Theor Chem Acc (2011) 130:515–530
DOI 10.1007/s00214-011-1036-3

REGULAR ARTICLE

Higher-order correlated calculations based on fragment molecular orbital scheme

Yuji Mochizuki · Katsumi Yamashita · Tatsuya Nakano · Yoshio Okiyama · Kaori Fukuzawa · Naoki Taguchi · Shigenori Tanaka

Received: 8 March 2011 / Accepted: 29 August 2011 / Published online: 5 October 2011

Abstract We have developed a new module for higher-order correlated methods up to coupled-cluster singles and doubles with perturbative triples (CCSD(T)). The matrix-matrix operations through the DGEMM routine were pursued for a number of contractions. This code was then incorporated into the ABINIT-MPX program for the fragment molecular orbital (FMO) calculations. Intra-fragment processings were parallelized with OpenMP in a node-wise fashion, whereas the message passing interface (MPI) was used for the fragment-wise parallelization over nodes. Our new implementation made the FMO-based higher-order calculations applicable to realistic proteins. We have performed several benchmark tests on the Earth Simulator (ES2), a massively parallel computer. For example, the FMO-CCSD(T)/6-31G job for the HIV-1 protease (198 amino acid residues)–lopinavir complex was completed in 9.8 h with 512 processors (or 64 nodes). Another example was the influenza neuraminidase (386 residues) with oseltamivir calculated at the full fourth-order Møller–Plesset perturbation level (MP4), of which job timing was 10.3 h with 1024 processors. The applicability of the methods to commodity cluster computers was tested as well.

Keywords Fragment molecular orbital · FMO · coupled cluster · CC · parallelization · OpenMP · MPI · vectorization · DGEMM · protein

1 Introduction

Kitaura et al. [1] proposed the fragment molecular orbital (FMO) scheme in 1999, by which fully quantum-

Dedicated to Professor Shigeru Nagase on the occasion of his 65th birthday and published as part of the Nagase Festschrift Issue.

Y. Mochizuki (✉) · N. Taguchi
Department of Chemistry and Research Center for Smart Molecules, Faculty of Science, Rikkyo University, 3-34-1 Nishi-ikebukuro, Toshima-ku, Tokyo 171-8501, Japan
e-mail: fullmoon@rikkyo.ac.jp

Y. Mochizuki · T. Nakano · Y. Okiyama · K. Fukuzawa
Institute of Industrial Science, The University of Tokyo, 4-6-1 Komaba, Meguro-ku, Tokyo 153-8505, Japan

Y. Mochizuki · T. Nakano · S. Tanaka
CREST Project, Japan Science and Technology Agency, 4-1-8 Honcho, Kawaguchi, Saitama 332-0012, Japan

K. Yamashita
1st Manufacturing Industries Solutions Division, NEC Soft Ltd., 1-18-6 Shinkiba, Koto-ku, Tokyo 136-8608, Japan

T. Nakano
Division of Medicinal Safety Science, National Institute of Health Sciences, 1-18-1 Kamiyoga, Setagaya-ku, Tokyo 158-8501, Japan

K. Fukuzawa
Mizuho Information and Research Institute Inc., 2-3 Kanda Nishiki-Cho, Chiyoda-ku, Tokyo 101-8443, Japan

S. Tanaka
Graduate School of System Informatics, Department of Computational Science, Kobe University, 1-1 Rokkodai, Nada-ku, Kobe 657-8501, Japan

mechanical (QM) calculations are made possible for large-scale systems just like proteins at the practical cost of computations. FMO-related methodological developments and associated applications have been achieved in diverse ways during this decade [2–4]. In the original two-body treatment [1], the target molecular system is divided into appropriate fragments, and a series of parallelized MO calculations are performed on the fragment monomers and dimers under the presence of environmental electrostatic potential to ensure chemical reliability. The Hartree–Fock (HF) calculations of all monomers are repeated until the self-consistent charge condition is satisfied, describing the electronic polarization of system. The electron delocalizations are then taken into account by the dimer calculations. The inclusion of correlation corrections is straightforward after the HF step [2–4].

The acceleration through an efficient dual-level parallelism is essential to provide the applicability of FMO calculations [2–4]. Namely, the fragment monomers or dimers are distributed over the groups of processors (upper level), and the internal calculation of each fragment is then parallelized for such as atomic orbital (AO) integrals within a given group (lower level). Hence, the FMO scheme has an inherent affinity for massively parallel processors. There are other similar schemes utilizing the concepts of fragmentation and many-body expansion as well as multiple polarization [5–29] (e.g., divide-and-conquer (DC) [7–9], X-Pol [15, 16] and effective fragment potential (EFP) [17, 18]). A rich variety of chemically relevant applications [2–4] beyond demonstrations might rather constitute a relative strength of the FMO scheme, however. Especially, the well-defined fragment-wise interaction energies derived from FMO calculations have been recognized as useful measures to grasp the insight of protein concerning residue–residue and residue–ligand interactions biochemistry and pharmachemistry [4].

In FMO application studies [2–4], the second-order Møller–Plesset perturbation (MP2) calculation [30] has become quite common as the default option to incorporate the electron correlation, because MP2 is the lowest and cheapest many-body theory retaining the size-consistency and unitary-invariance [31]. Nevertheless, MP2 is known to overestimate the dispersion type interactions by certain extent. Hence, beyond-MP2 methods [31, 32], in which the electron pair–-pair interactions and multiple excitations are incorporated as higher-order effects, could thus be desirable. For small molecules, the calculation of coupled-cluster singles and doubles (CCSD) with perturbative triples (CCSD+T(CCSD) [33] or CCSD(T) [34]) has been a golden standard method with flexible basis sets [31].

There have now been three major programs for FMO calculations. From the work of Ref. [35] at the HF level, Fedorov et al. have ever been adapting various standard MO methods implemented in the GAMESS program [36] into the FMO context [2–4]. Up to three-body FMO-MP2 calculations [37, 38] were available in GAMESS. The FMO-CCSD(T) [39] ability was provided as well, while no application to real problems has been performed, to date. The second FMO program, ABINIT-MP, was originally developed by Nakano et al. [40] for the FMO-HF calculations of proteins, by making efficient approximations in computing the electrostatic potentials. After that, Mochizuki et al. augmented a couple of MP2 modules with an integral-direct parallelism [41–43]; the enhanced version was frequently distinguished as ABINIT-MPX. Recently, the third-order MP perturbation (MP3) has been efficiently implemented [44], by which a giant influenza-related protein complex (hemagglutinin (HA) trimer and two Fab-fragments) consisting of more than two thousands residues was computed on a massively vector-parallel computer, the Earth Simulator (ES2)[1]. The third program is PAICS developed by Ishikawa and Kuwata [45] for FMO-MP2. The approaches of integral approximation like resolution-of-identity (RI) or Cholesky decomposition (CD) (refer to a recent review [46]) have attracted considerable interest in accelerating the correlated calculations. References [47–49] reported the MP2 works in this regard.

With the idea of fragmentation or locality, polymers or molecular clusters were treated at the correlated level by several research groups [19, 26, 27, 50–54], besides the FMO-oriented groups; Ref. [19, 26, 27, 52] reported beyond-MP2 calculations. Particularly, noteworthy are the various efforts of the so-called local correlation methods pioneered by Pulay and Saebø [55–57]. This kind of calculations [57–70] is designed to reduce the scaling of computational costs in a domain-specific fashion by the use of localized MOs. Efficient local CC methods [58, 60–64, 67–70] were devised and applied to some demonstrative examples, e.g., water clusters and linear hydrocarbons. Similarly to FMO [2–4], a natural parallelism was pointed out in Refs. [61–63] of the incremental scheme, Refs. [68, 69] of the cluster-in-molecule (CIM) approach and Ref. [70] of the divide-expand-consolidate (DEC) method. Actual applications with these methods are thus expected for larger real systems in the near future.

In this paper, we report the incorporation of a parallelized CC module into ABINIT-MPX [43, 44] for higher-order correlated calculations based on the FMO scheme [1–4]. The practical applicability to real proteins is pursued in our implementation. The fundamental working equations by Scuseria et al. [71] have been adopted for our implementation, as in the work by Kobayashi and Rendell [72] in which a

[1] http://www.jamstec.go.jp/es/en/.

hybrid strategy to use integrals having both MO and AO indices was taken in the parallelization. The DGEMM routine of level-3 BLAS[2] is extensively utilized for a number of contractions among amplitudes, integrals and intermediates. The CCSD(T) calculation [34] is available. Several simplified treatments such as the quadratic configuration-interaction singles and doubles (QCISD) [73] and the full fourth-order MP (MP4) [74] are supported as well. Essential equations in CC and related theories [30–32] are summarized in Sect. 2. Details of the implementation are described in Sect. 3. As in the previous study of FMO-MP3 [44], we take a mixed strategy of message passing interface (MPI)[3] and shared-memory OpenMP[4]. The former takes care of the inter-fragment parallelization, while the latter involves the intra-fragment one. In Sect. 4, several tests on commodity computers are performed, followed by tests on ES2, a massively vector-parallel computer [44]. The largest protein calculated at the FMO-CCSD(T) level on ES2 is the HIV-1 protease (198 amino acid residues) complex with lopinavir comprising 3225 atoms. The FMO-MP4 calculations are also done for larger influenza proteins. Promising timing data and performance will be shown.

2 Essential equations

The CCSD wavefunction is given in an intermediately normalized form [29]

$$|\Psi_{\mathrm{CCSD}}\rangle = \exp(T_1 + T_2)|\Psi_0\rangle, \tag{1}$$

where Ψ_0 is the HF ground state reference with closed-shell occupation. The exponential form of Ψ_{CCSD} ensures the size-consistency or size-extensivity [31, 32]. T_1 and T_2 are the singles and doubles excitation operators, respectively. Letting *ij* and *ab* specify occupied MOs and virtual MOs, respectively, a set of CCSD projection equations [71] is given as

$$\left\langle \Psi_0 | H - E_0 \middle| \left(1 + T_1 + T_2 + \frac{1}{2!}T_1^2\right) \middle| \Psi_0 \right\rangle = E_{\mathrm{CCSD}}, \tag{2}$$

$$\left\langle \Psi_i^a | H - E_0 \middle| \left(1 + T_1 + T_2 + \frac{1}{2!}T_1^2 + T_1T_2 + \frac{1}{3!}T_1^3\right) \middle| \Psi_0 \right\rangle = 0, \tag{3}$$

$$\left\langle \Psi_{ij}^{ab} | H - E_0 \middle| \left(1 + T_1 + T_2 + \frac{1}{2!}T_1^2 + \frac{1}{2!}T_2^2 + T_1T_2 + \frac{1}{3!}T_1^3 + \frac{1}{2!}T_1^2T_2 + \frac{1}{4!}T_1^4\right) \middle| \Psi_0 \right\rangle = 0. \tag{4}$$

[2] http://www.netlib.org/blas/.

