutowski M, Hess A, Jaffe J, Johnson B, Ju J, Kobayashi R, Kutteh R, Lin Z, Littlefield R, Long X, Meng B, Nakajima T, Niu S, Rosing M, Sandrone G, Stave M, Taylor H, Thomas G, van Lenthe J, Wong A, Zhang Z (2005) NWChem, A computational chemistry package for parallel computers, Version 4.7 Pacific Northwest National Laboratory, Richland, Washington 99352-0999, USA. High performance computational chemistry: An overview of NWChem a distributed parallel application
20. Kendall RA, Apra E, Bernholdt DE, Bylaska EJ, Dupuis M, Fann GI, Harrison RJ, Ju J, Nichols JA, Nieplocha J, Straatsma TP, Windus TL, Wong AT (2000) Comput Phys Comm 128:260–283
21. GAMESS-UK is a package of ab initio programs written by Guest MF, van Lenthe JH, Kendrick J, Sherwood P with contributions from Amos RD, Buenker RJ, van Dam H, Dupuis M, Handy NC, Hillier IH, Knowles PJ, Bonacic-Koutecky V, von Niessen W, Harrison RJ, Rendell AP, Saunders VR, Schoffel K, Stone AJ, Tozer D
22. Köster AM, Calaminici P, Flores-Moreno R, Geudtner G, Goursot A, Heine T, Janetzko F, Patchkovskii S, Reveles JU, Vela A, Salahub DR (2004) deMon2k, The deMon Developers