[3] http://www.mpi-forum.org/.

[4] http://www.openmp.org/.

Ψ_i^a and Ψ_{ij}^{ab} are the states of single and double excitations, respectively. E_0 is the HF energy, and E_{CCSD} is the CCSD correlation energy. These coupled equations for amplitudes, t_i^a (singles) and t_{ij}^{ab} (doubles), should be iteratively solved by computing the corresponding residual vectors until convergence, where the formal cost scales as N^6 per iteration; N is the molecular size parameter such as the number of basis functions. Based on the generator formalism of spin-adaptation [75, 76], Scuseria et al. [71] derived the explicit projection equations for singles and doubles amplitudes in terms of MO integrals as well as a variety of intermediate arrays, which were introduced to factorize the contractions for efficient computations. The resulting expressions presented in Ref. [71] are essentially equivalent to those of Refs. [77, 78] though such definitions of working arrays are different from each other.

E_{CCSD} is evaluated as [71, 72]

$$E_{\mathrm{CCSD}} = \sum_{ijab}[2(ia,jb) - (ib,ja)]\tau_{ij}^{ab}, \tag{5}$$

$$\tau_{ij}^{ab} = t_{ij}^{ab} + t_i^a t_j^b, \tag{6}$$

where the set of canonical orbitals with ε_i and ε_a as the orbital energies is assumed and also the Mulliken notation [30] is used for the MO integrals. The doubles amplitudes have a convenient symmetry

$$t_{ij}^{ab} = t_{ji}^{ba}, \tag{7}$$

and thus, the restriction of $i \geq j$ can halve the requirements of memory and computation. The first-order MP (MP1) doubles amplitudes

$$t_{ij}^{ab\,(1)} = \frac{(ia,jb)}{\Delta_{ij}^{ab}}, \tag{8}$$

$$\Delta_{ij}^{ab} = \varepsilon_i + \varepsilon_j - \varepsilon_a - \varepsilon_b \tag{9}$$

may be used as an initial vector for the iteration, yielding just the MP2 correlation energy. The costliest computation may be associated with the three- and four-virtual MO integrals for the doubles residual vector [71, 72]

$$\sigma_{ij}^{ab} \leftarrow \sigma_{ij}^{ab} + \sum_{cd} b_{cd}^{ab}\tau_{ij}^{cd} \tag{10}$$

$$b_{cd}^{ab} = (ac,bd) - \sum_k (ac,dk)t_k^b - \sum_k (bd,ck)t_k^a, \tag{11}$$

while the actual processing is done directly from the list of AO integrals, as described in Sect. 3. The a_{ij}^{kl} processing

$$\sigma_{ij}^{ab} \leftarrow \sigma_{ij}^{ab} + \sum_{kl} a_{ij}^{kl}\tau_{kl}^{ab}, \tag{12}$$

$$a_{ij}^{kl} = (ik,jl) + \sum_c (ik,cl)t_j^c + \sum_c (jl,ck)t_i^c + \sum_{cd} (kc,ld)\tau_{ij}^{cd} \tag{13}$$

is relatively cheap, and it may be done with MO integrals. Additionally, we show another couple of intermediate arrays having N^6 cost to construct [71]

$$j_{ic}^{ak} = (ia,kc) - \sum_l (il,ck)t_l^a + \sum_d (ad,ck)t_i^d - \frac{1}{2}\sum_{ld}(kc,ld)\left(t_{il}^{da} + 2t_i^d t_l^a\right) + \frac{1}{2}\sum_{ld}[2(kc,ld) - (kd,lc)]t_{il}^{ad}, \quad (14)$$

$$k_{ic}^{ka} = (ik,ac) - \sum_l (ik,cl)t_l^a + \sum_d (ac,dk)t_i^d - \frac{1}{2}\sum_{ld}(kd,lc)\left(t_{il}^{da} + 2t_i^d t_l^a\right), \quad (15)$$

coupled electron-pair approximation (CEPA) has a close relationship to CC and even size-inconsistent CI. In fact, the introduction of appropriate energy shifts into the residual vector provides CEPA-n ($n = 0 - 3$) [75, 81]. The performance and computational simplicity of CEPA have been re-evaluated recently [82, 83]. From a viewpoint of unitary-invariance, two options of CEPA-0 (linearized version) and CEPA-1 (pair energy-averaged version) are attractive to implement.

Once the CCSD iteration is complete, the triples correction [31–34] may be computed if needed to improve the reliability. This is effective in compensating the degradation from near-degeneracies [80], and the corresponding equation for the closed-shell HF reference has the following form

$$E_{(\mathrm{T})} = \frac{1}{3}\sum_{ijkabc} \frac{\left(W_{ijk}^{abc} + V_{ijk}^{abc}\right)\left(4W_{ijk}^{abc} + W_{kij}^{abc} + W_{jki}^{abc} - 4W_{kji}^{abc} - W_{ikj}^{abc} - W_{jik}^{abc}\right)}{\Delta_{ijk}^{abc}}, \quad (19)$$

where the parts of three-virtual MO integrals can be computed in a fashion similar to Eqs. 10 and 11 [72]. These arrays are contracted with t_{ij}^{ab} for the residual vector (N^6 cost).

As discussed in Refs. [78–80], QCISD [73] is obtained by dropping all nonlinear T_1-related terms in Eqs. 1, 2 and 3, except for T_1 T_2

$$\langle \Psi_0 | H - E_0 | (1 + T_1 + T_2) | \Psi_0 \rangle = E_{\mathrm{QCISD}}, \quad (16)$$

$$\left\langle \Psi_i^a | H - E_0 | (1 + T_1 + T_2 + T_1 T_2) | \Psi_0 \right\rangle = 0, \quad (17)$$

$$\left\langle \Psi_{ij}^{ab} | H - E_0 \left| (1 + T_1 + T_2 + \frac{1}{2!}T_2^2) \right| \Psi_0 \right\rangle = 0. \quad (18)$$

This simplification of QCISD relative to CCSD provides a substantial reduction of computations, especially in making the intermediate arrays. Note that $\frac{1}{2!}T_2^2$ retained for the doubles projection plays a crucial role in providing the size-consistency [30]. The complete omission of T_1 leads to CCD [30, 31] and QCID [73], in which no orbital relaxation effect is taken into account. Lee et al. [80] pointed out that QCISD is vulnerable to too large contribution from singles for the existence of near-degeneracies.

The fourth-order MP with singles, doubles and quadruples, MP4(SDQ), may be calculated similarly, where the doubles amplitudes vector should be updated twice giving the contributions from MP3 and MP4(D) in order [31]. According to Szabo and Ostlund's textbook [30], the

$$\Delta_{ijk}^{abc} = \varepsilon_i + \varepsilon_j + \varepsilon_k - \varepsilon_a - \varepsilon_b - \varepsilon_c, \quad (20)$$

$$W_{ijk}^{abc} = P_{ijk}^{abc}\left[\sum_d (ia,bd)t_{kj}^{cd} - \sum_l (ia,jl)t_{lk}^{bc}\right], \quad (21)$$

$$V_{ijk}^{abc} = (jb,kc)t_i^a + (ia,kc)t_j^b + (ia,jb)t_k^c. \quad (22)$$

P_{ijk}^{abc} in Eq. 21 is the six-fold permutation operator. The restrictions of $i \geq j \geq k$ or $a \geq b \geq c$ can be applied to Eq. 19 through the proper adjustment of pre-factors and index exchanges. (see Ref. [84] for example). The construction of W_{ijk}^{abc} array requires N^7 cost of computation but has no need of iteration. V_{ijk}^{abc} in Eq. 19 is replaced by 2 V_{ijk}^{abc} for QCISD(T) [34, 73]. The +T(CCSD) correction proposed by Urban et al. [33] is obtained only by W_{ijk}^{abc}. Finally, the MP4(T) energy for the full MP4 treatment [74], MP4(SDTQ), is evaluated with the MP1 doubles amplitudes.

3 Implementation

3.1 Overall parallel design of CC module

Prior to describing the overall design of our program module, we would like to brief the precedent papers concerning the parallelized calculations of CC or triples corrections, because they were informative for our implementation. The

UK group reported a series of pioneering studies [84–89] in early 1990s, by using various computers (vector-parallel or massively parallel). Rendell's works [86, 88] were based on the CCSD formulation by Scuseria et al. [71] as in Ref. [72]. Note also that a special middleware was prepared to hold the doubles vectors in a small distributed-memory of parallel machines in those days [88]. In other words, this paper reflected the potential issue how to handle the doubles vectors (consisting of amplitudes and residuals) in the case of large-scale direct CC computations. This issue was solved by the development of global array (GA) library [90], which emulates the shared-memory environment for distributed computing resources. Then, Kobayashi and Rendell [72] utilized GA to develop their direct CC module, where the "*aijkbc*" algorithm of Ref. [89] was adopted for the (T) processing. The use of GA [90] was helpful to avoid I/O interruptions deteriorating the total computational efficiency on massively parallel computers [91]. Koch et al. [92] reported an AO integral-direct CCSD calculation with parallelism.

In 2000s, several research groups made extensive efforts on the development of parallelized CC program systems. It is noteworthy that Piecuch and Landman [93] first introduced the OpenMP parallelization for the CC calculations (of state-universal type) on shared-memory computers. The computerized equation derivation and implementation system (known as TCE) was employed to generate highly correlated and thus complicated CC codes [94–96] with the GA-based parallelization [90]. By the use of an original middleware, distributed data interface (DDI) [97] for GAMESS, Gordon's group [98, 99] achieved an efficiently parallelized CCSD and (T) calculations, in which the management of MO integrals on memory and also the way of task distributions were analyzed well. In contrast, the array-file having a smart I/O utility [100] was used by Pulay et al. [101, 102] to enable large-scale calculations on commodity computers with their program PQS [103]; their largest QCISD calculation involved a total of 1512 AOs of a very flexible basis set for a benzene dimer. The standard MPI environment was used for the parallelization of CFOUR [104, 105], whereas a specially developed symbolic language (SIAL) to handle huge arrays in parallel was used in ACES-III [106, 107]. Currently, the US Pacific-Northwest laboratory group with NWCHEM [108] has been ahead of the huge-scale massively parallel CC computations [109–112]. Reference [111] reported the largest benchmark CCSD(T) calculations of $(H_2O)_{20}$ up to 96000 processors on the Jaguar system, where the tuned version of CC module by Kobayashi and Rendell [72] was employed under the GA environment [90].

The management of doubles vectors (amplitudes and residuals) is a critical concern, as addressed above. Meanwhile, the multi-thread OpenMP technology has become popular as a convenient and efficient parallelization tool on the recent many-core computers with sizable amount of memory per node. For example, a single node of commodity cluster computers can easily be equipped with a couple of quad-core (or hexa-core) chips and tenth of GB memory at low price. The advantage of OpenMP over flat MPI lies on the effective utilization of given memory space and also on the usability of directive-based modifications of loops. Figure 1 illustrates the comparable situation of memory usage in the case of four cores per node. Certainly, the parallelized MO calculations with OpenMP have been reported increasingly [44, 93, 113–119]. In our previous work on FMO-MP3 [44], the fully integral-direct parallelization of MP3 would have required larger memory space than that of MP2, and therefore, the OpenMP parallelization was utilized for the intra-fragment calculations. Hence, we decided to adopt the OpenMP parallelization in our CC module as the first option.

If memory is large enough under the OpenMP environment, up to two-virtual MO integrals can be held in addition to a current pair of doubles vectors (t_{ij}^{ab} and σ_{ij}^{ab}) during the iteration of CCSD or QCISD. In contrast, three- and four-virtual MO integrals may be too demanding to hold furthermore when N grows. As just done by Rendell et al. in Refs. [72, 88], the contributions from these bulky integrals (recall Eqs. 10, 11) could be computed on-the-fly with the AO integrals, through the direct Fock-like contraction [120] named the external exchange operator (EEO) technique. Pople et al. [121] first attempted such a way to escape from the explicit formation of (*ac*, *bd*) for the disk-based MP3 calculation. This approach was later generalized as the EEO or similar algorithms, which had been widely used to process three- or four-virtual MO integrals for correlated calculations [44, 56, 58, 60, 72, 76, 78, 88, 92, 98, 101, 102, 106, 122–124]. Since the EEO part could

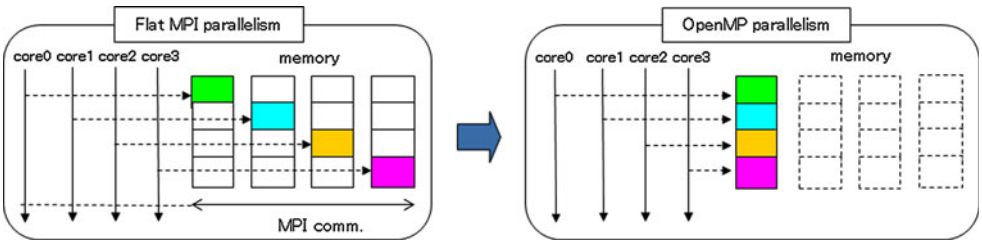

Fig. 1 Illustrative difference in memory usage between flat MPI and OpenMP

Table 1 Four-index arrays in CCSD calculation

Quantity	Dimensioning	Note
Integral		
(ia, jb)	$[b, a, j, i]$	
(ij, ab)	$[ab, ij]$	Canonical ij- and ab-pairs
(ia, jk)	$[k, a, j, i]$	
(ik, jl)	$[l, k, ij]$	Canonical ij-pair
Vector		
t_{ij}^{ab}	$[a, b, ij]$	Amplitude/canonical ij-pair
σ_{ij}^{ab}	$[a, b, ij]$	Residual/canonical ij-pair
EEO-working		
τ_{ij}^{qs}	$[ij, q, s]$	Density-like/reused as ρ_{ik}^{qs}
X_{ij}^{pr}	$[ij, p, r]$	Fock-like/reused as Y_{ik}^{pr}
Z_{ik}^{pq}	$[ik, p, q]$	For k_{ic}^{ka} array (Coulomb)

These arrays are held in shared-memory per node. See text in Sub-Sect. 3.2. FORTRAN style (or column-major) dimensioning is adopted here

govern a sizable portion of CCSD or QCISD calculations as discussed in Ref. [102], the efficient processing should be pursued. A clear merit by the EEO technique has a natural utilization of screenings based not only on the AO integral values but also on the "test density" [122] in order to reduce the computational cost. Another merit is the reduced memory requirements relative to the holding of all MO integrals [72, 88, 98]. By these circumstances, we followed a hybrid integral usage of Refs. [72, 88], in which up to two-virtual MO integrals (including also (ik, jl)) are held on memory for the explicit contractions, while three- and four-virtual MO integrals are processed through the EEO-based implicit contractions (refer to Table 1 later).

The triples correction may be calculated after the convergence of CCSD or QCISD iterations or the MP4(SDQ) calculation. In that stage, there is no need to hold the residual vector for doubles, and hence, a large memory space is released. For the (T) calculation, we thus assumed that the list of three-virtual MO integrals, (ab, ci), can be held on memory with the canonical packing of $a \geq b$, in contrast to the stage of iteration. The "*abcijk*" algorithm [89, 102] was employed, in which the unique *abc* triplets are assigned under the OpenMP parallelization. Working arrays are computed with a full cubic form of *ijk*, and this way may assist the effective usage of the cache area in processors.

We considered that the above-mentioned assumptions for the memory usage of vectors and integrals are acceptable for the case of actual FMO calculations, by which the interaction energies among fragments should be of main chemical interest [2–4]. In such application studies, the 6-31G and 6-31G* basis functions [125] have been typically employed as the sets of valence double-zeta (VDZ) type and its augmented variant with polarization (VDZP). The corresponding N could reach 500–700 for fragment dimers, depending on the combination of amino acid residues in proteins or nucleotides in DNA segments. Nowadays, such sizes of problems would be tractable by the beyond-MP2 methods under the OpenMP parallelization on modern computers. In summary, the overall parallel design of our CC module was oriented to the FMO calculations without I/O processing for a number of contractions. The solid-state disk (SSD) system with high-rate I/O ability has been popularized for commodity computers but still rare for massively parallel computers with which the FMO calculations of very large proteins like influenza HA [44] could be processed in a short time. This is a reason why we continued the parallelized direct style of computations, as in our previous works with ABINIT-MP(X) [40–44].

3.2 Contraction processing

The high affinity of CC and triples calculations with matrix-matrix operations was demonstrated in early works of Refs. [84–89] by the UK group. This should be attributed to the fact that a series of tensorial contractions among amplitudes, integrals and factorized intermediates constitute the main body of beyond-MP2 calculations [31, 32]. Recently, the heavy use of DGEMM has been emphasized in the same context [94, 96, 98, 99, 101, 102, 111]. We definitely followed this strategy for our CC module. The DGEMM library call can be executed in a highly efficient manner on modern commodity processors as well as vector processors.

The list of MO integrals should be generated by the transformations from AO integrals. The integral types of (ia, jb), (ij, ab), (ia, jk) as well as (ik, jl) are prepared and held on memory before the iteration starts. Table 1 lists the four-index arrays needed for CCSD. Note that (ia, jk) is unnecessary in the cases without singles, e.g., CCD. The list of (ia, jb) is most widely used for the tensorial contractions by Refs. [71, 72], and it is obtained as

$$(ia,jb) = \sum_{pqrs} c_{pi} c_{qa} c_{rj} c_{sb} (pq, rs), \tag{23}$$

where $\mathbf{c}$ is the AO-MO coefficient matrix. The transformation is actually done with an integral-direct parallelism in the following quarter steps

$$(iq, rs) = \sum_{p} c_{pi} (pq, rs), \tag{24}$$

$$(ia, rs) = \sum_{q} c_{qa} (iq, rs), \tag{25}$$

$$(ia,js) = \sum_{r} c_{rj} (ia, rs), \tag{26}$$

$$(ia,jb) = \sum_{s} c_{sb} (ia,js). \tag{27}$$

DGEMM can be used in each step on the vector computers [43, 44]; level-1 BLAS of DAXPY and DDOT are also usable upon request to utilize the screenings to reduce actual operation counts for commodity computers [41, 42]. Other MO integrals may be generated similarly. Clearly, the actual dimensioning of integral arrays to be held in the shared-memory environment of OpenMP should be optimized for a series of contractions during the iteration. The array for (ia, jb) has a form of $[b, a, j, i]$, for instance (refer to Table 1). On another front, the doubles vectors (both t_{ij}^{ab} and σ_{ij}^{ab}) are held for the canonical ij-pairs as $[a, b, ij]$.

The correlated methods [30–32] currently available in our CC module are listed as follows.

- MPn: MP2, MP3, MP4(D), MP4(DQ), MP4(SDQ), MP4(SDTQ)
- CC: CCD, CCSD, CCSD(T)
- QCI: QCID, QCISD, QCISD(T)
- CEPA: CEPA-0(D), CEPA-1(D), CEPA-0(SD), CEPA-1(SD)
- CI: CID, CISD (only for test purpose)

The CCSD calculation has both the highest cost of contraction per iteration and the largest memory requirement, where the schematic flow is given in Fig. 2 for later reference. The situation is better for the QCISD calculation, as denoted in Sect. 2. The omission of singles leads to the further reductions of cost and memory, since there is no need to process the contributions from three-virtual MO integrals. These situations may be understandable in the next two paragraphs.

Kobayashi and Rendell [72] well documented how to do the EEO processing with N^6 cost directly from the list of AO integrals, and we would refrain from repeating detailed descriptions, except on a couple of illustrative points. First, the representative EEO computation concerns the b_{cd}^{ab} term of Eqs. 10 and 11 for the doubles residual vector [91],

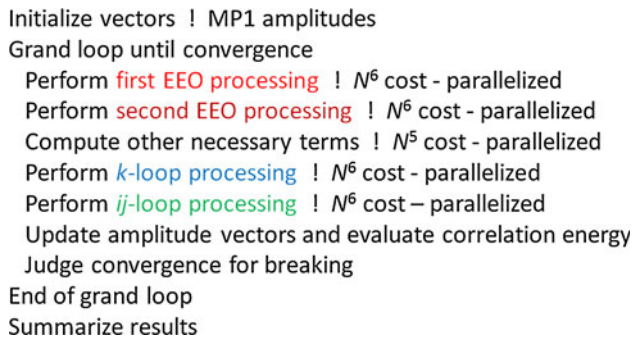

Fig. 2 Schematic flow of CCSD calculation. The second EEO processing is skipped for CCD because of no contribution from three-virtual MO integrals

$$\tau_{ij}^{qs} = \sum_{cd} c_{qc} c_{sd} \tau_{ij}^{cd}, \tag{28}$$

$$X_{ij}^{pr} = \sum_{qs} (pq, rs) \tau_{ij}^{qs}, \tag{29}$$

$$X_{ij}^{ab} = \sum_{pr} c_{pa} c_{rb} X_{ij}^{pr}, \tag{30}$$

$$\sigma_{ij}^{ab} \leftarrow \sigma_{ij}^{ab} + X_{ij}^{ab} - \sum_{k} X_{ij}^{ak} t_k^b - \sum_{k} X_{ij}^{kb} t_k^a. \tag{31}$$

The dimensioning of crucial working arrays are found in Table 1 (see also "first EEO processing" in Fig. 2). The canonical ij-pairs for the contraction in Eq. 29 is processed with DAXPY under the screenings of $pqrs$-quartet and "test density" [122]. Other matrix multiplications is done with DGEMM, in a similar way to the "2h-4p" case of MP3 [44]. The parallelized exchange matrix construction of Eq. 29 is actually performed with respect to shell units of AOs [43, 44]. Equations 28, 30 and 31 of N^5 steps are parallelized over ij. Only the (ac, bd) contribution contracted with t_{ij}^{cd} as X_{ij}^{ab} is required for other methods, QCISD or CCD; an ij-batch mode [44] would be selectable when memory resource is limited. Additionally, X_{ij}^{pr} just formed is usable for the singles residual vector as

$$\sigma_i^a \leftarrow \sigma_i^a + \sum_{kcd} [2(ad, kc) - (ac, kd)] \tau_{ki}^{cd}. \tag{32}$$

Second, some parts of intermediate arrays in CCSD are computed through another EEO processing. We exemplify the third term contributions of j_{ic}^{ak} and k_{ic}^{ka} (Eqs. 14, 15) involving three-virtual MO integrals written, respectively, as

$$j_{ic}^{ak} \leftarrow j_{ic}^{ak} + \sum_{d} (ad, ck) t_i^d, \tag{33}$$

$$k_{ic}^{ka} \leftarrow k_{ic}^{ka} + \sum_{d} (ac, dk) t_i^d. \tag{34}$$

To evaluate these contributions, a pseudo-density

$$\rho_{ik}^{qs} = c_{sk} \sum_{d} c_{qd} t_i^d \tag{35}$$

is prepared (ik is here noncanonical), and the contractions of not only usual exchange type for Eq. 33

$$Y_{ik}^{pr} = \sum_{qs} (pq, rs) \rho_{ik}^{qs} \tag{36}$$

but also coulomb type for Eq. 34

$$Z_{ik}^{pq} = \sum_{rs} (pq, rs) \rho_{ik}^{rs} \tag{37}$$

are carried out [72, 78, 124]. This AO-based Coulomb contraction is needed only for CCSD. It is trivial to do the remaining AO-MO transformations for Eqs. 33 and 34 when needed.

The DGEMM-based contractions involving j_{ic}^{ak} and k_{ic}^{ka} with the amplitude vector of doubles are driven by the outermost k-index for parallelization, where the order of cost is N^6 as denoted in Fig. 2. Two tentative arrays with $[a, c, i]$ dimensioning are formed for a given k, distinguished as $\tilde{k}$, to which the lists of one- and two-virtual MO integrals are used to compute the contributions; recall Eqs. 14 and 15. An example of contractions with these arrays is written as [72]

$$\sigma_{ij}^{ab} \leftarrow \sigma_{ij}^{ab} + \frac{1}{2} P_{ij}^{ab} \sum_{c} \left(2j_{ic}^{a\tilde{k}} - k_{ic}^{\tilde{k}a} \right) \left(2t_{\tilde{k}j}^{cb} - t_{\tilde{k}j}^{bc} \right). \tag{38}$$

P_{ij}^{ab} in this equation is the two-fold permutation operator. Contrastively, the outermost loop of contractions with a_{ij}^{kl}, g_c^a and g_i^k (refer to Refs. [71, 72] for the actual expressions of two g-arrays) is parallelized over the canonical ij-pairs.

When the construction of residual vectors for doubles and singles is complete in a certain iteration, the corresponding norm is checked. The amplitude vectors are straightforwardly updated in a first-order manner. The update for doubles is given as

$$t_{ij}^{ab} = t_{ij}^{ab} + \frac{\sigma_{ij}^{ab}}{\Delta_{ij}^{ab} + \eta}, \tag{39}$$

where an appropriate denominator shift (η) may be introduced [126]. The pair correlation energies for ij are evaluated and summed up for the total correlation energy; this stepwise summation is necessary for CEPA-1 [75, 81]. Both the energy difference and the norm of residuals are monitored to judge the convergence of iteration. To accelerate the convergence, the direct inversion iterative sequence (DIIS) [127, 128] can be used. We set the cycle of DIIS as three as a default option, yielding that the final convergence in six decimal places in energy (au) is usually achieved in 9–12 iterations for CCSD. If memory space is not enough, the noncurrent set of vectors should be written out for some external I/O devices and be read in at the time of DIIS extrapolation. It is here emphasized that any I/O event does not take place during a series of contractions.

After the CCSD or QCISD iterations as well as the MP4(SDQ) calculation, the parallelized direct transformation is performed again to generate three integral types: refer to Eqs. 8, 21 and 22. The dimensioning of (ia, jb) is changed to $[j, i, b, a]$, matching with the "*abcijk*" algorithm of parallelization [89, 102]. The unpacking for ab-pair should be made for (ab, ci) before the crucial N^7-order contraction [98] as the first term in Eq. 21. A total of 12 DGEMM calls are invoked to construct a tentative $W_{ijk}^{\tilde{a}b\tilde{c}}$ array for a given abc-triplet, where six working buffers of $[i, j, k]$ dimensioning are utilized to accumulate the permutation contributions. $V_{ijk}^{\tilde{a}b\tilde{c}}$ of Eq. 22 is prepared trivially. The (T) corrections are actually calculated for the respective virtual MOs (or the outermost a-index), and they are summed up finally. This treatment may be effective in maintaining the numerical tolerance.

3.3 FMO calculation with ABINIT-MPX

Our newly developed CC module was incorporated into ABINIT-MPX [43, 44], under the two-body FMO scheme for energy [1–4]

$$E^{(\mathrm{Total})} = \sum_{I>J} E^{(IJ)} - (M-2) \sum_{I} E^{(I)}. \tag{40}$$

In this Equation, M is the total number of fragments in the target molecular system, and IJ specify the fragment dimers. The intra-node calculations of both monomers and dimers assigned to nodes are processed with the OpenMP parallelism [44].

As addressed in Sect. 1, the list of fragment-wise interaction energies has been extensively used for chemical discussions [4]. This useful measure is called as inter-fragment interaction energy (IFIE) in ABINIT-MP(X) [129, 130] and PAICS [45] or pair interaction energy (PIE) in GAMESS [131]. The IFIE values at the HF level, denoted usually as $\Delta\tilde{E}^{(IJ)}$ [40, 130], is derived by rewriting Eq. 40 as

$$E^{(\mathrm{Total})} = \sum_{I>J} \Delta\tilde{E}^{(IJ)} + \sum_{I} \grave{E}^{(I)}, \tag{41}$$

where the $\grave{E}^{(I)}$ corresponds to the fictitious monomer energy by excluding the contribution from the environmental electrostatic potentials. The differential correlation energy can then be corrected in an additive way. For instance, the corresponding MP2 correction [37, 41, 42] is given as

$$\Delta E_{\mathrm{MP2}}^{(IJ)} = E_{\mathrm{MP2}}^{(IJ)} - E_{\mathrm{MP2}}^{(I)} - E_{\mathrm{MP2}}^{(J)}. \tag{42}$$

The correlated methods for FMO calculations should satisfy the size-consistency [30–32]; the simple method of CI singles and doubles (CISD) is unsuitable. Nakano et al. [40, 132] introduced the dimer-ES approximation in which the energies of fragment dimers consisting of distant monomers are evaluated not with the HF procedure but with the form of only electrostatic interactions. Unquestionably, the correlated calculations are omitted for such dimers whose population might increase for larger systems [2–4]. In other words, a great reduction in computational cost is naturally obtained. Our previous works by FMO-MP2 [43] and FMO-MP3 [44] on huge influenza proteins benefited from this favorable situation. The same is true also for higher-order correlated calculations.

The inner core MOs like 1s are kept frozen after the FMO-HF step for each fragment, as in our Refs. [41–44].

Additionally, we omit the very high-lying virtual MOs (say 10^6 in au) from the correlating space as well. These MOs are byproducts of the projection technique to cut single bond (without H-capping) at a special sp^3-carbon atom [40, 132]. A certain fragment dimer could contain five or ten such virtual MOs, depending on the way of fragmentation [4, 132]. Since the CC module has now provided up to the triples corrections of N^7 cost [31–34], the option of frozen-virtual MOs should reduce the total cost for proteins.

We did not yet introduce the dynamic (or adaptive) task distribution with MPI, and thus, the total efficiency of FMO calculations might be affected by the deviations in granularity due to the sizes of fragment monomers and dimers. On the ES2 system, the set of well-tuned math libraries are supplied, and an OpenMP-conformable environment is available as well. The DGEMM-oriented coding in our CC module would enjoy a high performance by the vectorized executions. The AO integral generator in ABINIT-MPX based on Obara's recursion method [133] had already been vectorized [43, 44]. Care should be taken for the parallelized Fock matrix construction for the EEO parts, in order to prevent the conflicted addressing. As in the case of "2h-4p" in MP3 calculation [44], the buffered index algorithm, initially devised for particle simulations in the plasma physics [134], was adopted [135], where the exchange array (see Eq. 29) should have a dimensioning of [*ij*, *p*, *r*, #*t*] (#*t* means the number of OpenMP threads). When memory resource is limited, the *ij*-pairs are divided for parallelization as an alternative option, leading to a shortening of length for the DAXPY processing. The switching is automatically made by judging from the available memory and number of threads to be invoked. Finally, the usability for commodity processor-based cluster computers is addressed. As pointed out by Janowski et al. [101], the modern processors, e.g., produced by Intel, have been optimized for the matrix-matrix operations with DGEMM. Our CC code should thus run on these computers with a reasonable performance. If possible, more cores and nodes would be desirable for higher-order correlated FMO calculations than those for the MP2 or MP3 calculations [41–44], since the relative costs for each fragment can be enlarged rapidly.

4 Results and discussion

4.1 Benchmark tests on commodity computer

Firstly, the parallel acceleration up to 4 cores was tested for the calculations of CCD, QCISD and CCSD on a single node of a shared-memory computer equipping 2 dual-core Xeon chips with 3.4-GHz clock-rate and 32 GB memory. The standard Intel compiler and math libraries were used.

Table 2 Timings (in minutes) of CCD, QCISD and CCSD calculations for cytosine, isoleucine and glucose, with respect to 1, 2 and 4 cores in a single node [2 dual-core Xeon (3.4-GHz clock-rate), 32 GB shared-memory]

	Cytosine		Isoleucine		Glucose	
	Time (m)	Acc.	Time (m)	Acc.	Time (m)	Acc.
CCD	#10		#10		#10	
1	11.8		72.5		283.3	
2	6.2	1.90	37.7	1.92	146.6	1.93
4	4.0	2.95	25.8	2.81	102.5	2.76
QCISD	#14		#13		#11	
1	35.3		206.8		650.4	
2	18.4	1.92	106.7	1.94	334.5	1.94
4	12.8	2.76	75.5	2.74	241.7	2.69
CCSD	#13		#11		#11	
1	52.7		259.6		950.6	
2	27.3	1.93	133.3	1.95	486.7	1.95
4	18.3	2.88	91.8	2.83	361.5	2.63

Acceleration factors are shown too

The numbers of 6-31G** basis functions [125] for cytosine, isoleucine and glucose were 145, 200 and 240, respectively. The frozen-core restriction was imposed. The number of occupied MOs correlated was thus 21, 27 and 36 in order. The number of iterations needed for a convergence in six decimal places of energy (au) is indicated with sharp symbol (#) as well

Three molecules, cytosine, isoleucine and glucose, were calculated with the 6-31G** basis set [125]. The timings and accelerations are summarized in Table 2, where the time for integral transformation spent in the CC modules is included. From this Table, it is immediately seen that CCSD is several times costly relative to CCD and also that the difference in cost between QCISD and CCSD is sizable. The observed degradation in acceleration from 2 cores to 4 cores may be caused by the 2×2 configuration of chips in this computer (machine dependence). Namely, a competition of memory access to such σ_{ij}^{ab} could take place by the threads of OpenMP parallelization [136]. Still, we consider that the acceleration of 2.6–2.8 by 4 cores is acceptable in practical calculations on modern commodity cluster computers with many-core chips.

Secondly, the relative timings with 4 cores are discussed for several methods with 6-31G** basis [125]. Table 3 shows the corresponding results of glucose and deoxyguanosine. The incremental costs of higher-order calculations relative to MP2 grow considerably, particularly when beyond the MP4(SDQ) level. This fact indicates that more cores per node are needed to use VDZP basis sets for these size of molecules at high-level treatments of correlation. It is expectable that the use of VDZ basis makes the situation less demanding. The full fourth-order MP or MP4(SDTQ) treatment [74] is still faster than the iterative methods

Table 3 Timings (in minutes) of several higher-order methods for glucose and deoxyguanosine, with 4 cores in a single node [2 dual-core Xeon (3.4-GHz clock-rate), 32 GB shared-memory]

	Glucose			Deoxyguanosine		
	Time (m)		Rel.[a]	Time (m)		Rel.[a]
MP2	1.6		1.0	5.1		1.0
MP3	10.9		6.8	59.5		11.7
MP4(DQ)	24.0		15.0	147.1		28.8
MP4(SDQ)	36.7		22.9	218.0		42.7
MP4(SDTQ)	145.2		90.8	878.0		172.2
CCD	102.5	#10	64.1	722.3	#11	141.6
CEPA-1(SD)	267.3	#13	167.1	2004.2	#17	393.0
QCISD	241.7	#11	151.1	1689.7	#13	331.3
QCISD(T)	350.8		219.3	2964.3		581.2
CCSD	361.5	#11	225.9	2167.1	#13	424.9
CCSD(T)	470.4		294.0	3434.2		673.4

The number of 6-31G** basis functions [125] for deoxyguanosine was 350: see Table 2 caption for glucose. The frozen-core restriction was imposed. The number of occupied MOs correlated was thus 51 for deoxyguanosine. The number of iterations needed for a convergence in six decimal places of energy (au) is indicated with sharp symbol (#) for iterative methods

[a] The relative cost to MP2 is indicated

involving singles. Thus, MP4(SDTQ) should be an attractive option in a tradeoff between accuracy and cost, though some care is necessary for near-degeneracies in a given target molecule [31, 32, 80]. Interestingly, the time needed for CEPA-1(SD) is rather longer than that for QCISD, where the number of iterations is incremented.

Thirdly, the FMO calculations with 6-31G basis of the standard VDZ type [125] were performed for a couple of real proteins. One was a complex of the HIV-1 protease and lopinavir; this complex had been employed in our benchmark FMO-MP2 calculations [42, 48]. The lopinavir ligand was divided into four fragments, and a single water molecule was retained in the pharmacophore of protease consisting of two subunits (99 residues per subunit). The total number of fragment was then 203, and the number of atoms and basis functions were 3225 and 17423, respectively. Another protein was the influenza neuraminidase (NA) complex with oseltamivir (well known as Tamiflu™ in markets) [44]. The oseltamivir moiety was treated as a single fragment. The numbers of atoms, residues (fragments) and basis functions were 5792, 386 (378) and 32549, respectively. To examine the applicability of higher-order correlated calculations for proteins, we used a small cluster computer system with 16 nodes, in which each node was equipped with 2 dual-core Xeon processors (3.33-GHz clock-rate) and 16 GB shared-memory (total 64 cores). A single node with 4 cores was supplied for the respective intra-fragment calculations of monomers and dimers under the OpenMP environment. The inter-node communications were controlled by MPI. Because the resources of both memory and number of nodes were limited, no method with singles was attempted. The job timings (including FMO-HF and dimer-ES parts) are listed in Table 4. The MP2 calculation was processed in the CC module just after the integral transformation of (*ia*, *jb*). The relative cost of MP3 to MP2 is 4 for both proteins preferably. The FMO-CCD job for the HIV-1 protease was completed in 149.6 h (or 6.2 days) and the FMO-MP4(DQ) job for the influenza NA in 98.0 h (4.1 days). We think that the FMO-based higher-order calculations have now been made applicable to proteins consisting of a few hundreds residues even on commodity computers, considering the rapid advance of recent many-core technologies as well as the availability of large memory space at low prices.

4.2 Benchmark tests on vector-parallel computer

As in our previous FMO-MP3 works [44], we utilized the ES2 system as one of the most powerful computers accessible for us in Japan. A single node of ES2 was equipped with 8 vector processors (102.4 GFLOPS per processor) and 128 GB shared-memory, and it was assigned to the processing of fragment monomers and dimers. The 6-31G basis set [125] was used throughout primarily due to an allotted time limit of 12 h. Table 5 presents the timing results of benchmark jobs of four proteins, by using 64 nodes (512 processors) and 128 nodes (1024 processors). Both the HIV-1 protease complex and influenza NA complex were tested again. The largest target protein was a partial complex model between influenza HA monomer and Fab-fragment [43, 44], whose numbers of atoms, residues (fragments) and basis functions were 14086, 921 (911) and 78390, respectively. An uniform model protein Trp_{127}+His (3068 atoms and 18659 basis functions), which had been used in Ref. [43], was calculated as well in order to consider the issue of load-balancing [136].

A variety of higher-order correlated calculations were carried out with 64 nodes for the HIV-1 protease–lopinavir complex, as can be seen in the upper entries of Table 5. The left graphics of Fig. 3 is the visualized IFIE results [129, 130] at the highest FMO-CCSD(T) level, showing that the interactions with lopinavir are not limited near the pharmacophore but spread over other parts of the protease. Differences of a few kcal/mol in IFIEs were observed between MP2 and CCSD(T). It is noteworthy that the relative cost factor of FMO-CCSD(T) to FMO-MP2 is as good as 61.4 on the ES2 system, and the corresponding value of QCISD(T) is 52.9. Meanwhile, the FMO-MP4(SDTQ) calculation is less costly with this factor of 21.9. The inclusion of (T) calculation increases the total

Table 4 Timings (in hours) of FMO jobs for the HIV-1 protease–lopinavir complex and the influenza NA–oseltamivir complex on a 16 nodes computer [2 dual-core Xeon (3.33-GHz clock-rate) and 16 GB shared-memory per node, total 64 cores]

	HIV-1		NA	
	Time (h)	Rel.[a]	Time (h)	Rel.[a]
FMO-MP2	3.75	1.0	10.37	1.0
FMO-MP3	14.98	4.0	42.33	4.1
FMO-MP4(DQ)	33.18	8.7	98.03	9.4
FMO-CCD	149.61	39.4	N/A	

The 6-31G basis set [125] was used, and the frozen-core and frozen-virtual restrictions were imposed for each fragment. The convergence threshold in energy was set in six decimal places. See text for molecular information such as the number of atoms

[a] The relative cost to FMO-MP2 is indicated. For NA, the FMO-CCD calculation was not attempted

Table 5 Timings (in hours) of FMO jobs for the HIV-1 protease–lopinavir complex, the influenza NA–oseltamivir complex, the influenza HA with Fab-fragment complex and Trp_{127}+His on 64 or 128 nodes of ES2 [8 vector processors (each speed 102.4 GFLOPS) and 128 GB shared-memory per node [44], 64 nodes and 128 nodes supply total of 512 and 1024 processors, respectively]

	Nodes	Time (h)	Rel.[a]	TFLOPS	Eff.[b] (%)
HIV-1					
FMO-MP2	64	0.16	1.0	1.24	2.36
FMO-MP2[c]	64	0.16	1.0	1.22	2.33
FMO-MP3	64	0.36	2.3	3.40	6.48
FMO-MP3[c]	64	0.37	2.3	2.82	5.38
FMO-MP4(DQ)	64	0.62	3.9	5.70	10.87
FMO-MP4(SDQ)	64	0.85	5.3	4.92	9.38
FMO-MP4(SDTQ)	64	3.51	21.9	13.05	24.89
FMO-CCD	64	2.90	18.1	6.15	11.72
FMO-QCISD	64	5.73	35.8	5.20	9.91
FMO-QCISD(T)	64	8.46	52.9	8.45	16.13
FMO-CCSD	64	7.82	48.9	4.41	8.40
FMO-CCSD(T)	64	9.83	61.4	7.75	14.78
NA					
FMO-MP4(SDQ)	64	2.86		4.26	8.13
FMO-MP4(SDTQ)	128	10.29		15.21	14.50
HA					
FMO-MP4(SDQ)	64	4.70		4.78	9.12
Trp_{127}+His					
FMO-MP4(SDQ)	64	1.79		9.58	18.27
FMO-MP4(SDTQ)	128	7.06		40.46	38.59

The 6-31G basis set [125] was used, and the frozen-core and frozen-virtual restrictions were imposed for each fragment. The convergence threshold in energy was set in six decimal places. See text for molecular information such as the number of atoms

[a] For HIV-1, the relative cost to FMO-MP2 is indicated

[b] Efficiency (in percentage) is defined for the observed gross speed relative to the theoretical peak speed of supplied processors

[c] Custom routines for MP2 [43] and MP3 [44] were used for comparison

performances in efficiency relative to a theoretical peak speed, as expected from the situation that the DGEMM operations for matrix-matrix multiplications dominate completely [84, 85, 89, 102, 111]. The high efficiency of 24.9 % in FMO-MP4(SDTQ) over 9.4 % in FMO-MP4(SDQ) is a demonstrative evidence. The utilization of vector processors in ES2 could provide an advantage in acceleration over commodity processors, based on the enhanced bandwidth in pipelined operations for large matrices. For example, the relative cost of FMO-CCD to FMO-MP2 is 39.4 (or 6.2 days as the job time) for a small cluster computer as found in Table 4, but it is reduced to

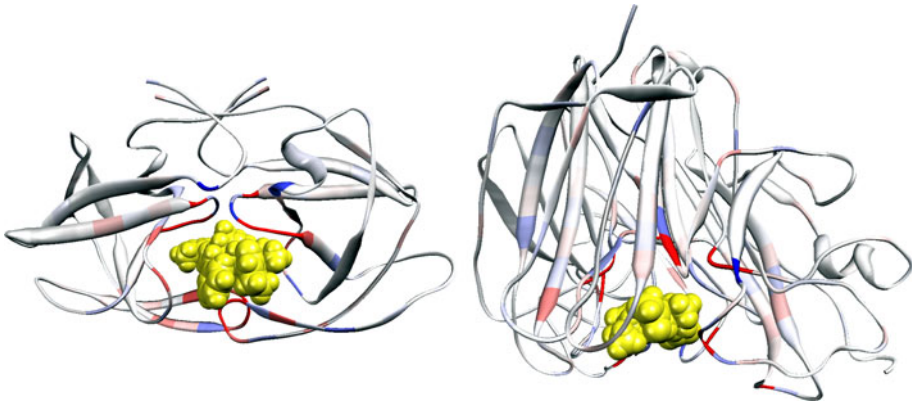

Fig. 3 Visualized IFIE results for the HIV-1 protease complex with lopinavir (*left*) and the influenza NA complex with oseltamivir (*right*). The calculation levels are FMO-CCSD(T)/6-31G for the former and FMO-MP4(SDTQ)/6-31G for the latter. The contributions from 4 fragments in lopinavir were properly summed up. *Red* and *blue* refer to the interaction energies with central drug molecule (drawn with *yellow balls*) in stabilization (negative) and destabilization (positive). The range from −10 to +10 kcal/mol is shown with gradation

18.1 (or only 0.1 days) for ES2. Similar comparisons are valid for FMO-MP4(DQ) as well as FMO-MP3. When the first priority is put on the time-saving, the massively vector-parallel computer should be used if available [43, 44].

As just discussed in the above paragraph, the FMO-based higher-order calculations up to CCSD(T) would be tractable for the complex of HIV-1 protease consisting of 198 residues. The intrinsic low-scaling nature of the FMO scheme [2–4] is responsible for this favorable achievement, though the high computational power of ES2 should also be contributive. It is therefore interesting to see the applicability to a couple of influenza proteins having larger sizes. The FMO-MP4(SDQ) job timings in Table 5 are 2.9 h for the NA–oseltamivir complex and 4.7 h for the HA plus Fab-fragment, with 512 processors. For the former, the FMO-MP4(SDTQ) job was furthermore executed with 1024 processors, and the resulted timing was 10.3 h with a reasonable efficiency. The right graphics in Fig. 3 presents the interaction energies with oseltamivir at this level of calculation. Supposing that the scalability from 64 nodes to 128 nodes is 2, the timing of FMO-MP4(SDTQ) with 64 nodes might be 20.6 h, which is 7.1 times longer than that of 2.9 h FMO-MP4(SDQ). In contrast, the observed ratio for the HIV-1 protease is only 4.1, implying that the incremental cost for triples depends on the size of target proteins.

The load-imbalance has been well known as a critical factor to degrade the total performance in parallel processings [136]. In the case of FMO calculations, the differences in fragment sizes causes such a degradation, as addressed in SubSect. 3.3. The FMO-MP4(SDTQ) job for Trp_{127}+His recorded a notably high efficiency of 38.6 % for 1024 processors (refer to Table 5), and the efficiency for FMO-MP4(SDQ) was observed as favorable as 18.3 %. These values of Trp_{127}+His shed light on the importance of load-balancing and prove the potential of kernel modules in our ABINIT-MPX at the same time. It is fair to note that GAMESS has an advantage in controlling the task distributions in an adaptive fashion with the generalized DDI (GDDI) [35, 97]. The load-balancing for fragments has been a long-term issue for ABINIT-MPX [43, 44].

In our previous study based on the custom MP3 module [44], a full influenza antigen–antibody model consisting of HA trimer and two Fab-fragments was computed on the ES2 system, where the molecular size was 2.6 times larger (2325 fragments and 201276 basis functions) than that of the HA monomer model employed presently and also in Ref. [43]. The residue-specific recognition mechanism as well as the mutation possibilities [137, 138] has been analyzed in detail [139]. In such analyses, the MP2.5 approach [140], in which the third-order correlation contributions are scaled by 0.5, was used in evaluating IFIE values, since it could provide a better correspondence to the results of CCSD(T) than that by the naive MP3. Surely, the FMO-CCSD(T) job should be too demanding for such a giant protein even with the ES2 system.

4.3 Future direction

Besides the load-balancing, efforts are still necessary to enhance a total applicability of higher-order calculations to real problems. Three concerns are addressed as below.

The timings of benchmark calculations have just shown that CCD is roughly 3–4 times cheaper than CCSD, as found in Tables 2, 3 and 5. This fact is consistent with the reduced complexity of equations [78–80, 101], omitting the orbital relaxations due to singles. The method of Brueckner doubles (BD) is a macro-iterative CCD approach with updating the MOs occupied in the Brueckner determinant, by which the relaxation effect with some resistance against

near-degeneracies can be taken into account [80, 141–143]. In BD, the orbital rotation parameters is obtained as the solution of effective singles equation containing the doubles amplitudes just determined by the CCD step. Testing with a prototype BD code indicated that a few macro-iterations are usually enough for a practical convergence. Namely, a total computational cost of BD may be comparable to that of CCSD for which more memory is required to process the three-virtual MO integrals through the EEO route [72]. BD with perturbative triples (BD(T) [142, 143]) is straightforward to do, where there is no need for the list of (*ia*, *jb*). The modification works for BD and BD(T) have been underway.

We had developed a set of routines related to the CD integral approximation [46]. They were utilized for FMO-MP2 calculations with an acceleration of ten times [48], where the (*ia*, *jb*) integrals were factorized. References [144–146] reported the applications of CD techniques to CC calculations with promising results. Therefore, the CD-based integral processing is worth to consider as one of future extensions in ABINIT-MPX, especially for the use of basis sets of better quality (e.g., 6-311G** [125]). An alternative route to make the incremental costs for CC calculations moderate may be the utilization of modified virtual MOs. Namely, the HF virtual MOs might be pre-transformed to the optimized correlating MOs, and the primary subset with substantial contributions would be used at a small loss of accuracy [147].

The basis set superposition error (BSSE) [20, 148] is particularly influential for dispersion type interactions that are relevant in proteins as well as DNA. Ishikawa and Kuwata [45] first provided the counterpoise correction (CP) for FMO-MP2. Very recently, we [149] have implemented the CP procedure at the FMO-MP3 level [44] (including MP2.5 [140]). In Ref. [149], sizable differences were found not only in the interaction energies with and without BSSE-CP corrections but also in the results obtained by MP2 and MP2.5 treatments. The implementation of CP procedure with the general CC module is a future issue.

5 Summary

In this paper, we reported the development of parallelized direct CC module in our ABINIT-MPX [43, 44], by which a variety of higher-order correlated calculations [30–32] were made available in conjunction with the FMO scheme [1–4]. A mixed integral usage with MO and AO indices was adopted for a series of tensorial contractions [72]. The intra-node parallelization was done with the thread-based OpenMP for the efficient usage of memory. The kernel processing of matrix-matrix multiplications was done with DGEMM. The standard MPI control was used to distribute the tasks of fragments (monomers and dimers) over computing nodes. Namely, a hybrid parallelism was taken as in Ref. [44]. Some test calculations without FMO scheme first showed reasonable performance on a single node of commodity computer. A small cluster computer with 64 cores (4 cores per node and total 16 nodes) was used for the FMO calculations on the HIV-1 protease–lopinavir complex (203 fragments) [41, 48] as well as the influenza NA–oseltamivir complex (378 fragments) [44]. The FMO-CCD/6-31G job for the former was completed in 149.6 h (6.2 days), while the FMO-MP4(DQ)/6-31G job for the latter was finished in 98.0 h (4.1 days). These timing data suggested an applicability of beyond-MP2 calculation with the FMO scheme to proteins even on commodity computers. ES2, a massively vector-parallel computer, provided a considerable acceleration in processing and allowed the inclusion of singles as well as triples. With the 6-31G basis set, various treatments were applied to the HIV-1 protease complex. The highest-level FMO-CCSD and FMO-CCSD(T) jobs were completed in 7.8 and 9.8 h, respectively, with 512 processors (8 processors per node and total 64 nodes), indicating that the cost of (T) calculations is quite accessible on the vector computer. More approximated methods were performed less costly (e.g., 5.7 h for QCISD). The FMO-MP4(SDQ)/6-31G calculation was applied to a model complex of influenza HA monomer with Fab-fragment (911 fragments) [43], and the corresponding timing with 512 processors was 4.7 h. The FMO-MP4(SDTQ)/6-31G job for the NA–oseltamivir complex was completed in 10.3 h with 1024 processors (128 nodes). The same job for the Trp_{127}+His model protein achieved a remarkable efficiency of 38.6 % relative to the theoretical peak performance. These test calculations proved a high usability of the FMO-based higher-order calculations for proteins with ABINIT-MPX. As documented by de Jong et al. [150], extensive efforts by theoretical chemists around the world have been conducted toward the peta-scale computational chemistry through the advanced parallel processing technologies. We should continue works to enhance the ability of our CC module. For instance, the implementation of BD and BD(T) [141–143] is in progress. The improvement of task distribution for fragments is another issue when the number of available processors/nodes grows dozens of times in near future. In this regard, the use of smaller units in fragmentation, e.g., through the separation of main chains and side chains in amino acid residues, may be convenient to control the granularity of tasks for better load-balance, where up to four-body corrections (FMO4) are to be incorporated to maintain chemical accuracy of calculations [151]. The reduction of fragment size may be helpful to perform higher-order calculations as well. Finally, we expect the

developments of multi-node OpenMP environment [152, 153], by which the intra-fragment processing over nodes is enabled.

Acknowledgments This work was supported by the CREST project operated by the Japan Science and Technology Agency (JST) and also by a Grant-in-Aid for Scientific Research on Priority Area "Molecular Theory for Real Systems" from the Ministry of Education, Culture, Sports, Science and Technology of Japan (MEXT). YM and TN are indebted for various supports by the RISS project at the Institute of Industrial Science (IIS) of the University of Tokyo. YM acknowledges the SFR-aid by Rikkyo University as well. Computing resource for all the calculations on the Earth Simulator (ES2) was supplied by the Japan Agency for Marine-Earth Science and Technology (JAMSTEC). The authors thank Dr. Yuto Komeiji for critical comments on the manuscript. Finally, YM is grateful to the late Dr. Hideki Katagiri, who passed away in December 2010, for stimulating discussions on CC theories at the early stage of this work.

References

1. Kitaura K, Ikeo E, Asada T, Nakano T, Uebayasi M (1999) Chem Phys Lett 313:701
2. Fedorov DG, Kitaura K (2007) J Phys Chem A 111:6904
3. Gordon MS, Mullin JM, Pruitt SR, Roskop LB, Slipchenko LV, Boatz JA (2009) J Phys Chem B 113:9646
4. Fedorov, DG, Kitaura, K (eds) (2009) The fragment molecular orbital method: practical applications to large molecular systems. CRC Press, Boca Raton
5. Imamura A, Aoki Y, Maekawa K (1991) J Chem Phys 95:5419
6. Aoki Y, Imamura A (1992) J Chem Phys 97:8432
7. Yang W (1991) Phys Rev Lett 66:1438
8. Akama T, Kobayashi M, Nakai H (2007) J Comput Chem 28:2003
9. He X, Merz KM Jr (2010) J Chem Theor Comp 6:405
10. Babu K, Gadre SR (2003) J Comput Chem 24:484
11. Zhang DW, Zhang JZ (2003) J Chem Phys 119:3599
12. Huang L, Massa L, Karle J (2005) Int J Quant Chem 103:808
13. Li W, Fang T, Li S (2006) J Chem Phys 124:154102
14. Li W, Li S, Jiang Y (2007) J Phys Chem A 111:2193
15. Gao J (1997) J Phys Chem B 101:657
16. Song L, Han J, Lin YI, Xie W, Gao J (2009) J Phys Chem A 113:11656
17. Day NP, Jensen HJ, Gordon SM, Webb PS (1996) J Chem Phys 105:1968
18. Nagata T, Fedorov DG, Kitaura K, Gordon MS (2009) J Chem Phys 131:024101
19. Hirata S, Valiev M, Dupuis M, Xantheas SS, Sugiki S, Sekino H (2005) Mol Phys 103:2255
20. Kamiya M, Hirata S, Valiev M (2008) J Chem Phys 128:074103
21. Dahlke EE, Truhlar DG (2007) J Chem Theory Comput 3:46
22. Wang B, Truhlar DG (2010) J Chem Theor Comp 6:359
23. Beran GJO (2009) J Chem Phys 130:164115
24. Söderhjelm P, Ryde U (2009) J Phys Chem A 113:617
25. Mayhall NJ, Raghavachari K (2011) J Chem Theory Comp 7:1336
26. Stoll H (1992) Chem Phys Lett 191:548
27. Paulus B, Fulde P, Stoll H (1995) Phys Rev B 51:10572
28. Seijo L, Barandiarán Z (2004) J Chem Phys 121:6698
29. Seijo L, Barandiarán Z, Soler JM (2007) Theor Chem Acc 118:541
30. Szabo A, Ostlund NS (1982) Modern quantum chemistry. MacMillan, New York
31. Shavitt I, Bartlett RJ (2009) Many-body methods in chemistry and physics. Cambridge Univeristy Press, Cambridge
32. Bartlett RJ, Musiał M (2007) Rev Mod Phys 79:291
33. Urban M, Noga J, Cole SJ, Bartlett RJ (1985) J Chem Phys 83:4041
34. Raghavachari K, Trucks GW, Pople JA, Head-Gordon M (1989) Chem Phys Lett 157:479
35. Fedorov DG, Olson RM, Kitaura K, Gordon MS, Koseki S (2004) J Comp Chem 25:872
36. Schmidt MW, Baldridge KK, Boatz JA, Elbert ST, Gordon MS, Jensen JH, Koseki S, Matsunaga N, Nguyen KA, Su SJ, Windus TL, Dupuis M, Montgomery JA (1993) J Comp Chem 14:347 GAMESS
37. Fedorov DG, Kitaura K (2004) J Chem Phys 121:2483
38. Fedorov DG, Ishimura K, Ishida K, Kitaura K, Pulay P, Nagase S (2007) J Comp Chem 28:1476
39. Fedorov DG, Kitaura K (2005) J Chem Phys 123:134103
40. Nakano T, Kaminuma T, Sato T, Fukuzawa K, Akiyama Y, Uebayasi M, Kitaura K (2002) Chem Phys Lett 351:475
41. Mochizuki Y, Nakano T, Koikegami S, Tanimori S, Abe Y, Nagashima U, Kitaura K (2004) Theor Chem Acc 112:442
42. Mochizuki Y, Koikegami S, Nakano T, Amari S, Kitaura K (2004) Chem Phys Lett 396:473
43. Mochizuki Y, Yamashita K, Murase T, Nakano T, Fukuzawa K, Takematsu K, Watanabe H, Tanaka S (2008) Chem Phys Lett 457:396
44. Mochizuki Y, Yamashita K, Fukuzawa K, Takematsu K, Watanabe H, Taguchi N, Okiyama Y, Tsuboi M, Nakano T, Tanaka S (2010) Chem Phys Lett 493:346
45. Ishikawa T, Kuwata K (2009) J Comp Chem 30:2594
46. Pedersen TB, Aquilante F, Lindh R (2009) Theor Chem Acc 124:1
47. Ishikawa T, Kuwata K (2009) Chem Phys Lett 474:195
48. Okiyama Y, Nakano T, Yamashita K, Mochizuki Y, Taguchi N, Tanaka S (2010) Chem Phys Lett 490:84
49. Rahalkar AP, Katouda M, Gadre SR, Nagase S (2010) J Comp Chem 31:2405
50. Kobayashi M, Akama T, Nakai H (2006) J Chem Phys 125:204106
51. Kobayashi M, Imamura Y, Nakai H (2007) J Chem Phys 127:074103
52. Kobayashi M, Nakai H (2009) J Chem Phys 129:044103
53. Makowski M, Korchowiec J, Gu FL, Aoki Y (2010) J Comp Chem 31:1733
54. Ohnishi Y, Hirata S (2010) J Chem Phys 133:034106
55. Pulay P (1983) Chem Phys Lett 100:151
56. Saebø S, Pulay P (1987) J Chem Phys 86:914
57. Saebø S, Pulay P (1993) Ann Rev Phys Chem 44:213
58. Hampel C, Werner HJ (1996) J Chem Phys 104:6286
59. Schütz M, Hetzer G, Werner HJ (1999) J Chem Phys 111:5691
60. Schütz M, Werner HJ (2001) J Chem Phys 114:661
61. Friedrich J, Hanrath M, Dolg M (2007) J Chem Phys 126:154110
62. Friedrich J, Dolg M (2008) J Chem Phys 129:244105
63. Friedrich J, Dolg M (2009) J Chem Theory Comp 5:287
64. Scuseria GE, Ayala PY (1999) J Chem Phys 111:8330
65. Subotnik JE, Head-Gordon M (2005) J Chem Phys 123:064108
66. Subotnik JE, Sodt A, Head-Gordon M (2008) J Chem Phys 128:034103
67. Li S, Ma J, Jiang Y (2002) J Comput Chem 23:237
68. Li W, Piecuch P, Gour JR, Li S (2009) J Chem Phys 131:114109
69. Li W, Piecuch P (2010) J Phys Chem A 114:8644
70. Ziółkowski M, Jansík B, Kjaergaard T, Jørgensen P (2010) J Chem Phys 133:014107
71. Scuseria GE, Janssen CL, Schaefer HF III (1988) J Chem Phys 89:7382

72. Kobayashi R, Rendell AP (1997) Chem Phys Lett 265:1
73. Pople JA, Head-Gordon M, Raghavachari K (1987) J Chem Phys 87:5968
74. Pople JA, Head-Gordon M, Raghavachari K (1988) Int J Quant Chem Symp 22:377
75. Meyer W (1976) J Chem Phys 64:2901
76. Pulay P, Saebø S, Meyer W (1984) J Chem Phys 81:1901
77. Koch H, Jensen HJA, Jørgensen P, Helgaker T, Scuseria GE, Schaefer HF III (1990) J Chem Phys 92:4924
78. Hampel C, Peterson KA, Werner HJ (1992) Chem Phys Lett 190:1
79. Scuseria GE, Schaefer HF III (1989) J Chem Phys 90:3700
80. Lee TJ, Rendell AP, Taylor PR (1990) J Phys Chem 94:5463
81. Ahlrichs R (1979) Comp Phys Comm 17:31
82. Neese F, Hansen A, Wennmohs F, Grimme S (2009) Acc Chem Res 42:641
83. Taube AG, Bartlett RJ (2009) J Chem Phys 130:144112
84. Rendell AP, Lee TJ, Komornicki A (1991) Chem Phys Lett 178:462
85. Baker DJ, Moncrieff D, Saunders VR, Wilson S (1991) Comp Phys Comm 62:25
86. Rendell AP, Lee TJ, Lindh R (1992) Chem Phys Lett 194:84
87. Moncrieff D, Saunders VR, Wilson S (1992) Comp Phys Comm 70:345
88. Rendell AP, Guest MF, Kendall RA (1993) J Comp Chem 1993(14):1429
89. Rendell AP, Lee TJ, Komornicki A, Wilson S (1993) Theor Chim Acta 84:271
90. Nieplocha J, Harrison RJ, Littlefield RJ (1996) J Supercomp 10:169
91. Watts JD (2000) Para Comp 26:857
92. Koch H, Sánchezde Merás A, Helgaker T, Christiansen O (1996) J Chem Phys 104:4157
93. Piecuch P, Landman JI (2000) Para Comp 26:913
94. Hirata S (2003) J Phys Chem A 107:9887
95. Piecuch P, Hirata S, Kowalski K, Fan PD, Windus TL (2006) Int J Quant Chem 106:79
96. Auer AA, Baumgartner G, Bernholdt DE, Bibireata A, Choppella V, Cociorva D, Gao X, Harrison R, Krishanmoorthy S, Krishnan S, Lu Q, Lam C, Nooijen M, Pitzer R, Ramanujam J, Sadayappan P, Sibiryakov A (2006) Mol Phys 104:211
97. Fletcher GD, Schmidt MW, Bode BM, Gordon MS (2000) Comp Phys Comm 128:190
98. Olson RM, Bentz JL, Kendall RA, Schmidt MW, Gordon MS (2007) J Chem Theory Comp 3:1312
99. Bentz JL, Olson RM, Gordon MS, Schmidt MW, Kendall RA (2007) Comp Phys Comm 176:589
100. Ford AR, Janowski T, Pulay P (2007) J Comp Chem 28:1215
101. Janowski T, Ford AR, Pulay P (2007) J Chem Theory Comp 3:1368
102. Janowski T, Pulay P (2008) J Chem Theory Comp 4:1585
103. Baker J, Wolinski K, Malagoli M, Kinghorn D, Wolinski P, Magyarfalvi G, Saebo S, Janowski T, Pulay P (2009) J Comp Chem 30:317
104. Harding ME, Metzroth T, Gauss J, Auer AA (2008) J Chem Theory Comp 4:64
105. Prochnow E, Harding ME, Gauss J (2010) J Chem Theory Comp 6:2339
106. Lotrich V, Flocke N, Ponton M, Yau AD, Perera A, Deumens E, Bartlett RJ (2008) J Chem Phys 128:194104
107. Tomasz Kuś, Lotrich VF, Bartlett RJ (2009) J Chem Phys 130:124122
108. Valiev M, Bylaska EJ, Govind N, Kowalski K, Straatsma TP, Van Dam HJJ, D. Wang, Nieplocha J, Apra E, Windus TL, de Jong WA (2010) Comp Phys Comm 181:1477
109. Fan PD, Valiev M, Kowalski K (2008) Chem Phys Lett 458:205
110. Kowalski K, Hammond JR, DeJong WA, Sadlej AJ (2008) J Chem Phys 129:226101
111. Aprà E, Harrison RJ, Shelton WA, Tipparaju V, Vázquez-Mayagoitia A (2009) J Phys Conf Ser 180:012027
112. Van Dam HJJ, Vishnu A, de Jong WA (2011) J Chem Theor Comp 7:66
113. Sosa CP, Scalmani G, Gomperts R, Frisch MJ (2000) Para Comp 26:843
114. Shellman SD, Lewis JP, Glaesemann KR, Sikorski K, Voth GA (2003) J Comp Phys 188:1
115. Rudberg E, Rubensson EH, Sałek P (2008) J Chem Phys 128:184106
116. Kurashige Y, Yanai T (2009) J Chem Phys 130:234114
117. Woods CJ, Brown P, Manby FR (2009) J Chem Theor Comp 5:1776
118. Ishimura K, Kuramoto K, Ikuta Y, Hyodo S (2010) J Chem Theor Comp 6:1075
119. Hohenstein EG, Sherrill CD (2010) J Chem Phys 133:014101
120. Almlöf J, Faegri K, Korsell K (1982) J Comp Chem 3:385
121. Pople JA, Binkley S, Seeger R (1976) Int J Quant Chem Symp 10:1
122. Taylor PR (1987) Int J Quant Chem 31:521
123. Saebø S, Almlöf J (1989) Chem Phys Lett 154:83
124. Schütz M, Lindh R, Werner HJ (1999) Mol Phys 96:719
125. Foresman JB, Frisch A (1996) Exploring chemistry with electronic structure methods, 2nd edn. Gaussian Inc., Pittsburgh
126. Szakács P, Surján P (2008) Intern J Quant Chem 108:2043
127. Pulay P (1980) Chem Phys Lett 73:393
128. Scuseria GE, Lee TJ, Schaefer HF III (1986) Chem Phys Lett 130:236
129. Mochizuki Y, Fukuzawa K, Kato A, Tanaka S, Kitaura K, Nakano T (2005) Chem Phys Lett 410:247
130. Amari S, Aizawa M, Zhang J, Fukuzawa K, Mochizuki Y, Iwasawa I, Nakata K, Chuman H, Nakano T (2006) J Chem Inf Model 46:221
131. Fedorov DG, Kitaura K (2007) J Comp Chem 28:222
132. Nakano T, Kaminuma T, Sato T, Akiyama Y, Uebayasi M, Kitaura K (2000) Chem Phys Lett 318:614
133. Obara S, Saika A (1986) J Chem Phys 84:3963
134. Nishiguchi A, Orii S, Yabe T (1985) J Comp Phys 61:519
135. Mochizuki Y, Matsumura M, Yokura T, Hirahara Y, Imamura T (2002) J Nucl Sci Tech 39:195
136. Janssen CL, Nielsen IMB (2008) Parallel computing in quantum chemistry. CRC Press, Boca Raton
137. Iwata T, Fukuzawa K, Nakajima K, Aida-Hyugaji S, Mochizuki Y, Watanabe H, Tanaka S (2008) Comp Bio Chem 32:198
138. Takematsu K, Fukuzawa K, Omagari K, Nakajima S, Nakajima K, Mochizuki Y, Nakano T, Watanabe H, Tanaka S (2009) J Phys Chem B 113:4991
139. Yoshioka A, Fukuzawa K, Mochizuki Y, Yamashita K, Nakano T, Okiyama Y, Nobusawa E, Nakajima K, Tanaka S (2011) J Mol Graph Modell 30:110
140. Pitoňák M, Neogrády P, Černý J, Grimme S, Hobza P (2009) Chem Phys Chem 10:282
141. Chiles RA, Dykstra CE (1981) J Chem Phys 74:4544
142. Handy NC, Pople JA, Head-Gordon M, Raghavachari K, Trucks GW (1989) Chem Phys Lett 164:185
143. Scuseria GE (1994) Chem Phys Lett 226:251
144. Koch H, Sánchez de Merás A, Pedersen TB (2003) J Chem Phys 21:1981
145. Lee TJ, Huang X, Dateo CE (2009) Mol Phys 107:1139
146. Sánchezde Merás AMJ, Koch H, Cuesta IG, Boman L (2010) J Chem Phys 132:204105
147. Neogrády P, Pitoňák M, Urban M (2005) Mol Phys 103:2141
148. Boys SF, Bernardi F (1970) Mol Phys 19:533

149. Okiyama Y, Fukuzawa K, Yamada H, Mochizuki Y, Nakano T, Tanaka S (2011) Chem Phys Lett 509:67
150. de Jong WA, Bylaska E, Govind N, Janssen CL, Kowalski K, Müller T, Nielsen IMB, van Dam HJJ, Veryazov V, Lindh R (2010) Phys Chem Chem Phys 6:6896
151. Nakano T, Mochizuki Y, Yamashita K, Watanabe C, Fukuzawa K, Segawa K, Okiyama Y, Tsukamoto T, Tanaka S (to be published)
152. Cai J, Rendell AP, Strazdins PE, Wong HJ (2008) Proceeding of 13th IEEE Asia-Pacific computer systems architecture conference:1
153. Yang R, Cai J, Rendell AP, Ganesh V (2009) International Workshop on OpenMP 2009, LNCS 5568:53